Technical Briefing

Software Measurement

Introduction

When we go for a health check we expect the doctor to measure certain things like our pulse, blood pressure, temperature, weight and height and then interpret these measurements along with some more subjective measures to make an overall diagnosis. Software measurement can act as a health check for your software project. As with a medical, it can be used to monitor the current situation and act as an early warning of problems ahead and also to provide data to estimate aspects of future projects.

Selecting what to measure is one of the most important issues of any measurement programme. Collecting the wrong data is not only costly in wasted effort, it could also cost you the staff support needed to make a measurement programme work. The data you decide to collect should be driven by your company's business objectives.

Many texts will recommend that the number of measures initially selected be no more than five. These are sometimes referred to as global measures i.e. they are measured across the development life cycle and are not specific to one particular phase.

Three of the most basic measurements are:

- size
- time - schedule/effort
- defects

Size

Size is a measure of project deliverables, usually code and documentation. It can be used to measure progress and to estimate future work. There are two main techniques for measuring code size;

- lines of code (LOC)
- number of Function Points

The first of these, LOC, would initially seem easy to measure. However, unfortunately there is no standard definition for a line of code. In general, it is best to keep your definition clear, simple and, most importantly, consistent. An example is given below (Software Engineering Metrics and Models, Conte, 1986):

A line of code is any line of program text that is not a comment or blank line, regardless of the number of statements or fragments of statements on the line. This specifically includes all lines containing program headers, declarations, and executable and non-executable statements.

However, it is still the most useful size measure when tracking progress.

Another method for defining size is to use Function Point Analysis, FPA. There are two basic methods of Function Point Analysis:

- Albrecht, used mainly in the USA
- Mark II, used mainly in Europe (in particular the UK)

There is no consensus of opinion as to which technique is better. Work, at an international level, to produce a set of standards in this area is near completion.

FPA claims to be language independent by measuring the functionality of the system and is also independent of programming style, therefore making it valuable when measuring productivity.

The function point count is available early in the life cycle and so can be used in the estimation process.

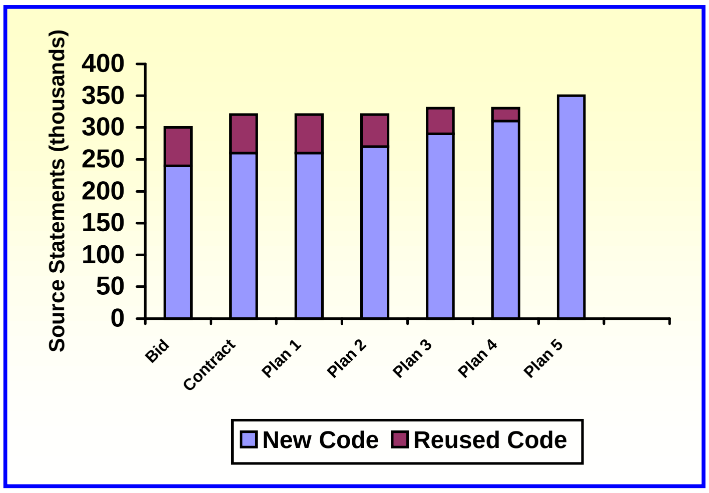


Figure 1 Exposing Potential Cost Growth - the disappearance of reused code (based on a figure from CMU/SEI-92-TR-19)

Documentation can be measured in, for example, words, lines or pages but again, as for code, there must be a clear definition of the unit of measure.

Time

Time is often being recorded already as part of project management activities. There are two measures that are of interest:

Effort

Effort is best measured in hours as there are different definitions of days (how many hours in a standard day) and weeks (what is a standard working week). It is important to record all time including unpaid overtime.

It is also useful to record the type of employee e.g.

- programmer,
- consultant,
- user,
- manager

and with which activity they were involved e.g. requirements analysis, testing etc. This will provide a greater level of granularity in analysis reports.

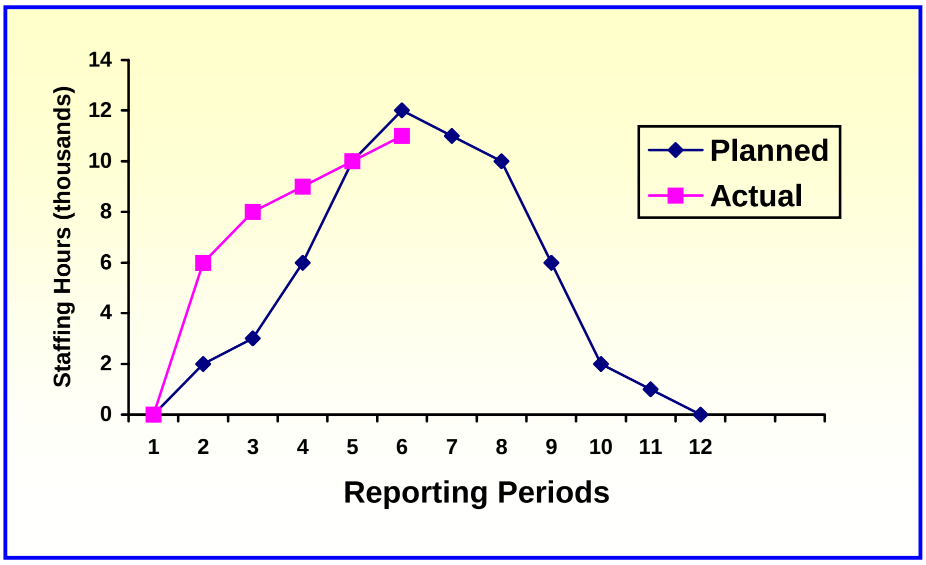


Figure 2 Effort Profile for Total System Expenditure by Month - either staff allocated too early or more analysis effort required than estimated (based on a figure in CMU/SEI-92-TR-21)

Schedule

Schedule is measured in calendar time and records the completion of events like project milestones, review, audits and deliverables. This information is of interest to both project managers and estimators. It is important that everyone knows what is meant by the completion of a particular event so schedules can be compared and interpreted correctly by different people.

Formalising the definitions and making them consistent across projects makes it possible to compare projects. Problems and differences can be identified and investigated further.

Defects

Measures associated with defects can give an indication as to the perceived quality of a system. As with time, information on defects is probably already being recorded but may not be being analyzed to provide vital feedback on process and product quality.

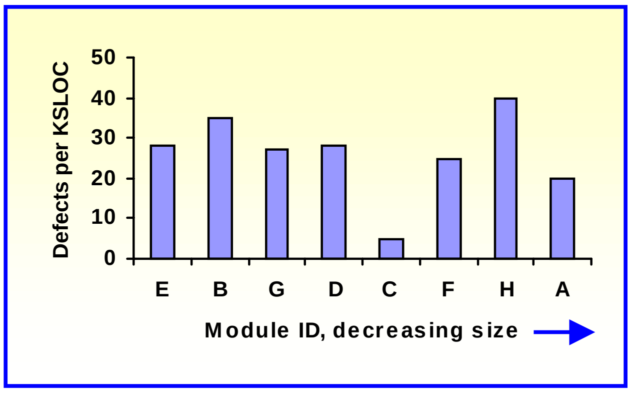
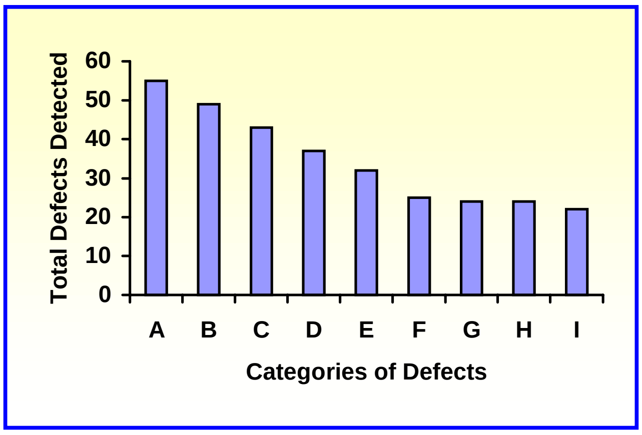


Figure 3 Identifying Modules with High Defect Densities - unusual modules can be selected for examination, restructuring or redesign (based on a figure in CMU/SEI-92-TR-22)

By counting and tracking problems and defects consistently it is possible to display trends and highlight the effects of changes in the process.

Figure 4 Histogram of Categories of Defects Detected - if percentages are used, inter-project comparisons are possible (based on a figure in CMU/SEI-92-TR-25)



Further Information:

The following is a selection of the material available in the CSE Library:

Software Metrics: A Rigorous Approach & Practical Approach (Second Edition)
by Fenton & Pfleeger *ISBN 0-534-95600-9, 1997*

A quantitative approach to Software Management - The ami Handbook
by Pulford, Kuntzmann-Combelles and Shirlaw *ISBN 0-201-87746-5, 1996*

Defining & Using Software Measures
by Software Engineering Institute Carnegie Mellon University, 1992

Applied Software Measurement
by Capers Jones *ISBN 0-07032813-7, 1991*

Software Metrics: Establishing a Company-Wide Program
by Grady & Caswell *ISBN 0-13-821844-7, 1987*

Centre for Software Engineering Ltd, Dublin City University Campus, Dublin 9, Ireland.
Telephone: +353 1 7045750 Fax: +353 1 7045605 Email: admin@cse.dcu.ie