TOTAL METRICS

Uses and Benefits of Function Points

April 2001

USES AND BENEFITS OF FUNCTION POINTS

USES AND BENEFITS OF FUNCTION POINTS

1 INTRODUCTION

Industry experience has shown that an emphasis on project management and control offsets much of the risk associated with software projects. One of the major components of better management and control of both in-house development and a package implementation is **measurement**.

This includes measurement of:

- the scope of the project e.g.
  - ⇒ *software units to be delivered*
- performance indicators of efficiency and cost effectiveness e.g.
  - ⇒ *cost per unit of software delivered,*
  - ⇒ *staff resources per unit of software delivered ,*
  - ⇒ *elapsed time to deliver a unit of software.*
- quality indicators e.g.
  - ⇒ *number of defects found per unit of software delivered*

The outcome of a Function Point count provides the metric 'unit of software delivered' and can be used to assist in the management and control of software development, customisation or major enhancements from early project planning phases through to the ongoing support of the application.

Knowing the software size facilitates the creation of more accurate estimates of project resources and delivery dates and facilitates project tracking to monitor any unforeseen increases in scope. The measurement of the performance indicators enables benchmarking against other development teams and facilitates better estimating of future projects. These are only some of the ways in which Function Point Analysis (FPA) can assist IT management. These and other lesser known ways in which FPA can assist IT to move towards 'best practice' in the management of their software products and processes, are discussed in the following sections.

The benefits of using measurement to support management decision-making, can only be achieved if the information supporting these decisions is relevant, accurate and timely. In order to ensure the quality of their measurement data, organisations need to implement a 'measurement process'. The cost of implementing the activities, procedures and standards to support the function point counting process will depend on the size and structure of the organisation and their measurement needs. These considerations are discussed in the last section "Costs of Implementing Function Point Analysis".

2 MANAGING PROJECT DEVELOPMENT

2.1 FPA Uses and Benefits in ***Project Planning***

2.1.1 Project Scoping

A recommended approach for developing function point counts is to first functionally decompose the software into is elementary functional components (base functional components). This decomposition may be illustrated graphically on a functional hierarchy. The hierarchy provides a pictorial 'table of contents' or 'map' of the functionality of the application to be delivered. This approach has the advantage of being able to easily convey the scope of the application to the user, not only by illustrating the number of functions delivered by each functional area, but also a comparative *size* of each functional area measured in function points.

2.1.2 Assessing Replacement Impact

If the software to be developed is planned to replace existing production applications it is useful to asses if the business is going to be delivered more, less or the same functionality. The replacement system's functionality can be mapped against the functionality in the existing system. A quantitative assessment of the difference can be measured in function points. Note, this comparison can only be done if the existing applications have already been sized in Function Points.

2.1.3 Assessing Replacement Cost

Multiplying the size of the application to be replaced by an estimate of the [1]dollar cost per function point to develop, enables project sponsors to develop quick estimates of replacement costs. Industry derived costs are available and provide a ballpark figure for the likely cost. Industry figures are a particularly useful reference if the re-development is for a new software or hardware platform not previously experienced by the organisation. Ideally organisations should establish their own 'cost per function point' metrics for their own particular environment based on project history.

If you are considering implementing a 'customised off the shelf' package solution then this provides a quick comparison of the estimated package implementation costs to compare with an in-house build. Package costs typically need to include the cost of re-engineering the business to adapt the current business processes to those delivered by the package. These costs are usually not a consideration for in-house developed software.

[1] International Software Benchmarking Standards Group Release 6 Report April 2000 provides cost value for software projects in 1999 – median costs to develop a function point = $US716 , average costs = $US849 per function point
Cost data is derived from 56 projects representing a broad cross section of the software industry. Industry sectors represented are banking, insurance, communications, government and financial services organizations. They include a mixture of languages, platforms, application types, development techniques, project types, size (50 – 3,000 function points) and effort (from under 1000 to 40,000 hours). Most were implemented between 1995 and 1997. All costs include overheads and the effort data was recorded for the development team and support services.

2.1.4 Negotiating Scope

Initial project estimates often exceed the sponsors planned delivery date and budgeted cost. A reduction in the scope of the functionality to be delivered is often needed so that it is delivered within a predetermined time or budget constraints. The functional hierarchy provides the 'sketch-pad' to do scope negotiation. I.e. it enables the project manager and the user to work together to identify and flag (label) those functions which are:

- *mandatory* for the first release of the application,
- *essential* but not mandatory,
- *optional* and could be held over to a subsequent release.

The scope of the different scenarios can then be quickly determined by measuring the functional size of the different scenarios. E.g.: the project size can be objectively measured to determine what the size (and cost and duration) would be if:

- all functions are implemented
- only *mandatory* functions are implemented
- only *mandatory* and *essential* functions are implemented.

This allows the user to make more informed decisions on which functions will be included in each release of the application based on their relative priority compared to what is possible given the time, cost and resource constraints of the project.

2.1.5 Evaluating Requirements

Functionally sizing the requirements for the application quantifies the different types of functionality delivered by an application. The function point count assigns function points to each of the function types, External Inputs, Outputs and Enquiries and Internal and External Files.

Industry figures available from *ISBSG Repository*[2] for projects measured with IFPUG function points indicates that 'complete' applications tend to have consistent and predictable ratios of each of the function types. The profile of functionality delivered by each of the function types in a planned application can be compared to that of the typical profile from implemented applications, to highlight areas where the specifications may be incomplete or there may be anomalies.

The following pie chart illustrates the function point count profile for a planned *Accounts Receivable* application compared to that from the ISBGS data. The reporting functions (*outputs*) are lower than predicted by industry comparisons. Incomplete specification of reporting functions is a common phenomena early in a project's lifecycle and highlights the potential for substantial growth creep later in the project as the user identifies all their reporting needs.

The quantitative comparison below shows that the reporting requirements were lower than expected by about half (14% compared to the expected 23% of the total function points). The project manager in this case verified with the user that the first release of the software would require all reporting requirements and the user indicated that more reports were likely to be specified. The project manager increased the original count to allow for the

[2] International Software Benchmarking Standards Group (ISBSG) is an international group of representatives from international metrics organizations who collect project data from countries including Australia, Austria, Canada, Denmark, Germany, Hong Kong, India, Japan, New Zealand, Norway, Poland, United Kingdom and the United States.

extra 9% and based his early project estimates on the higher figure that was more likely to reflect the size of the delivered product. The function point measurement activity enabled the project manager to quantify the potential missing functionality and justify his higher, more realistic estimate.

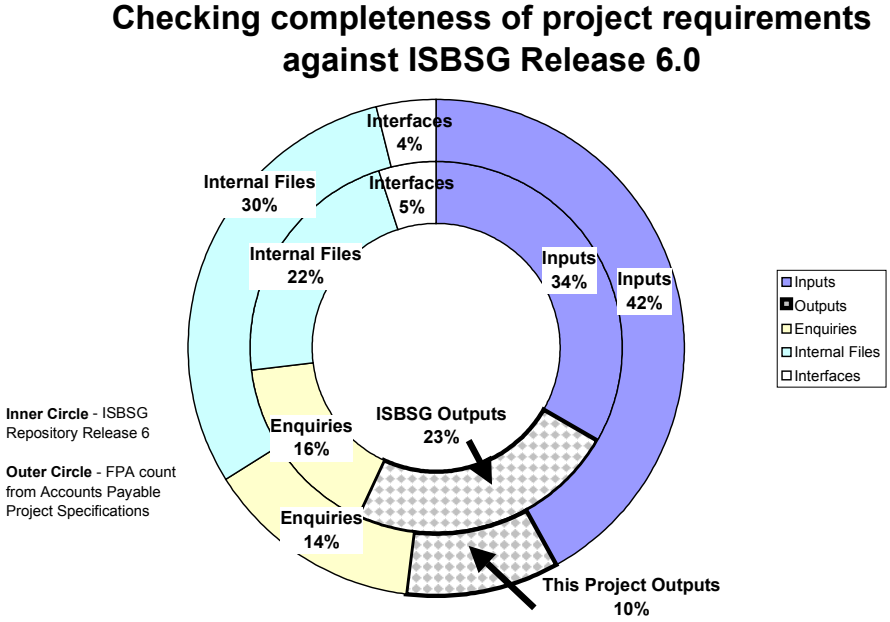

Figure 1

2.1.6 Estimating Project Resource Requirements

Once the scope of the project is agreed the estimates for effort, staff resources, costs and schedules need to be developed. If productivity rates (*hours per function point, $cost per function point*) from previous projects are known, then the project manager can use the function point count to develop the appropriate estimates. If your organisation has only just begun collecting these metrics and does not have sufficient data to establish its own productivity rates then the ISBGS industry data can be used in the interim.

2.1.7 Allocating Testing Resources

The functional hierarchy developed as part of the function point count during project development can assist the testing manager to identify high complexity functional areas which may need extra attention during the testing phase. Dividing the total function points for each functional area by the total number of functions allocated to that group of functions, enables the assessment of the relative complexity of each of the functional areas.

The effort to perform acceptance testing and the number of test cases required is related to the number and complexity of the user functions within a functional area. Quantifying the relative size of each functional area will enable the project manager to allocate appropriate testing staff and check relative number of test cases assigned.

2.1.8 Risk Assessment

Many organisations have large legacy software applications, that due to their age, are unable to be quickly enhanced to respond to the needs of their rapidly changing business environments. Over time these applications have been patched and expanded until they have grown to monstrous proportions. Frustrated by long delays in implementing changes,

lack of support for their technical platform and expensive support costs, management will often decide to redevelop the entire application. For many organisations this strategy of rebuilding their super-large applications has proved to be a disaster resulting in cancellation of the project mid-development. Industry figures show that the risk of project failure rapidly increases with project size. Projects less than [3]500 function points have a risk of failure of less than 20% in comparison with projects over 5000 function points which have a probability of cancellation close to 40%[4]. This level of risk[5] is unacceptable for most organisations.

Assessing planned projects for their delivered size in function points enables management to make informed decisions about the risk involved in developing large highly integrated applications or adopting a lower risk phased approach described below.

2.1.9 Phasing Development

If the project manager decides on a phased approach to the project development then related modules may be relegated to different releases. This strategy may require temporary interfacing functionality to be built in the first release to be later decommissioned when the next module is integrated. The function point count allows project managers to develop 'what if scenarios' and quantify the project scope of each phase as a means of making objective decisions. Questions to which quantitative answers can be provided are:

- how much of the interfacing functionality can be avoided by implementing all of the related modules in release one?
- what is the best combination of potential modules to group within a release to minimise the development of temporary interfacing functions?

If it is decided to implement the application as a phased development then the size of each release can be optimised to that which is known to be manageable[6]. This can be easily done by labelling functions with the appropriate Release and performing 'what-if' scenarios by including and excluding functions from the scope of the count for the release.

2.2 FPA Uses and Benefits in *Project Construction*

2.2.1 Monitoring Functional Creep

Function point analysis provides project management with an objective tool by which project *size* can be monitored for change, over the project's lifecycle.

[3] Data within the ISBSG Repository Release 6 supports the premise that smaller projects are successful. Over 65% of the projects in the repository are less than 500 function points and 93% of the projects are less than 2000 function points. The repository is populated by industry projects, voluntarily submitted by organizations that want to benchmark their project's performance against industry projects with a similar profile. Consequently organizations tend to submit successfully completed projects which have better than average performance i.e. the ones which did not 'fail'.

[4] Software Productivity Research

[5] At a median industry cost of $716/fp delivered, a 5000 function point project is risking $3.5 million dollars.

[6] Industry experience suggests that the best managed projects which deliver quality software on time and within budget tend to less than 700 function points and up to 1500 function points.

As new functions are identified, functions are removed or changed during the project the function point count is updated and the impacted functions appropriately flagged. The [7]project scope can be easily tracked and reported at each of the major milestones.

If the project size exceeds the limits allowed in the initial estimates then this will provide an early warning that new estimates may be necessary or alternatively highlight a need to review the functionality to be delivered by this release.

2.2.2 Assessing and Prioritizing Rework

Function Point Analysis allows the project manager to objectively and quantitatively measure the scope of impact of a change request and estimate the resulting impact on project schedule and costs. This immediate feedback to the user on the impact of the rework allows them to evaluate and prioritise change requests.

The cost of rework is often hidden in the overall project costs and users and developers have no means to quantify its impact on the overall project productivity rates. Function point analysis enables the project manager to measure the functions that have been reworked due to user-initiated change requests. The results provide valuable feedback to the business on the potential cost savings of committing user resources early in the project to establish an agreed set of requirements and minimising change during the project lifecycle.

2.3 FPA Uses and Benefits after ***Software Implementation***

2.3.1 Planning Support Resources and Budgets

The number of personnel required to [8]maintain and support an application is strongly related to the application's size. Knowing the functional size of the application's portfolio allows management to confidently budget for the deployment of support resources. The following figure demonstrates this relationship as demonstrated within an Australian financial organisation. The average maintenance assignment scope (number of function points supported per person) for this organisation is 833 function points per person. The assignment scope has been found to be negatively influenced by the age of the application and the number of users i.e. as both these parameters increase the assignment scope decreases. [9]Capers Jones figures show similar assignment scopes where for ageing, unstructured applications with high complexity an assignment scope of 500 function points per person is not unusual whereas newer, structured applications, skilled staff can support around 1500 – 2000 function points.

[7] The Victorian State Government in Australia has adopted a recommended policy for Government departments to manage and control government out-sourced development projects using Function Points. Suppliers tender for the development based on a fixed price in dollars per function point. Scope changes are automatically charged by the supplier at a pre-determined contracted charge-rate based on the number of function points impacted and the stage at the life cycle when the change was introduced. The government policy underpinning this approach is called 'Southern Scope'. More information is available at: www.mmv.vic.gov.au/southernscope

[8] Where maintenance and support includes defect repairs and very minor enhancements.

[9] Capers Jones – Applied Software Measurement – Assuring Productivity and Quality – McGraw Hill – Software Engineering series 1991 – Chapter 3 Page180.

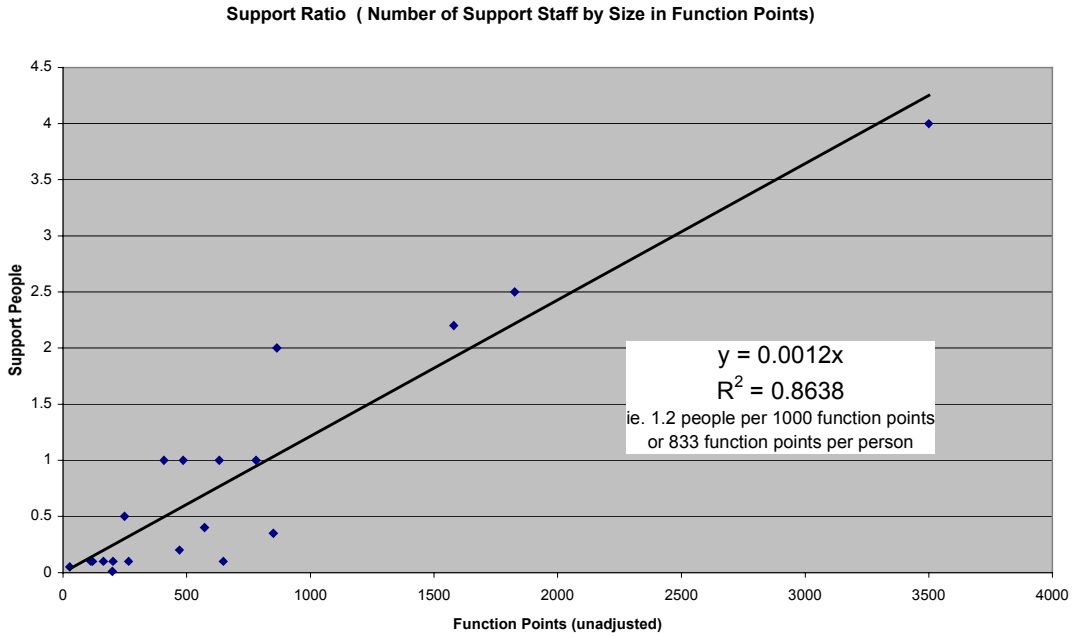

Figure 2 Relationship between the Size of an Application and the Number of Support staff (Source - Total Metrics 1999)

Once implemented, applications typically need constant enhancement in order to respond to changes in direction of an organisation's business activities. Function points can be used to estimate the impact of these enhancements. The baseline function point count of the existing application will facilitate these estimates. As the application size grows with time the increasing assignment scope will provide the justification to assign more support staff.

2.3.2 Benchmarking

The function point count of delivered functionality provides input into productivity and quality performance indicators. These can then be compared to those of other in-house development teams and implementation environments. Benchmarking internally and externally with industry data enables identification of best practice. External benchmarking data is readily available in the ISBSG[10] Repository.

2.3.3 Identifying Best Practice

Project managers seeking 'best practice' in their software development and support areas recognise the need to adopt new tools, techniques and technologies to improve the productivity of the process and quality of the products they produce. Baselining current practice enables management to establish current status and set realistic targets for improvement. Ongoing measurement of productivity and quality key performance indicators enable management to assess the impact of their implemented changes and identify where further improvements can be made. Function points are the most universally accepted method to measure the output from the software process. They are a key metric

[10] For information on how to access the ISBSG data visit : www.ISBSG.org.au

within any process improvement program because of their ability to normalise data from various software development environments combined with their ability to measure output from a business perspective as compared to a technical perspective.

2.3.4 Planning New Releases

The functional hierarchy of the functionality delivered by an application can also assist the support manager in planning and grouping change requests for each new release of the application. The hierarchy illustrates closely related functions and their relative size. If the impact of change is focussed on a group of related functions then development effort will be reduced particularly in the design, testing and documentation stages of the project. This strategy of evaluating the scope of impact of a change request also reduces project risk by restricting projects to a manageable size and focussing change on a restricted set of related business functions.

2.3.5 Software Asset Valuation

Function Point Analysis is being used increasingly by organisations to support the 'valuation of their software assets'. In the past, software has been considered an expense rather than a capital asset and as such was not included in an organisations asset register. The most commonly used software valuation method is based on the 'deprival method'. This method values the software based on what it would cost to replace in today's technical environment rather than what it cost originally to build. The industry build rate (dollar cost per function point) is determined and the total replacement value is calculated based on the current functional size of the application.

Since FPA provides a means of reliably measuring software then some organisations have implemented accrual budgeting and accounting in their business units. Under this directive, all assets must be valued based on deprival value and brought to account, thus ensuring better accountability of the organisations financial spending. Funding via budget allocation is based on assets listed in their financial accounts and their depreciation. In the past, the purchase price of the software recorded as an expense within an accounting year. These more recent accounting practices mean that it can now be valued as an asset and depreciated.

Publicly listed organisations have found that by using this accrual accounting method of measuring software as an asset rather than an expense they can amortise the depreciation over five years rather than artificially decrease the current year's profit by the total cost of the software. This strategy has a dramatic effect on their share price since once their software is listed as a capital asset it contributes to the overall worth of the company and the total cost of that asset has a reduced impact on the current year's reported profit.

2.3.6 Outsourcing Software Production and Support

The benefits of Functional size measurement in outsourcing contracts, is that functional size enables suppliers to measure the cost of a unit of output from the IT process to the business and enables them to negotiate on agreed outcomes with their client.

Specifically these output based metrics based on function point analysis has enabled **suppliers** to:

- quantitatively and objectively differentiate themselves from their competitors
- quantify extent of annual improvement and achievement of contractual targets

- negotiate price variations with clients based on an agreed metric
- measure financial performance of the contract based on unit cost of output
- at contract renewal be in a stronger bargaining position supported by an eststablished set of metrics

Conversely these output based metrics based on function point analysis has enabled **clients** to:

- Objectively assess supplier performance based on performance outputs delivered rather than concentrating on inputs consumed.
- Establish quantitative performance targets and implement supplier penalties and bonuses based on achievement of these targets
- measure the difference between internal IT costs compared to the cost of outsourcing based on similar output
- quantitatively compare competing suppliers at contract tender evaluation stage.

Most of the international outsourcing companies use function point based metrics as part of their client service level agreements. Whilst this method of contract management is relatively new its proponents are strong supporters of the usefulness of the technique. In our experience once an outsourcing contract has been based on Function Point metrics subsequent contract renewals expand on their use.

Metrics initiatives have a high cost and need substantial investment, which is often overlooked at contract price negotiation. Both the supplier and the client typically incur costs. However, given the size of the penalties and bonuses associated with these contracts it soon becomes obvious that this investment is necessary.

3 Customising Packaged Software

3.1 Background

For selected MIS applications, implementing a packaged 'off the shelf' solution is the most cost effective and time efficient strategy to deliver necessary functionality to the business.

All of the benefits and uses of Function Point Analysis which applied to in-house development projects as described in the previous section can also apply to projects which tailor a vendor supplied package to an organisations specific business needs.

Experience shows that Function Point Counting of packages is not always as straightforward as sizing software developed in-house, for the following reasons:

- only the physical and technical functions are visible to the counter. The logical user view is often masked by the physical implementation of the original logical user requirements.
- in most cases the functional requirements, functional specifications, and logical design documentation are not delivered with the software. The counter may have to rely on the User Manual or online help to assist in interpreting the user view.

  ⇒ *The modelling of the logical business transactions often requires the function point counter to work with the client to identify the logical transactions. They do this by investigating the users functional*

requirements and interpreting the logical transactions from the package's physical implementation.

- in most cases the names of the logical files accessed by the application's transactions are not supplied by the package vendor.

  ⇒ *The function point counter will need to develop the data model by analysing the data items processed by the application.*

However, with sufficient care a reasonably accurate function point count of packaged applications can usually be obtained.

3.2 Estimating Package Implementations

The project estimates for a package solution need to be refined for each implementation depending on the percentage of the project functionality which is:

- native to the package and implemented without change
- functionality within the package which needs to be customised for this installation
- functionality contained with the organisations existing applications which needs to be converted to adapt to the constraints of the package
- to be built as new functions in addition to the package functions
- to be built to as new functions to enable interfacing to other in-house applications
- not to be delivered in this release.

The productivity rates for each of these different development activities (to implement, customise, enhance or build) are usually different. This complexity of assigning an appropriate productivity factor can be compounded when the package provides utilities which enable quick delivery based on changes to rule tables. Change requests, which can be implemented by changing values in rule-based tables, can be implemented very efficiently compared to a similar user change request, that requires source code modification. It is recommended that these different types of activities are identified and effort collected against them accordingly so that productivity rates for the different activity types can by determined.

The functions can be flagged for their development activity type and their relative contributions to the functional size calculated. This will enable fine-tuning of the project estimates.

Another area of concern when developing estimates for package integration is the need to determine the extent that the application module needs to interface with existing functionality. The function point count measures the external files accessed by transactions within this application. A high percentage of interface files (>10%) suggests a high degree of coupling between this application and existing applications. A high degree of interfacing tends to have a significant negative impact on productivity rates and needs to be considered when developing estimates.

4 SUMMARY

Function Point Analysis is a technique that until now has been restricted within many organisations to only be used for better estimating or input into benchmarking productivity

rates. The above examples illustrate a wider range of uses where it can contribute to the better management and control of the whole software production environment.