Using Standard Cases and Pairwise Comparison to Assess the Validity of Software Size Metrics

Gavin Finnie[1]
Eberhard Rudolph[2]
Gerhard Wittig[1]

[1] Bond University
Australia
Email: gfinnie@bond.edu.au

[2] Hochschule Bremerhaven
Germany

Abstract

A variety of metrics have been proposed to measure the size of software, including Function Points, Mk2 Function Points and Full Function Points (FFP). This paper describes research in progress that is attempting to assess the extent to which user perceptions of functionality mirror the metrics used to measure software size. Users and developers have some intuitive understanding of software size, at least in comparing one project to another. This project aims to quantify these perceptions. In the first instance, these results will form part of the COSMIC group initiative that is performing a range of trials to assess the validity of the FFP approach.

Keywords

Software Metrics; Function Point Analysis; Analytic Hierarchy Process

INTRODUCTION

Accurate and effective prediction of software project size has proved an elusive goal for software developers. Without a practical measure of size, prediction of development effort becomes very difficult. Reliable prediction of software development effort from the system size is a necessary prerequisite to effective project planning and control.

A wide variety of software metrics have been proposed. Of these, Function Points (FP) (Albrecht, 1983) and Mk2 Function points (Symons, 1988, 1991) have probably had the widest use. However the Function Point approach has been criticised on several grounds, including the fact that the original methodology was based on MIS type projects and hence biased towards data processing applications. More recently, Full Function Points (FFP) have attracted considerable attention (Abran, 1999; Morris and Desharnais, 1998; Oligny at al, 1999) as a potential methodology, which could be used for estimating several categories of software project. The COSMIC (Common Software Measurement International Consortium) research effort (Abran, 1999; Symons, 1999) is currently performing a range of trials to assess the validity of the approach for different types of software and to fine tune the method's measurement function if necessary.

For a software size metric to be accepted in the user community, it should conform to the user's perception of software size i.e. if a user perceives one system to be twice the size of

another, the software metric should reflect similar proportions. This paper describes a research project that will compare perceived user functionality of a number of small software projects and the functional size of these projects as measured using several metrics ie FFP, FPA and Mk2. A sample of "users/developers" with some software development experience will be asked to complete a survey, using either a manual questionnaire or an interactive data collection program. The comparison of each project's functionality is done on a pair-wise basis and the results are analysed using the Analytic Hierarchy Process (AHP). The project has undergone a pilot test to refine the measurement instrument and more complete data will be gathered from October 2000.

The use of pair-wise comparison is widely accepted in the social sciences as a means of establishing the relative merit of an item with respect to others of the same type. Finnie et al (1996) proposed the use of AHP in conjunction with expert judgement to assess software development effort for new projects. More recently, Miranda (1999) proposed AHP to provide a means of establishing software size by paired comparisons. In this approach, the projects to be estimated would be sized using UML Use Cases. Provided at least one developed project exists with a known Use Case measure to act as an anchor, the other projects can be estimated on the basis of their ratio to the known case.

DEFINING FUNCTIONALITY

Function Points and Functionality

The measurement of user perceived functionality requires some anchor to provide a framework for functionality. In earlier research on functionality and function point analysis, the authors provide a definition of functionality that was framed around the function point analysis approach (Rudolph et al (1998), Wittig at al (1999)).

In terms of the function point analysis method, software comprises processes and data. This required that the functionality provided by both these components had to be defined. The definition included:

(1) The number of tasks necessary to complete the function, ie the number of decisions needed to be made or the number of questions that need to be asked

(2) the amount of information that needs to be handled by the function, ie. written down, analysed, received or produced

(3) The skill level of the person required to complete the function, ie their:
- capability to understand the problem
- capability to perform the necessary calculations and algorithms
- relevant experience with similar problems
- knowledge of the correct rules to be applied to the problem

(4) File functionality which could be viewed as a measure to include one or more of the following:
- the amount of information that is contained in one file record
- the capacity of the record to provide a range of information
- the capacity of the file to be able to cross reference data in another file

This definition was framed in the context of function point analysis and could suffer from bias because of that. For the current research it was felt that a more general approach would be of value and a model based on the ISO proposed standard 14143-5 has been proposed. Although this ISO standard is currently a work in progress document and will likely undergo

some revision, it does provide a research framework which can be used as a basis for defining functionality.

ISO Standard 14143-5

ISO/IEC PDTR 14143-5 identifies 3 general categories of characteristics that affect functional size. These are:

- Dynamic event control (DE)
- Data retention and manipulation (DR)
- Logical operations (LO)

Each of these three general categories contains a number of "characteristics relevant to functional user requirements" (CHARs in ISO terminology). Each category is explained in more detail below.

Dynamic event control

The Functional User Requirements (FUR) dictate that the software operates concurrently with, or controls, external or internal events. The relevant characteristics are:

Response: Software must respond to Internal/External events (e.g., occurrence of a data condition that is recognized by the software, and that requires action to be taken to restore the system to a normal, consistent condition)

System Monitor: Software monitors environment (either via passive or polled data crossing the boundary) to detect out of bounds/emergency data values (i.e., self directed sampling), upon which its processing sequence can be adjusted by stimulus priority (e.g., high priority stimulus can interrupt or alter processing of services)

Interfaces/Control: Interfaces and control of external objects or other software are critical.

Artificial Intelligence: Requirements for the software to be self-learning (i.e., software dynamically adapts its behaviour or logic based on historical data or events).

Business Rules: Business process rules adjust based on date, time, season or other external considerations (e.g., state).

Data Retention and Data Manipulation

The data architecture, relationship requirements, movement and processing of the data prescribed by the Functional User Requirements constrain the software. The relevant characteristics are:

Complex Data: Complex data or control relationships/interdependencies

Persistence: The persistence or logical storage of control information or current state data is an important part of the FUR

Movement: High manipulation of data (e.g., multitude of different functions operating on strings of data such as in word processing and operating system functions)

Services and Logical Operation

The Functional User Requirements specify that particular types of algorithmically intensive services and/or complex operations must be performed by the software. The relevant characteristics include:

Scientific/engineering: Scientific/engineering, mathematical or logical algorithms required (e.g., high precision and accuracy, includes matrix inversion, statistical analysis)

Calculation Rules: Software constrained by rules and relationships that define the calculations

THE CASES

In order to allow pair-wise comparison of software development project, a suitable set of sample projects was needed. Projects had to be fairly simple to allow participants to understand all complete projects within a relatively short time (at most 30 minutes) but had to be non-trivial to allow a realistic size comparison.

Eight small sample cases were drawn from ISO/IEC PDTR 14143-4 and 14143-5. The number eight was chosen to provide two cases for each of the relevant characteristics (CHARs) which were predominantly based on that characteristic and two cases which include a more balanced set of all relevant characteristics. However, using the full set of eight cases in pair-wise comparisons leads to data collection problems given the number of comparisons required. Following the pilot study, the survey was adapted to use two separate sets of four cases.

As an example, one of the cases is given below:

Traffic light change

The user requires a system (for convenience called LIGHTS) that controls the change of traffic lights at a road intersection. The system will monitor the number of cars waiting in front of a traffic light. Depending on the location of the traffic light, the time of day and the number of cars waiting in front of a traffic light the system will have to change the traffic lights using 10 business processing rules:

If location is non-arterial roads then change red light when >= 3 cars wait
A traffic light at a non-arterial road should be green at least for 15 seconds
If location is arterial/non-arterial road and time < 5am then change red light at non-arterial road if 1 car waits
If location is arterial/non-arterial road and time >5am then change red light at non-arterial when >3 cars wait
If location is arterial/non-arterial road and time >5am then change red light at arterial when >15 cars wait
If location is arterial/non-arterial road change green light at non-arterial road when no car waits at non-arterial
If location is arterial roads and time < 6am then change red light when >1 cars wait
If location is arterial roads and time 6-9am or 3-6pm then change light when >15 cars wait
If location is arterial roads and time 9am – 3pm then change light when >7 cars wait
If location is arterial roads and time >6pm then change light when >3 cars wait

PROJECT METHODOLOGY

Users and developers with differing levels of computing experience will be required to complete the survey. Respondents will be asked to identify themselves as users, developers or managers. Since effective completion of the survey will require some time, particularly with needing to understand the functionality requirements of four cases, survey respondents will need to be selected and persuaded to participate.

A pilot study has been undertaken (see below) and has resulted in major restructuring of the survey instrument. The next phase will involve gathering data from several sources:

- Several metrics related conferences. Attendees at these conferences have an interest in software metrics and the availability of a test suite of cases could be of value to them. Three conferences have been identified and will be attended by individuals involved in the data gathering exercise.
- Interested organizations. A number of companies that have shown some interest in software metrics have been approached and others will be targeted. Each company is asked to suggest some participants.
- Software metrics consultants. The group has a number of interested consultants who will also serve to identify interested individuals.

The methodology suffers from the problem of lack of random selection of respondents. However the test cases considered do not favour any one methodology and it is unlikely that any bias will arise. The order of case presentation can be randomised to reduce some of the bias effect. Once the initial data gathering is complete, the data will be analysed to determine the level of variability between responses and the possible need for refinement and further data collection.

AN ANALYTIC HIERARCHY PROCESS FUNCTIONALITY MODEL

The analytic hierarchy process (AHP) was developed by Saaty (1980,1996) as a technique for multiple criteria decision-making. AHP facilitates the determination of relative weightings of several multiple, and sometimes conflicting criteria, towards a specific goal. The process is able to deal with both tangible and intangible factors.

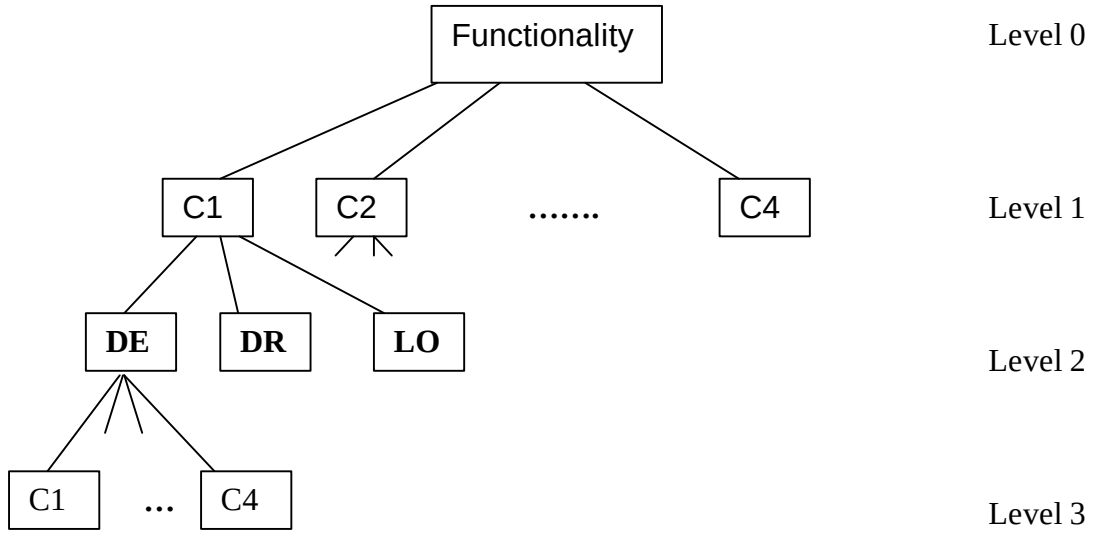

Figure 1: AHP Hierarchy

The first step in classical AHP defines the problem in a form of a hierarchy, where the higher levels reflect the objectives and the lower levels the factors influencing them. The next step requires pair-wise comparisons to be made between each two elements of a given level of the hierarchy, with respect to their contribution towards a factor from a higher level related

to them. Once all comparisons have been performed, AHP uses an eigenvector approach to apportion weights to all nodes in the tree. The final result is a weighted score for each of the node values which reflects the relative importance of each node at that level.

Saaty (1996) defines a consistency measure called the consistency ratio which identifies those comparisons where a revision of judgement is necessary. This is suggested when the consistency ratio exceeds 0.10. This consistency ratio simply reflects the consistency of the pair-wise judgements and shows the degree to which various sets of importance relativities can be reconciled into a single set of weights. For example, if the judgement is made that case A provides more functionality than case B, and B provides more functionality than C, and case C provides more functionality than case A, then the consistency score would be poor, and would be considered a violation of the axiom of transitivity.

The aim of the AHP analysis for this project is for each user to produce, with the given cases, measurements of the amount of functionality in each case. AHP will produce a normalised value giving the weight of each component at the leaf nodes of the AHP hierarchy. The sum of all components equals one. The output is a set of ratio values ie a value of 0.2 indicates twice as much functionality as a value of 0.1. A figure of 0.2 also indicates that this project has 20% of the total functionality for all projects. The model based on four cases is given in Figure 1.

For the lowest level, we require a comparison of the amount of dynamic event control (DE) functionality for case one versus case 2, case 1 vs case 3, etc, For the next level in the hierarchy, for case 1(2,3,..), we need a measure of the amount of dynamic event control relative to data manipulation and logical operations. At the top level, we require a relative assessment of the total amount of functionality in case 1 versus case 2, case 1 versus case 3, etc.

Pairwise comparison in AHP uses a nine point verbal scale, with numeric equivalent values ranging from 1 for equality to 9 for extreme difference between the pair of items. The scale uses the terms equal, moderate, strong, very strong and extreme with points in between each of the terms i.e. the scale is

Equal
Equal to moderate
Moderate
Moderate to strong
Strong
Strong to very strong
Very Strong
Very strong to extremely strong
Extremely strong

Other scales have been proposed e.g. Miranda (1999) suggests that a range from 1 to 6 might be more appropriate for the software domain. The current survey instrument uses the verbal scale with the Saaty numeric values but these will be subject to review.

Although comparison of all eight cases would be preferable to the four case approach, the pairwise comparison approach suffers from a combinatorial explosion when the number of level and items per level increases. At the leaf node level, the number of comparisons required for the eight cases relative to the dynamic event control CHAR is 28 ie. (7+6+5+4+3+2+1) or for n comparisons, the sum of 1 to n-1. With the three functional components (dynamic event control, data manipulation and logical operations) this becomes

84 comparisons. As these comparisons remain the same for the other leaf nodes, there is no need to repeat the comparison at this level.

For the next level, i.e. the amount of each functional component (dynamic event control, data manipulation, logical operations) relative to the other functional components for each case, we need a further 24 comparisons (eight cases with three comparisons). For the final layer, a further 28 comparisons are required, giving a total of 136 comparisons. Maintaining consistency and respondent interest with this number of comparisons becomes a major problem.

For the four case AHP model in Figure 1, the number of comparisons becomes:

Level 1: 6

Level 2: 3 x 4 = 12

Level 3: 3 x 6 = 18

Total = 36

PRELIMINARY ANALYSIS OF PILOT DATA

Nine sample surveys were performed at the Research Laboratory in Software Engineering Management at the University du Quebec in Montreal. These surveys used a form of the AHP model with all eight cases which provided for pair-wise comparison of case functionality for overall functionality as well as for dynamic event control, data manipulation and logical operations functionality. There was no comparison of the amount of dynamic event control, data manipulation or logical operations functionality in each case. Of these nine cases four had insufficient entries to be used and the remaining five were complete. The five cases were analysed using AHP.

The pilot data suggested that it was difficult for respondents to maintain consistency and that some components of functionality (as defined above) may be easier to visualise than others in terms of maintaining consistency. For the dynamic event control functional category, four of the five respondents were within a consistency rating of 0.1 (acceptable by the Saaty convention) while the data manipulation dimension had only two within this range. No respondents managed to maintain consistency in all four aspects of functionality measured (the three components and overall functionality) although three users had three of the four less than 0.1. The issue of consistency is addressed further below but was a major factor in reducing the hierarchy from eight cases to four.

FP, FFP and MK2 counts were performed for each of the eight cases and correlated with the estimated functional size from the five samples (using the mean size for each functional component and the overall functionality). The small sample size makes any results very suspect but the data does indicate a relationship between the data manipulation component and user perceived functionality for all three metrics. Although the other components did not appear significant, some relationships may be established with more data.

Regression was also used to determine how strongly the three functional components related to the predicted overall functionality. All three components were significant, providing some support for the ISO model.

AUTOMATING DATA COLLECTION

The problem of maintaining consistency in judgements makes data collection a problem if rapid feedback is not available. If users complete a manual questionnaire for later analysis,

there is usually no possibility of revising inconsistent responses detected during analysis unless the respondents are still available. Even if they are, the time delay could make revision of the results difficult and time-consuming. In an ideal situation, inconsistency would be detected as soon as possible during data collection and the users given the opportunity to revise responses as needed.

A simple data collection program has been written which leads users through completion of the questionnaire and provides immediate feedback as soon as any inconsistency is detected, Users can navigate forward and backward between responses and change their responses as they wish. Although the current version requires access to a PC with the program resident for data collection, we are proposing to encapsulate this as an ActiveX component and make it available on a web site for data collection..

An additional problem in the AHP analysis identified during the pilot studies was incomplete data. AHP assumes a complete set of comparisons. Using electronic data collection allows prompting and verification of the data. A further advantage of using the program is that data is already in an electronic form, which obviates having to manually enter it for analysis.

CONCLUSION

Users should feel comfortable with whatever size metrics they are using. Unless the estimated size (using a specific metric) of one project or component relative to another is roughly in agreement with the perception of the user, it is possible that the cognitive dissonance induced could result in a loss of confidence in the metric. The approach being taken in this research will determine how users perceive software project size and will relate the perceived size to several software metrics.

The COSMIC project is a major initiative to broaden the basis of software size estimation. COSMIC aims to "develop, test, bring to market and gain acceptance as an industry standard, a new generation of software sizing methods which are applicable for performance measurement

- As a component of estimating methods from early in a software item's life
- In as wide a range of software 'domains' as possible;" [Symons, 1999]

A worldwide set of field trials is being undertaken to advance the method to a 'proven' status. The goals are to test whether the method gives acceptable size measures, is unambiguously defined and gives repeatable results for a reasonable sample of projects. The research project described in this paper is a small part of the overall validation effort of COSMIC but will also be available as a benchmark for new software size metrics.

REFERENCES

Abran, Alain,(1999) FFP Release 2,0: An Implementation of COSMIC Functional Size Measurement Concepts, Presented at FESMA 99, Amsterdam, October 4-7, 1999

Albrecht, A.J and J.E. Gaffney, (1983) "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", *IEEE Transactions on Software Engineering*, vol. 9, no. 6, pp. 639–648.

Finnie G.R., Wittig G.E. and Desharnais J-M., Using AHP to Incorporate Complexity and Productivity Differences into Software Development Effort Estimation" 6th

Australasian Conference on Information Systems in Hobart, Tasmania, 11-13 December 1996, pp231-240.

Miranda, Eduardo (1999), An Evaluation of the Paired Comparisons Method for Software Sizing, Proceedings of the 9th International Software Measurement Workshop

Morris, P.; Desharnais, J.-M. (1998), Measuring ALL the Software not just what the Business Uses, in IFPUG Fall Conference, Orlando, Florida, September, 1998 21-25.

Oligny,S., Abran,A., St-Pierre,D.,(1999) Improving Software Functional Size Measurement, Proceedings of COCOMO and Software Cost Modeling International Forum 14, Los Angeles, USA, October 26-29, 1999.

Saaty, T.L.(1980) *The Analytic Hierarchy Process*, McGraw-Hill, New York.

Saaty, T.L (1990) "How to make a decision: The analytic hierarchy process", *European Journal of Operations Research*, no. 48, pp. 9 - 26, 1990.

Saaty,T.L. (1996) Multicriteria Decision Making: the Analytic Hierarchy Process, RWS Publications

Symons, Charles (1988): Function Point Analysis: Difficulties and Improvements, IEEE Transactions on Software Engineering,14 (1): 2-11 (1988)

Symons, Charles(1991), *Software Sizing and Estimating MkII FPA*, John Wiley & Sons, Chichester

Symons, Charles(1999) COSMIC FFP - Aims, Design Principles and Progress, Presented at FESMA 99, Amsterdam, October 4-7.

Rudolph E.E., Wittig G.E, Finnie G.R and Morris P.M. (1998) Verifying Function Point Values, European Software Measurement Conference (FESMA'98), Antwerp, Belgium, May, 1998

Wittig G.E, Rudolph E.E., Finnie G.R and Morris P.M. (1999) "A Web-Based Research Project to Gather International Functional Size Data for Function Point Coefficient Determination" in Rob J Kusters, Adrian Cowderoy, Fred J. Heemstra and Erik P.W.M. van Veenendal (eds), "Project Control for Software Quality: Proceedings of ESCOM-SCOPE 99", Shaker Publishing Co. Masstricht, pp. 237-246.

COPYRIGHT