Case Study of a Successful Measurement Program

Version 1.2 - September 2007

Pam Morris

(BSc.Grad Dip Comp.Dip Ed, CFPS, CSMS (Level 3))
CEO Total Metrics Australia.
WWW.totalmetrics.com
Pam.Morris@Totalmetrics.com

Abstract

The paper reviews the implementation of measurement using the ISO/IEC15939 [5] framework and examines how the measures have provided new (and some unexpected) insights into organisations' development and enhancement processes.
The original objective of the measurement and benchmarking activity was to assist the IT Department demonstrate the cost efficiencies and benefits achieved by the gradual re-factoring of their large legacy application (~14,000 fps) over a period of 4 years. However, the process of baselining, collecting and analysing the measures has provided additional valuable insights into the Departments current development processes that have enabled significant productivity and quality improvements to be realised. These insights have changed the way that the Project managers plan and estimate their ongoing change requests, package quarterly Releases, and where they focus their development effort. Measurement results provide a major input into setting directions for their process improvement initiatives. The paper examines the key success factors of this measurement program that has been effective and evolving for over 2 years and how senior management responded to the measurement data and incorporated it in their decision making.

1. Background

Howard Rubens [1] reported in 1995, that 4 out of 5 software measurement programs 'fail to succeed', where a successful program is one that lasts for more than 2 years and it impacts the organisation's management decisions. Total Metrics is a metrics consultancy that has assisted numerous clients to implement measurement programs over the past 13 years. Frustratingly we have experienced similar levels of failure to that reported by Rubens in 1995. This paper looks at a case study of one or our 'successful' programs and investigates why it has succeeded when so many fail.

The paper is based on the experiences of a section within an Australian Government Department employing approximately 60 developers to enhance, maintain and support their large (~14,000 fps) legacy Asset Licensing and Registration (ALRS) system. The majority of the personnel within the IT area are long term, highly skilled contractors who have had significant experience in the competitive private sector. The ARLS system provides a significant revenue stream to the Department and is required by the business to remain profitable and competitive.

In 2004 management decided to replace their 1992 Cool:GEN application but recognising the high risk of project failure associated with such a large project, they decided to [1]re-factor the application 'component by component', rather than undergo a complete re-development project. In order to assess the benefits of this approach they decided to baseline their current environment and monitor it over a 4 year period.
Late 2004, the Department engaged Total Metrics to establish a measurement process that would support Benchmarking the productivity and quality of their ALRS development process. To optimise the success of the measurement implementation Total Metrics followed the process recommended by the ISO standard *ISO/IEC15939: [5] Systems and software engineering — Measurement Process* described in the diagram below.

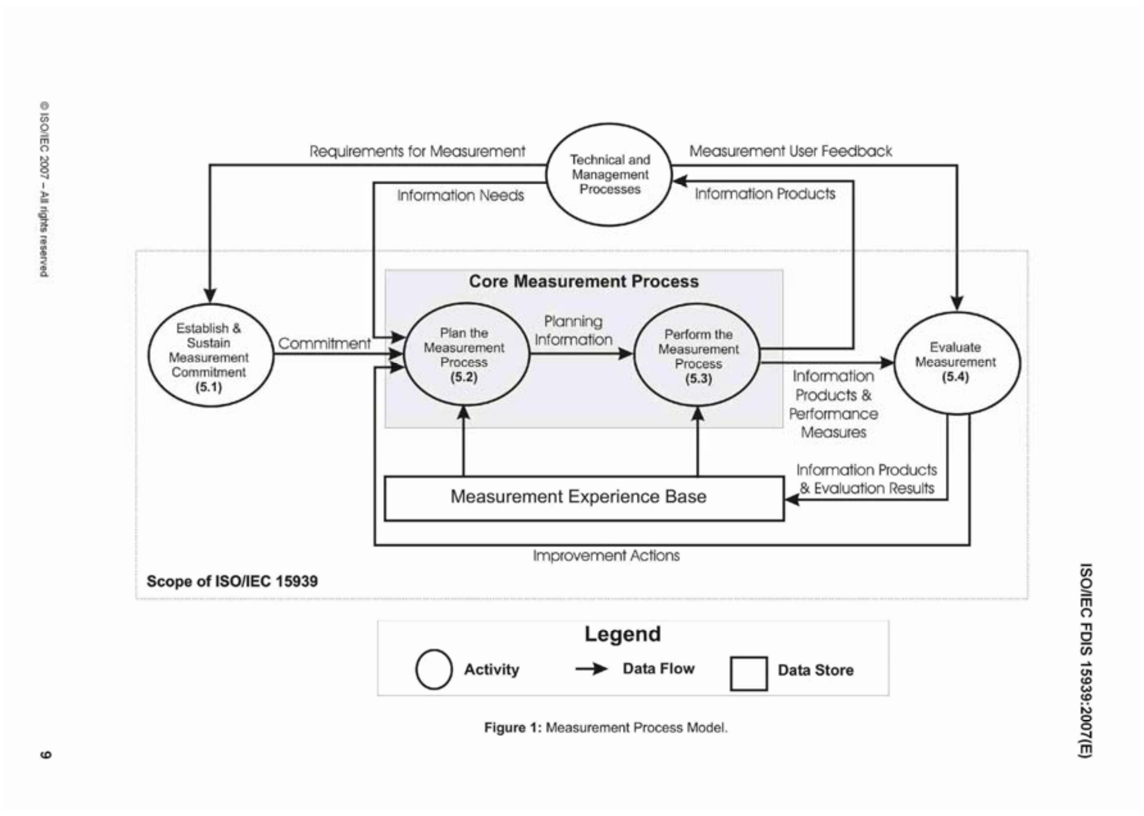
© ISO/IEC 2007 – All rights reserved
Requirements for Measurement
Technical and Management Processes
Measurement User Feedback
Information Needs
Information Products
Core Measurement Process
Establish & Sustain Measurement Commitment (5.1)
Commitment
Plan the Measurement Process (5.2)
Planning Information
Perform the Measurement Process (5.3)
Evaluate Measurement (5.4)
Information Products & Performance Measures
Information Products & Evaluation Results
Measurement Experience Base
Improvement Actions
Scope of ISO/IEC 15939
Legend
Activity
Data Flow
Data Store
Figure 1: Measurement Process Model.
ISO/IEC FDIS 15939:2007(E)
6

Figure 1 ISO/IEC 15939: Measurement Process

2. Implementing the Measurement Program

2.1. Establish and Sustain Measurement and Management Commitment

The primary objective of the Benchmarking exercise was to quantify the expected productivity and quality gains from the re-factoring activities in order to evaluate cost of implementation versus the proposed benefits of faster development, higher quality delivered product, lower maintenance and support costs. It was anticipated that the higher productivity

[1] Re-factoring involves improving the design of each functional component with the aim of ensuring that software continues to adapt, improve and remain easy to read and modify without altering its observable behaviour. The objective is to have software that is efficient, fresh and adaptable.

of the development process and better product quality would have a direct positive effect on the revenue earned by the ARLS system for the Department.

The client recognised that these proposed benefits may take several years to become apparent and therefore committed financial and personnel resources to their measurement program for the 4 years of their re-factoring project. They also acknowledged that in order to optimise their process improvement activity that they needed respond to the results of the measurement and to implement any recommendations proposed in the Benchmarking reports. This ongoing long term management vision and commitment has been a key component to their success.

2.2. Plan the Measurement Process

Total Metrics worked with the management team over a period of 4 weeks, using workshops and prototyping, to identify the key performance indicators (KPIs) needed to demonstrate improvement, quantify the benefits of re-factoring and highlight any weaknesses in their development processes. After several iterations of developing draft analyses and reporting structures, management agreed on the data requirements, an initial set of 27 metrics and the 5 fundamental base measures to support reporting requirements.

Five Fundamental Base Measures

1. **Functional size** of each enhancement project in each 3 monthly Release would be measured at project implementation by an IFPUG certified (CFPS) practitioner in [2]IFPUG 4.2.1 function *Un*adjusted points and tracked using [3]SCOPE Project Sizing Software™.
2. **Effort** hours (time expended) would be collected daily by each of the development team (Level 1 ISBSG [3]) and at the Release Administration (Level 2 ISBSG), using [4]NIKU software. Effort would be collected against each project activity e.g. Design, build, test etc. and also against rework and maintenance (bug fixing) during each project.
3. [5]**Defects found**, their **origin** and **severity** would be collected at all testing phases and also in the first month of production after implementation using the *Test Track Pro*[6] (TTP) Tool. It was decided to use an in-house weighted method for defect recording to enable quality comparison i.e. each defect was weighted for their severity (Critical = 10, Major = 5, Average = 2 and Minor = 1). The weighted defects were also normalized (ie. Weighted Defects per 1000 ufps) which allowed comparative reporting of quality across projects and releases. Unless otherwise stated, defects reported in this paper are 'normalized weighted defects'[7].
4. **Full Time Equivalents (FTE) –** was measured as the number of personnel that work 40 hours per week to provide support for the ALRS application for the 3 month period. Part-time personnel effort was converted to full time equivalents.

[2] International Function Point Users Group Counting Practices Manual Version 4.2.1 and ISO/IEC 20926
[3] Total Metrics Pty Ltd – www.totalmetrics.com
[4] CA Clarity™
[5] Defect –A failure of some part of an application.
[6] Seapine Software, Inc.
[7] Ie. **1** *Major* defect found in a 1000 fp project = **5** normalised weighted defects

5. **Duration** (Calendar months) – elapsed time from project initiation to project delivery.

To ensure comparability with Industry figures data definitions for the base measures were aligned with those defined within the International Benchmarking Standards Group.[3].

Industry Data against which ARLS was benchmarked was primarily sourced from the International Software Benchmarking Standards Group Release 9 (Jan 2002) and Release 10 of the ISBSG repository (Jan 2007) [4].

It was agreed that the Department would be responsible for all data collection of Effort and Defects and Total Metrics CFPS metrics consultant would be responsible for the functional sizing data collection and for all the data analysis and metrics reporting.

The step of Planning the Measurement Process and establishing the framework to proceed with measurement consumed 22 days of effort from an experienced senior consultant over an elapsed time of one calendar month.

2.3. Perform the Measurement Process

2.3.1. Establishing the Baseline

The original baseline functional size of the ARLS system was determined to be 10,455 ufps. The FPA analyst used the function point recording software (SCOPE) to develop a functional model of the ARLS system to 8 subordinate functional levels, the lowest of which comprised over 1600 [8]elementary processes. The initial baseline was counted as a Level 4 industry default count [2] where transactions are defaulted to 'average' complexity and data groups to 'low' complexity. Transactions and data groups were cross-referenced to each other where possible. The functional decomposition provided a framework to which subsequent change requests could be easily and quickly mapped and counted. The baseline size is currently 14,009 fps.

The step of developing the functional model, functionally sizing and documenting the baseline count of the ARLS system consumed 33 days of an experienced senior metrics consultant's effort over an elapsed time of 8 weeks.

2.3.2. Ongoing Measurement

After establishing the initial baseline metrics, measures were then collected for each quarterly Release. The productivity and quality are compared as part of a bi-annual 'Release on Release' Benchmark Report.

Effort, duration and defects are collected as part of the development process activities.

The functional size of each Project in a Release is:

- '*approximated'* early in the Release cycle to assist in quantifying the Release size and estimating Release Effort.
- '*measured'* after the implementation of each Release.

[8] See IFPUG 4.2.1 CPM definition for an Elementary Process

The baseline functional size is automatically updated by the net results of the Change Requests at the completion of each Release. If a project fails to be implemented within a Release then the software enables the count to be 'held over' to the following Release, without having to re-count.

Estimating the functional size of each of the projects (~6) that make up the Release and estimating the Total Release effort consumes on average 1 day of an experienced senior Metrics consultant's effort each 3 months.

Measuring the functional size of each of the projects (~6) that make up the Release (846 fps) consumes on average 5 days of an experienced senior Metrics consultant's effort each 3 months.

2.3.3. <u>Analysis of Results</u>

To date, data from 42 Enhancement projects, implemented in 8 Releases have been evaluated and reported in 4 Benchmark Reports over a 2 year period. The metrics are derived from the raw base measures data using statistical and reporting functions within MS Excel®

Setting up the analysis of the metrics results and developing the graphs for the projects that make up the two Releases in a Benchmark consumes on average 5 days of an experienced senior Total Metrics consultant's effort each 6 months.

2.3.4. <u>Reporting the Results</u>

A Benchmark Report is produced on a Quarterly basis which reviews each KPI *(49)* and interprets the results from the perspective of what the data is showing about the strengths and weaknesses of the project development life cycle processes. Hypotheses on the root cause of observed phenomena are put forward and recommendations are made as to how to improve the process.

Assessing the results and writing the Benchmark report (~110 pages, including 50 graphs and tables) consumes 12 -15 days of an experienced senior Metrics consultant's effort each 6 months.

2.4. Feedback into Technical and Management Processes

Each successive Benchmark Report has highlighted areas where the most cost effective improvements could be made and the client has responded by focussing on these areas and in most cases has implemented the report recommendations.

Two years on, the re-factoring activity has only just been approved after a proof of concept pilot study. Due to the limited amount of re-factored functionality, the four bi-annual Benchmarking reports so far have had only limited capability of commenting on the effectiveness of the re-factoring project. However the analysis of the measures, as part of the Benchmarking activity, has provided considerable insights into the strengths and weaknesses of the current ALRS development process and the recommendations from the reports have resulted in a number of changes to the process which has shown measurable improvement in quality and productivity.

A summary of the issues and observations on the Key Result areas from the Reports and the outcome from implementing the recommendations is described below.

2.4.1. Product Quality

2.4.1.1 Observations

The overall quality of their development process and the time the development team spent in testing compared favorably to industry. The total number of un-weighted normalized defects found in production was consistently <10 un-weighted defects per 1000 fps compared to 23 un-weighted defects found in comparative environments in the ISBSG data. Whilst quality of the delivered product has improved over time the metrics analysis raised concerns regarding the number of defects being injected and the low defect removal efficiency at early testing stages.

2.4.1.2 Lessons Learned

Measuring the number, severity and origin of defects found, has highlighted the following areas for improvement in their development process:

- **Code inspections and Unit Testing** were not detecting coding defects – Most of the defects found later in the development process and also in production were injected in the ‘build phase’ which includes coding and unit testing.
- **Testing** was introducing a significant number of defects – Testing phases (Product stability testing, System testing and Integration testing) were the next largest contributor to delivered defects. Ie. These were defects identified and fixed during the testing phases but often the implementation of the fix introduced even more defects.
- **Testing Efficiency** was significantly below Industry standards of 90% – System and Integration testing were each only removing around 70% of the defects, leaving up to 70 normalised *weighted* defects still being found in the first month of production.
- **Time spent in Early Life Cycle activities** was less than expected when compared to Industry effort profiles and appeared to be a contributing reason for early life cycle defects. A significant number of defects originated in analysis and design. Time typically spent in analysis and design was 15 – 23% which is significantly less than ISBSG Design = 27%. Many of these defects were not being found until System or Integration testing indicating more time needed to be spent in design and on design reviews.
- **Process Variability –** there were large variances in percentage of project effort spent in the different project Phases, indicating a lack of consistency in the project development life cycle process practised by different project teams. Lack of repeatability of the process potentially contributed to some of the wide variations in project quality and productivity experienced.

2.4.1.3 Improvements Introduced

- **Peer Reviews and Inspections and a more formal Unit Testing Process –** The first two Benchmark reports indicated that immediate improvements were needed in the Build and Unit test process to reduce the overall number of Build defects being injected. The introduction of Peer Reviews and a more formal process for Unit testing in the 3rd Benchmark period, resulted in a 66% reduction in the number of Build injected defects and no defects originating in the Build phase being found in production.
- **Increased Focus on Capturing defects earlier in System Testing –** In the 3rd Benchmark a focus on Testing efficiency resulted in System Testing removing 90% of the defects compared to 60 – 70% in previous benchmarks. More than 87% of Build defects that had been injected were found in System testing and almost all were found and fixed prior to implementation compared to previous Benchmarks where up to 36 build defects were in the delivered product.
- **Introduction of a Formal Requirements Management and Design Process –** The lack of rigour and consistency of the development process, particularly in the early development phases was addressed by introducing the [9]DOORS Requirements Management tool. Developers were formally trained in its use and it is now the standard method for the development of Requirements and Design Specifications for all projects. The impact of implementing a more rigorous approach to Requirements analysis and design has yet to be realised as the number of defects originating in design had not decreased in the 4th Benchmark report, although the design defects are now being found earlier in System Testing compared to being found late in production.

2.4.1.4 Changes to the Defect Data Capture Process

The 3rd Benchmark Report results were shared with the project team who immediately identified areas where they believed that the defect data was incomplete or inconsistently collected. Realising the importance of accurate collection and reporting of defect data the following changes were implemented.

- **Improved Recording of Early life Cycle Defects –** Teams now capture and report *all* Defects not just those found after the Build Phase in formal testing – Many defects had not been previously recorded particularly in cases where the project team considered the defect trivial or if they were defects in the documentation eg. Analysis defects.
- **Recording of Defects for Unit Test –** Teams now capture and report all defects found in Unit Test and Extended Unit Test – So that these could be included in the Benchmark figures and provide a more complete picture regarding the origin of Defects, particularly early life cycle defects.
- **Changes to the allocation of the way defects are recorded –** After the results of the 4th Benchmark, the department continued to review the way defects were being recorded. In the past, when defect were identified they had been recorded against testing phase the project was "officially" in rather that the type of testing being

[9] Telelogic DOORS®

performed. Ie, if a User found a defect whilst acceptance testing but the project was meant to be in integration testing then it was recorded as being found in Integration testing.

2.4.1.5 Discussion

Whilst significant improvements in quality could be seen after the 3rd Benchmark, any improvements to product quality were masked in the two Releases comprising the 4th Benchmark Report since the project teams had increased the rigour and scope for collecting defect data resulting in more defects being recorded. This made it difficult to draw any conclusions when comparing the latest data with earlier benchmarks. However it is anticipated that this 4th Benchmark will reset the baseline level for defects against which further improvements can be measured. It also highlights the issues with continuous improvement and the difficulties assessing trends over time when processes are changed as part of the improvement.

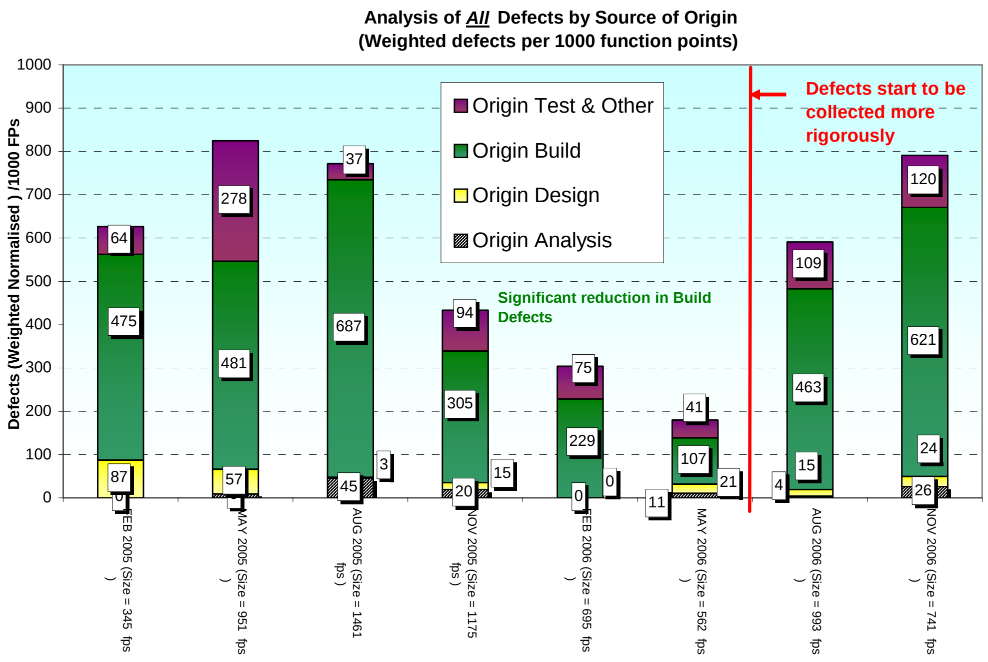

Figure 2 Positive Impact on Number of Defects originating in the Build Phase

2.4.2. Productivity

2.4.2.1 Background

Productivity was measured at both the Project Level and overall Release Level. Where the Release Level included all overhead effort (eg project management, planning) involved with implementing the Release. Project sizes varied from the very small (16fps) to large (635 fps). (Median = 87 fps, average = 141 fps). The quarterly Release sizes varied between 502 and 1461 fps with a median size of 846 fps.

2.4.2.2 Observations

The median project delivery rate across the 42 projects is 14.9 hours per fp. This lies closer to the Industry 3rd quartile range for Cool:GEN projects (12.5 hours per fp) than the median value (9.1 hours per fp). The lower than median industry rates for productivity can partially be explained by the following contributing factors which have been reported to have a negative impact on the enhancement productivity rate:

- **Very large size** (>14,000 fps) of the ALRS application ie within the top 1% of all applications size. Large applications tend to be more architecturally complex and harder to maintain.
- **The age (>14 years old)** – Older applications tend to be less structured, more poorly documented and no longer have the original development team available for consultation, making implementation of changes more time consuming.
- **The hierarchical management structure** – Bureaucracy of government departments impedes fast turnaround in decision making.
- **Multi platform application** – Primarily Cool:GEN with 15% is Java J2EE Internet components.

Whilst Project Productivity has not shown any significant improvement over the 2 year Benchmark Period there has been improvement at the Release Level (where effort includes all Release Overhead effort and an aggregate of Project size). The lack of significant improvement at project level despite some limited re-factoring and the implementation of process improvement initiatives has been attributed to a number of factors:

- **Learning Curve –** The extra effort time consumed by the implementation of new tools, techniques and technologies has slowed many of the stages of the life cycle.
- **Small Project Size –** The median project size has almost halved over the Benchmark period and for the last two Benchmarks has been around 80 fps, ie. half the optimum minimum size of 180 fps where highest productivity has been experienced. The decrease in project size has not been a deliberate decision but the result of the nature of the change requests during that period.
- **Including Extra Effort in Project Effort Data –** Quality Control (QC) effort for a project is now recorded against the individual project effort whereas in previous Benchmarks this effort was recorded in the Release Overhead effort.

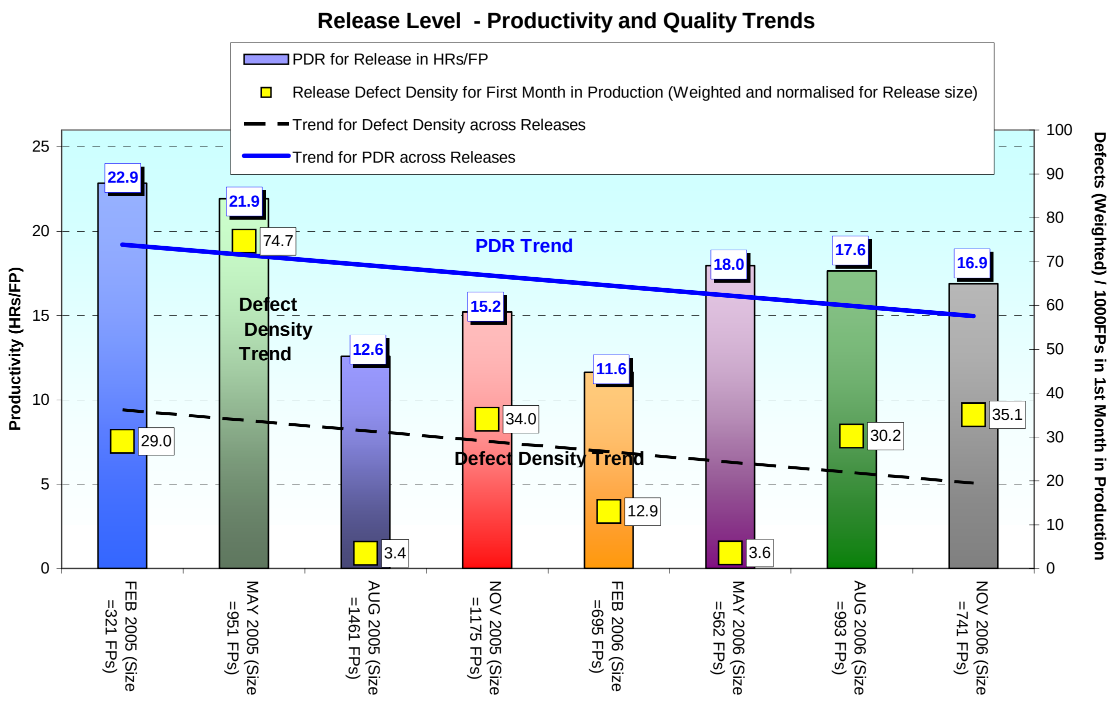

Figure 3 Positive Improvement in Release Level Productivity and Product Quality

2.4.2.3 Lessons Learned

Tracking productivity over the 2 years of the Benchmark activity has highlighted the following areas for the IT section to address:

- **Small Projects behave less predictably than larger projects.** Small projects (< 100 fps) experienced significantly more variance in productivity (4 to 69 hours per fp) than larger projects >180fps) – 9 to 13 hours per fp. This makes it harder to accurately estimate the effort and plan the resources to implement small projects.

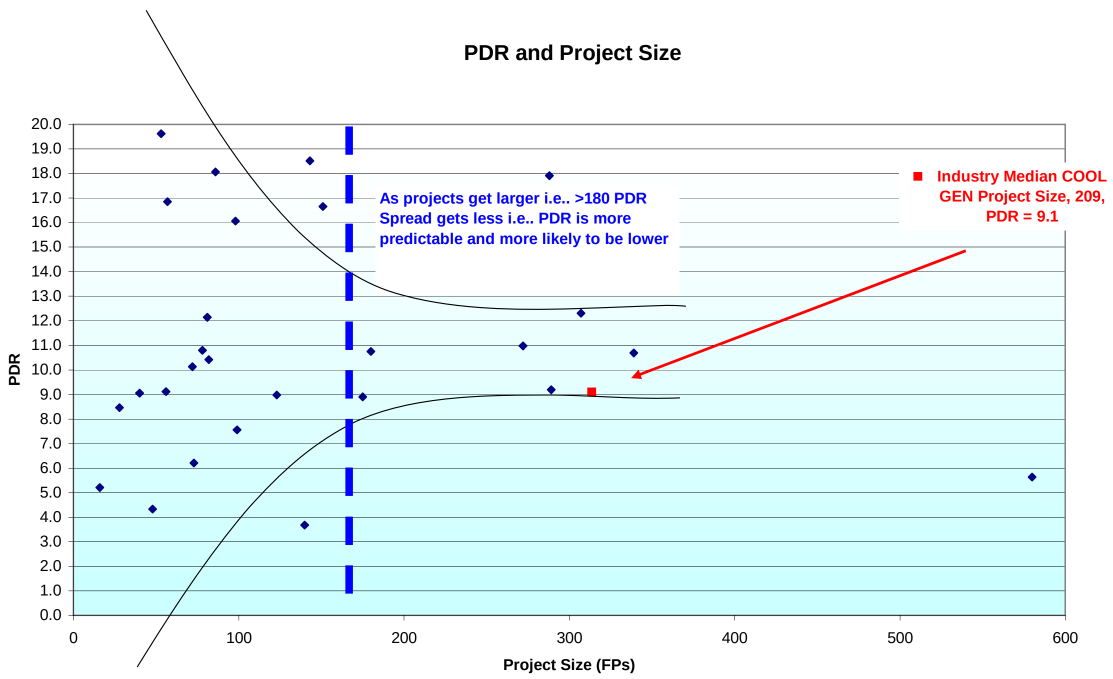

Figure 4 Large Variations in PDR demonstrated by Projects <180 fps

- **Small Projects are not as cost effective as larger Projects.** Small projects consistently exhibited lower productivity (Project delivery rates > 16 hrs per fp) compared to larger projects (Project delivery rates <12 hrs per fp). Since the Project size within each release is becoming progressively smaller (median project size has reduced from 200 fps to 87 fps) this has contributed to a negative impact on project productivity reported for the later Releases. It also highlights the high cost of implementing small projects, rather than functionally grouping the Change Requests so that the project size exceeds the minimum optimum size of 180 fps.
- **Large Projects (>250fps) are difficult to implement within the Users optimum delivery time of 12 months** – The business ideally would like all planned projects to be implemented within a 12 month time frame. Projects >250 fps typically take longer than 12 months from planning to implementation. Indicating that an ideal project size is 180 to 250 fps in order to optimise reproducibility, productivity and timely implementation.
- **Accurate Top Down Estimates of Release Size and Effort can be developed Early in the Release cycle –** Data collected over the 8 Releases shows that there is a strong correlation (r^2=0.658) between the project size in function points and the effort to develop the project. (Effort hours = 8.6* size + 573). When Project and Release sizes were 'estimated' at design stage via questionnaires the predicted Project and Release sizes were within 15% of the measured implemented size. The effort predicted by the Release size was within 10% of actual effort. The top down method of effort estimation using project size is very quick and is able to be completed in less than a day for all projects.

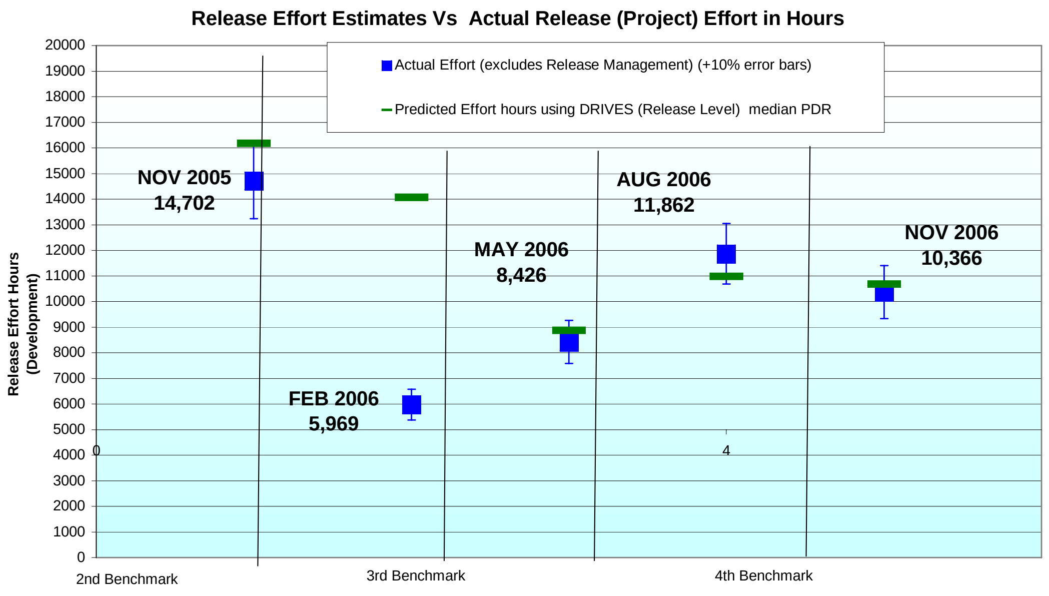

Figure 5 Release Sizes Estimated via Questionnaire Accurately Predicted Release Effort using Median PDR

- **Only around 33% function points counted for each Release delivers new Functionality to the Business** – Most projects make changes to existing functionality rather than delivering new functionality to the business. There is a negative trend in the overall Net Growth to the application's functionality delivered by a Release.
- **Extra time, effort and cost needs to be allocated to Projects with a high number of Stakeholders** – Projects with a high number of external stakeholders (>6) experienced up to 4 times lower productivity than projects with only 1 to 2 stakeholders.
- **When the implementation of a project is delayed there is significant additional cost to implement it in a later Release** – The cost of rework to retro fit the functionality to the latest version of the interfacing functionality is significant.

2.4.2.4 Improvements Introduced

In response to these observations in the 4th Benchmark Report the following initiatives have been planned.

- **Group Change Requests into Projects of Optimal Size** – Management has undertaken to work with the Business Users to plan new Releases so functionally related change requests are grouped into projects that fit within the range of 180 to 250 fps to optimise cost efficiency, estimation accuracy, project productivity and reduce project productivity variations.
- **Prioritise Work Requests to align with Strategic Business needs** – At the moment projects are initiated by sponsors based on funding availability rather being prioritised based on strategic direction.

- **Use the early life cycle estimates of size for Project Planning –** Currently estimates of project and release sizes are only used as a double check of work breakdown based bottom up estimates. It is planned to use the top-down size based estimating techniques earlier in the Release cycle to reduce risk of project cancellations or late implementation.
- **Profile Project Attributes** and their impact on PDRs – This enables project managers to consider the productivity impacts of particular attributes when estimating planned projects.

2.5. Evaluate Measurement

The effectiveness of the Benchmarking process, data collection, analysis and KPIs reported are reviewed on a 6 monthly basis and improvements implemented in the subsequent releases. Whilst the ongoing changes compound the difficulty of Release on Release comparisons it is agreed that this should not take precedence on improving the overall measurement process.

The Review process takes around half a day for 6 people in a workshop. Retrofitting the data and the measurement templates to accommodate the changes take between 1 to 5 days depending on the degree of change.

3. Critical Success Factors for the Measurement Program

Why is this measurement program a success where so many others fail? The following factors differentiate this client from many others we have dealt with over the past 13 years:

- **Measurement was implemented as a formal process –** Measurement and Analysis was formally recognised by the organisation and the implementers as an IT process and adequate budget, resources and time frames were allocated to it.
- **Clear Stated Objectives for Measurement –** The IT Management were conscious of the need to improve their accountability and the revenue from the ARLS system. They were already attuned to using measurement to monitor other processes and they had clear objectives for the measurement program. They recognised that measurement could objectively verify the benefits of their proposed process improvement initiative (ie refactoring activity).
- **Long Term Commitment –** The IT Management committed a significant budget for a 4 year period to support the program. This long term investment highlights the importance of the program to the participants who can be confident that their diligence and hard work collecting accurate metrics will not be wasted.
- **Management had Realistic expectations of:**
    - what the measurement program could deliver within the proposed frame
    - the time lag between collecting measures and being able to get useful feedback on their processes,
    - the amount of resources required to assist in the collection, validation and recording the data,
    - the impact on the capability to compare and determine trends whenever changes are made to the measurement process.

- **Act on the Results –** IT Management reviews the findings and recommendations of each Benchmarking Report and responds by focussing the improvement initiatives on the weaknesses exposed by the report.
- **Open to Change –** Although the original objective was to evaluate the benefits of their re-factoring initiative the client has expanded the scope of the program to provide insights into their whole development process.
- **Commitment is at all Levels –** The program has champions from senior management all the way down to their project managers and team leaders who enthusiastically believe in the benefits of measurement, and respond to the information it provides.
- **Accepts Bad News –** The IT Management accepts bad news on their performance as identification of an improvement opportunity rather than a reason to cancel the program or attribute blame to their team.
- **Makes Measurement Important –** Does not let other project priorities get in the way of the need to collect the measures. Recognising the importance of measurement management has assigned a senior staff member to be responsible for the Program and has contracted experienced metrics consultants to provide the analysis.
- **Trains Staff in Collection of Measurement Data –** This is a continuous process, required for each new staff member or if data collection methods are revised.
- **Shares the Benchmarking results with the developers –** The IT Management welcomes contributions of ideas for improvements. When staff are aware that the measurements that they collect are monitored, they ensure that they are accurate and consistent and provide valuable feedback into the process.
- **Provides Resources to set up the Framework for Measurement –** The management committed time and resources to develop an accurate baseline function point count of ARLS on which enhancement counts can be applied. They purchased tools and the infrastructure necessary for the success of the program.
- **Commits to Ensuring Quality Data –** Committed Resources to automate the configuration control and facilitate reporting of Function point counts.
  Function point counts are:
  - recorded in a software tool that enables ongoing configuration control of Release Function point counts and individual change requests can be measured concurrently on the same baseline. This automation of count recording ensures the accuracy and consistency of project size measurements. It also enables re-factored functionality to be ‘flagged’ so that future projects can track productivity improvements when projects impact re-factored functionality,
  - measured by a single certified experienced counter to reduce variations in measures size.

  Effort and Defects are also recorded in Software Tools to assist in easy accurate collection.
- **Is committed to building quality Software, cost effectively –** They only use experienced developers who develop high quality documentation which assists not only in the accuracy of the function point counting but also in the quality and repeatability of the development process.
- **Contribute to ongoing Process Improvement of their Measurement Process –** They constantly work towards not only improving their development processes but also the measurement process to ensure its consistency and accuracy. If they need to change the way they partition their work and the measures need to be changed, then

they spend the time to ensure data is retrofitted and that staff are trained in the new data collection procedures.

4. Conclusion

Measurement has enabled this organisation to focus their process improvement to get maximum benefits for productivity of their development team and quality of their delivered product. This translates to their bottom line enabling them to deliver more functionality, faster and cheaper to their users since a better quality product means higher user satisfaction and potentially more revenue from higher utilisation and less down-time.

Total Metrics works with many organisations that recognise the importance of measurement but are not committed to spending the time, money or skilled resources to implement it as a formal process. They treat measurement as an end in itself rather than a monitoring tool for their critical processes and consequently when the measurement program consumes resources without providing satisfactory returns in the short term, it is hastily cancelled. The program described within this paper is successful primarily because the Management understood the importance of implementing measurement as a formal process and the financial benefits that measurement could provide. They have invested the time and committed the financial and personnel resources over the longer term to plan and improve the program and are committed to its long term success.

References

[1] Goethert, W., Hayes, W - Experiences in Implementing Measurement Programs November 2001 - Software Engineering Measurement and Analysis Initiative - Technical Note CMU/SEI-2001-TN-026 Carnegie Mellon University

[2] Morris, P.M., Levels of Counting IFPUG Conference Las Vegas 2001 (USA) *http://www.totalmetrics.com/*

[3] International Software Benchmarking Standards Group - Glossary - *http://www.isbsg.org/html/terminol.html*

[4] International Software Benchmarking Standards Group - ISBSG Estimating, Benchmarking and Research Suite R10 - *http://www.isbsg.org*

[5] ISO/IEC 15939: Software Engineering - Measurement Process: 2002