Lies, Damned Lies and Statistics Software Metrics

Author: Kawal S Banga

Abstract

Most software organisations have at some time or other implemented a metrics programme. Some of these organisations have managed to make significant improvements in software development as a consequence of an effective metrics programme. However, experience shows that most metrics programmes either fail completely or at most achieve limited success. The author contends that one of the major contributing factors for this lack of success is that staff in these organisations, particularly senior managers and directors, either do not understand how to collect, interpret and utilise metrics or deliberately misrepresented and misused metrics for political reasons. This results in a lack of confidence in the metrics programme, particularly by those who are expected to provide the raw metrics data. Eventually this downward spiral results in the demise of the metrics programme.

This paper documents the author’s experiences of how metrics have been misused in a variety of business sectors including banking, insurance, transportation, utilities and a software house. The paper will also draw on the findings of the author’s as yet unpublished research on “An Investigation into the Training and Use of Measurement in Software Engineering” which surveyed the Times Top 1000 companies and UK Universities.

Metrics practitioners and organisations implementing metrics programmes need to understand the potential for the misrepresentation and misuse of metrics in order to avoid such pitfalls. The author will also discuss various approaches that can be utilised to ensure the success of a metrics programme.
Table of Contents

1. INTRODUCTION 1

2. SIZE, EFFORT AND SCHEDULE 1
   2.1 The Accuracy and Use of FP Counts 1
   2.2 Backfiring 2
   2.3 Omissions from Effort Data 2
   2.4 The Elusive Staff-month 3
   2.5 Project Elapsed Time (Schedule) 3

3. BENCHMARKING 4
   3.1 The Metrics Database 4
   3.2 External Benchmark Data 6
   3.3 The Hawthorne Effect 7

4. PRESENTING THE ANALYSIS AND FINDINGS 7
   4.1 The Audience 7
   4.2 Lies, Damned Lies and Averages 7
   4.3 Productivity, Defect and other Trends 8
   4.4 Normalising Data 8
   4.5 The Elusive Single ‘Silver Bullet’ Measure 10
   4.6 Feedback 11

5. CONCLUSION 11

6. APPENDIX A: BACKFIRING 14

7. APPENDIX B: DEFECT CLASSIFICATION SYSTEMS 14

8. APPENDIX C: AVERAGES 15

9. REFERENCES 16
1. Introduction

In 1990 Rubin [1] reported that out of 300 major US IT companies that had implemented measurement programmes, only sixty were successful. In 1995, Rubin [2] conducted further research that revealed that most IT measurement programmes fail within 8-15 months of initiation. Every metrics guru will have his or her own list of reasons why so many metrics programmes fail i.e. a list of do's and don'ts. The author’s experiences lead him to believe that one of the major contributing factors for this lack of success is that senior managers and directors, do not really understand the intricacies of gathering, interpreting and using metrics. This paper will give real live examples of the abuse of software metrics.

2. Size, Effort and Schedule

2.1 The Accuracy and Use of FP Counts

Research conducted on behalf of IFPUG [3] showed that FP counts are only accurate to within plus or minus 10%. Therefore an organisation developing 10,000 FPs in a year, at say a cost of £500 per FP, could be up to £0.5M out. Thus budgets, outsourcing costs/deals and estimates would be severely impacted.

Research conducted by the UKSMA (UK Software Metrics Association) [4] found a variation of plus or minus 22.5% between FP counters. Novices (less than four years experience) had a inter-counter variation of 25.9% and experts (four or more years experience) an inter-counter variation of 12.8%. The research also indicated a tendency to under count, experts by 8% and novices by 17%. This and other research [5] indicates the greatest errors made in counting function points are not misinterpretation of the counting rules, but a misunderstanding of the requirements. This can be addressed by making an expert user available to the FP counters.

The author’s personal experiences include the following:

- In one organisation both IFPUG and Mark 2 versions are used, depending upon who does the count. Unfortunately the significance of the difference is lost on many of the managers and project managers who do not have the necessary metrics backgrounds.

- A case study in a book [6] describes a project which claimed to have achieved a productivity rate of 78 Function points per man-month, using Mark 2 FPs. It was then stated that this compared favourably with the ISBSG (International Software Benchmarking Standards Group) median value for all projects of 5.7 hours per FP (21 FP per man-month). However, the vast majority of projects in the ISBSG database are based on IFPUG FPs and not Mark 2 FPs. The latest version (Release 5) of the ISBSG database was made in March of 1998. Of the 451 projects in that release only 18 (4%) were Mark 2 projects whereas 90% were sized using IFPUG.

- An organisation decided not to move from version 3.2 to 3.4 of the IFPUG counting rules as this would substantially lower their productivity rates. They stopped quoting the IFPUG version number whenever quoting the figures in the public domain. Yet another organisation applied the IFPUG supplied 3.4 to 4.0 conversion formula to their entire portfolio of applications, despite the fact that there were many applications that were purely batch, which would have the same FPs whether counted under the 3.4 rules or the 4.0 rules.

- Even though Lines of Code (LOC) have been discredited as an appropriate measure of size, particularly when comparing across different programming languages, many organisations, academics and authors insist on using it. For example, Ashley [7] quotes that the “Space shuttle achieved 0.1 defects per KLOC during first 3 years of operational use”. What does this mean? How can this be used as a comparative measure without knowing the particular language(s) used? A system with 0.1 defects per KLOC of assembler would generate at least a magnitude more errors than a system with 0.1 defects per KLOC of a 4GL, both systems delivering the same amount (measured in FPs) of functionality.

- In another organisation the author was requested to estimate the FP counts for some projects from the requirements and design specifications. Unfortunately these were eventually used as the actual counts in an external benchmarking exercise.
2.2 Backfiring

Another problem area is the source of the FP data itself. Not all companies actually count FPs, so they either estimate the FP counts or derive them from lines of code using backfiring. Backfiring is only accurate to plus or minus 25%, and in some extreme cases can vary by more than 50% [8]. Some organizations only sample an application and scale the count up accordingly. Such approaches are prone to high error margins. The main strengths and weaknesses of backfiring function points are listed in Appendix A.

Studies conducted in the 1970's by Allan Albrecht and his colleagues at IBM [9] found some interesting but not perfect correlation between source code size and function points for many programming languages. Figure 1 illustrates some typical ratios of logical source code statements and equivalent volumes of function points. This is a small excerpt from the main SPR table of languages with almost 500 entries [10].

![Figure 1: Ratio of Source Code Statements to Function Points](image)

Backfiring has become the most popular method for ascertaining function point sizes of ageing legacy applications. In fact, for many legacy applications backfiring is the only convenient method for developing function point totals because the specifications are often missing and the original developers have departed. However, some organizations are submitting data for external benchmarking with FP counts derived on backfiring.

2.3 Omissions from Effort Data

Jones [8] lists the following as the commonest omissions from historical data, along with the corresponding magnitude of the error:

- Unpaid overtime by exempt staff (up to 25% of reported effort).
- Charging time to the wrong project (up to 20% reported effort).
- User effort on projects (up to 20% reported effort).
- Management effort on projects (up to 15% reported effort).
- Specialist effort on projects (up to 15% reported effort).
- Effort spent prior to "turning on" the project tracking system for the project (up to 10% reported effort).
- Inclusion of non-project tasks (up to 5% reported effort).

This gives an overall error magnitude of 110%.

The author worked for one organization within which it is routine for projects that have exceeded, or will soon exceed, their authorized budgets, to record effort against projects that are on track to underspend their budgets. In another organization, one particular project had already had 3 false starts and had spent £1.5 M even before there was a fourth
attempt. There had been no attempt to even measure the cost of these false starts, until the author estimated the costs to-date based on limited timesheet returns from the past.

In another organisation managers deliberately excluded overtime effort (an average of 15%) so that the productivity figures were higher. Furthermore, staff were willing to work the extra hours unpaid for fear of becoming the next batch of individuals targeted for down-sizing within the organisation. This meant that future estimates based on this data were automatically flawed i.e. under-estimates of 15%.

Another common theme appears to be the deliberate omission of low productivity projects and/or projects that are cancelled, despite having utilised a large proportion of the IT departments overall resources. Typical reasons given to the author for excluding a project include the following:

- "This is a special one-off project which is not typical".
- "There was a deadline to meet and we didn’t have time to take any measurements”.
- "There was no time to include the project’s metrics in the benchmark - we’ll include it in with next years projects”.
- "This project was cancelled”.

An organisation may have a high productivity rate for those projects it does deliver, but may have a high incidence of cancelled projects. In order to make a true comparison between organisations, the organisational productivity i.e. Total FPs delivered divided by total effort of all projects should also be compared.

2.4 The Elusive Staff-month

The author interviewed 19 project managers within the same organisation and received the following definitions of a staff month for the purposes of estimating: 20, 18, 18.5 and 17 days. Furthermore, the definition of a staff day was 8, 7, 7.5, 7.4, 5 or 4.75 hours. In this same organisation the author discovered that some staff were recording gross hours (includes non-project time), and others net hours (excludes non-project time) on the effort tracking system.

In another organisation, the senior managers tried to compare the productivity of two projects, without realising that one team consisted wholly of contractors who worked 8 hour days and the other team consisted mainly of permanent staff who worked 7 hour days. This particular exercise was aimed at demonstrating that contractors were more productive than permanent staff and permanent staff should be replaced. Not only was the exercise flawed, it was very divisive and resulted in a lot of staff dissatisfaction and uncertainty.

In some organisations the effort expended is estimated because no detailed time-recording took place, or 'got lost'.

The various options used then are:

- The estimated effort at the start of the project.
- The original budget allocated divided by a notional standard daily rate.
- The total cost of the project (from financial records) divided by a notional standard daily rate.

2.5 Project Elapsed Time (Schedule)

Project elapsed time i.e. schedule can also be ambiguous. When does a project actually start? Most software projects go through an initial phase where clients, sponsors, development and project management staff spend time deciding upon the feasibility of the project, the approach, make or buy decisions and various other issues. Most projects, and their time recording systems, do not record any of this upfront effort. Similarly, the end of a project can also be ambiguous. Many projects fail to deliver the full functionality at the planned end date. A "mini-release" usually follows a few weeks or months later. What was the end date? The date when the original limited functionality release was delivered or when the “mini-release” with the missing functionality was released. The author has been involved in benchmarking studies in several organisations where the start and end dates where not clearly defined and had to be ascertained by talking to development staff, who did not always agree upon the same date. Jones [8] states that of the thousands of projects that he and his colleagues have analysed, less than 1% had a clearly defined starting point and more than 15% were ambiguous as to when they were delivered.
3. Benchmarking

3.1 The Metrics Database

There is a school of thought that states that if there exists a metrics database, then it must be useful. However, this is far from the truth.

The typical metrics database from a medium to large size organization will consist of a variety of projects - small to large in size, varying productivity rates, data from many different years, a mixture of new developments and enhancements projects and a variety of different languages - 2GL, 3GL, and 4GL with some a mixture of these. Some projects take two to three years to complete and will appear in the metrics database under year three. However, over a three year period processes, tools, technology and personnel may have changed significantly. Thus comparisons between such projects against others which started and finished in the same year i.e. year three may not be valid. Without this detailed understanding, the data is almost useless. Many organizations use averages derived from such databases and are thus not comparing apples with apples. For example, Jones [11] demonstrates clearly the need to distinguish between new development, enhancement and maintenance projects in figure 2. The productivity profiles are quite different.

![Figure 2: Productivity Profiles for New, Enhancement and Maintenance Development (Source: Jones [11], reproduced with permission)](image-url)

The author worked in an organisation that was proud of its metrics database which was actively being used for estimating and benchmarking. Average productivity figures were being quoted. However, a detailed analysis of the data revealed the true picture (figures 3 to 8).

![Figure 3: Range of Productivities in FFilSM of Projects in Metrics Database](image-url)
Figure 4: Productivity in FPs/SM vs Size in FPs of Projects in Metrics Database

<table>
<thead>
<tr>
<th>FP Range</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-250</td>
<td>98</td>
<td>68.1%</td>
</tr>
<tr>
<td>251-500</td>
<td>24</td>
<td>16.7%</td>
</tr>
<tr>
<td>501-750</td>
<td>11</td>
<td>7.6%</td>
</tr>
<tr>
<td>751-1000</td>
<td>5</td>
<td>3.5%</td>
</tr>
<tr>
<td>1001-1250</td>
<td>3</td>
<td>2.1%</td>
</tr>
<tr>
<td>&gt;1250</td>
<td>3</td>
<td>2.1%</td>
</tr>
<tr>
<td>Total</td>
<td>144</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Figure 5: Size of Projects in Metrics Database

<table>
<thead>
<tr>
<th>Year</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>42</td>
<td>29.2%</td>
</tr>
<tr>
<td>1995</td>
<td>58</td>
<td>40.3%</td>
</tr>
<tr>
<td>1996</td>
<td>35</td>
<td>24.3%</td>
</tr>
<tr>
<td>Unknown</td>
<td>9</td>
<td>6.3%</td>
</tr>
<tr>
<td>Total</td>
<td>144</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Figure 6: Currency of Projects in Metrics Database

<table>
<thead>
<tr>
<th>Type</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhancement</td>
<td>101</td>
<td>70.1%</td>
</tr>
<tr>
<td>New</td>
<td>39</td>
<td>27.1%</td>
</tr>
<tr>
<td>Unknown</td>
<td>4</td>
<td>2.8%</td>
</tr>
<tr>
<td>Total</td>
<td>144</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Figure 7: Type of Projects in Metrics Database
3.2 External Benchmark Data

Commercial organisations such as CSC Index or Gartner Group offer external benchmarking services. However, organisations that make use of such benchmarks should be aware of all of the issues already discussed in this paper regarding the quality of data. The author is aware of at least one organisation that put forward its best projects. It was of course no surprise that they came top in their industry sector with an average productivity of 25 FPs/SM. However, this backfired on the IT organisation when they were "blowing their own trumpet". A director of one of the business units that was supplied by the IT organisation queried why they claimed an average productivity of 25 FPs/SM, but were using a value of 5 FPs/SM when estimating a new project. In fact some organisations measure only those projects they intend to submit for a benchmark assessment. This can give a very distorted view if the findings from the limited number of projects is used to assess the capabilities of the software development organisation.

Another subtle difference is between FPs developed and FPs delivered i.e. the level of re-use. For example, an off the shelf package may be 10,000 FPs but only cost ten staff months of effort to customise and install. This gives a productivity of 1,000 FPs per staff month (FPs/SM), which of course is ludicrous.

When the benchmarking company does not have sufficient data for the specific industry sector, they simply use data from across different industry sectors. This again is flawed as different industries tend to have different productivity and quality profiles. Furthermore there are differences between the standard hours and days worked by various industries, not to mention international and cultural differences. Comparing metrics from different countries can be particularly difficult for this reason. For example, the Japanese working week is 5.5 days and the Japanese do far more unpaid overtime than US or UK averages. The UK has 20-25 days paid vacation per year which is far more than in the US.

It is not always obvious how industry average graphs have been produced. Different sources of these industry trends have different shapes. These may have been produced with linear regression analysis or other techniques. The source of data wills have differing levels of quality and "data cleansing" and there is no way of knowing whether or not the constituent projects were homogeneous or not.

There are no consistent industry wide standard definitions for even the most basic of software metrics e.g. size, effort and defects. Size and effort have already been discussed in this paper. With regards to defects, the author has come across many defect classification systems and the number of categories of defect have numbered from 3, 4, 5 and 6 (see Appendix B). The definitions of these defect category systems have not been compatible.

Another common scenario is as follows. A benchmark shows an IT department in a good light. They introduce new tools, techniques or methods and discover that productivity has fallen dramatically. This is of course as one would expect, as the introduction of new methods of working requires training and familiarization before staff fully realise the benefits. The IT department either can't explain the figures to the boardsenior management. or think that they won't be keen on learning that the £500,000 spent on new tools has not delivered the promised 25% productivity increase. So, they try to bury the results of the latest benchmark. In one organisation the author was instructed by management not to distribute copies of an external benchmarking study to senior management as it showed the organisation in a bad light.

<table>
<thead>
<tr>
<th>Language Type</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2GL</td>
<td>5</td>
<td>5.5%</td>
</tr>
<tr>
<td>2GL and 3GL</td>
<td>22</td>
<td>15.3%</td>
</tr>
<tr>
<td>3GL</td>
<td>26</td>
<td>19.4%</td>
</tr>
<tr>
<td>4GL Standalone</td>
<td>15</td>
<td>10.4%</td>
</tr>
<tr>
<td>4GL, Client Server</td>
<td>25</td>
<td>17.4%</td>
</tr>
<tr>
<td>4GL and 4 CS</td>
<td>7</td>
<td>0.7%</td>
</tr>
<tr>
<td>Unknown</td>
<td>2</td>
<td>1.4%</td>
</tr>
<tr>
<td>Total</td>
<td>144</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Figure 8: Language Types of Project in Metric Database
One organisation was told repeatedly by its benchmarking company what it must do to further improve its performance. The same messages came out at the end of each subsequent benchmark exercise. However, the management team did not follow through on these actions. They were only concerned with using the data to show the main board how wonderful they were at delivering IT solutions.

3.3 The Hawthorne Effect

It is well known that the observation, and measurement, of activities involving people can influence the outcome of the activities, the so called Hawthorne Effect. The people involved will become interested in the measurement process and the measurements themselves and influence the final results. This is particularly so if the measurements will be used to appraise individuals. This can manifest itself in many ways in a software metrics programme. Low productivity projects deliberately being omitted from benchmarks, overtime being omitted from effort data, defects being fixed without being recorded, time being recorded against a different project’s budget and programmers not writing Help screens because there are so few FPs associated with Help screens.

This effect is not restricted to just the staff being measured. The author has experienced at first hand how management have apparently provided extra ‘hidden’ resources for projects which they knew would be measured and metrics consultants selecting projects on the basis of personal interest in the technology being employed.

4. Presenting the Analysis and Findings

4.1 The Audience

There are usually many different audiences for the metrics information, including, but not restricted to the following:

- Senior Managers and Directors.
- Internal and external customers.
- Line Managers and Project Managers.
- Metrics staff.
- Project staff.
- Other specialists such as estimators, planners and marketing staff.

The needs of these are all different and any reports and analysis should be customised accordingly. For example, senior managers and directors have no interest in details e.g. breakdown by project life-cycle but may be interested in the organisation’s bottom line cost trends. Project Managers on the other hand may be interested in details to enable them to better estimate future projects. In fact many senior managers just want a single number, figure or chart that tells them everything they need to know about the software capabilities of their organisation. Such simplistic approaches tend to make use of averages ...

4.2 Lies, Damned Lies and Averages

The authors experiences demonstrate that a large proportion of the metrics staff are not trained or experienced in the use of statistical techniques to analyse and report metrics information. Training is essential if the metrics are to be presented correctly and consistently, leaving no room for misinterpretation by the audience.

Statistically speaking, the average is a ‘measure of central tendency’. There are at least six types of average in common use:

1. The Arithmetic Mean.
2. The Mode.
3. The Median.
4. The Weighted Average.
5. The Geometric Mean.
6. The Harmonic Mean.

See Appendix C for a brief description of these with a sample data set and calculations.
Most metrics data is presented without any reference whatsoever to the type of average used. This can prove to be very misleading, particularly if benchmarking is being carried out. Quoting an average figure in isolation can be misleading and almost meaningless unless you know something about the sample or population from which the data is derived. A 'measure of dispersion' or variation should be quoted with the average, but rarely is. The most common measure of dispersion is the Standard Deviation. The standard deviation is a measure of how widely values are dispersed from the Arithmetic Mean value. The author has a preference for knowing the range i.e. minimum value and maximum value and number of data items used to calculate the averages.

The analysis in figure 9 is for the same data as that for figures 3 to 8.

<table>
<thead>
<tr>
<th>Average Type</th>
<th>1994</th>
<th>1995</th>
<th>1996</th>
<th>Unknown</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic Mean</td>
<td>14.2</td>
<td>30.3</td>
<td>38.9</td>
<td>59.0</td>
<td>29.5</td>
</tr>
<tr>
<td>Median</td>
<td>9.0</td>
<td>11.7</td>
<td>22.0</td>
<td>53.4</td>
<td>11.7</td>
</tr>
<tr>
<td>Mode</td>
<td>None</td>
<td>11.7</td>
<td>22.0</td>
<td>None</td>
<td>11.7</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>64</td>
<td>11.9</td>
<td>20.3</td>
<td>40.6</td>
<td>12.2</td>
</tr>
<tr>
<td>Harmonic Mean</td>
<td>8.4</td>
<td>6.2</td>
<td>10.5</td>
<td>23.7</td>
<td>5.9</td>
</tr>
<tr>
<td>Minimum</td>
<td>0.9</td>
<td>1.3</td>
<td>1.3</td>
<td>4.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Maximum</td>
<td>179.6</td>
<td>249.8</td>
<td>330.3</td>
<td>330.3</td>
<td>530.3</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>26.5</td>
<td>35.8</td>
<td>39.2</td>
<td>47.9</td>
<td>50.2</td>
</tr>
<tr>
<td>Weighted Average</td>
<td>5.1</td>
<td>9.5</td>
<td>6.7</td>
<td>12.8</td>
<td>7.5</td>
</tr>
<tr>
<td>Number of projects</td>
<td>42</td>
<td>54</td>
<td>35</td>
<td>9</td>
<td>104</td>
</tr>
</tbody>
</table>

Figure 9: Illustration of the variation in the values of the various averages using data from a metrics database.

The weighted average is determined by summing the FPs of all projects and dividing by the sum of the Effort of all projects. This gives a measure of organisational productivity. The value of the weighted average can be heavily influenced by a few large unsuccessful projects and can negate the high productivities of many smaller projects. The smaller more successful projects become invisible and the organisation is seen as a poor performer on the basis of a few large unsuccessful projects.

Any good book on statistics [12, 13, 14] will provide more details on these and other statistical techniques.

4.3 Productivity, Defect and other Trends

Many, if not most, organisations tend to show trend charts without too much thought. Many of these use such charts for political reasons, to demonstrate to internal management or perhaps external customers how wonderful they are at developing quality software.

A particular organisation appeared to show an upwards trend over many years with regards to productivity. However, it suffered from many of the problems listed above. Also, in the latter years, a lot of package implementations and small projects have helped to artificially boost the productivity rates.

Many defect charts show all defects consolidated, whereas the customer may only be interested in severity 1 (system unusable) and severity 2 (some functions unusable) defects i.e. those that really impact their business. The author has seen such a trend chart in an organisation where the actual trends in severity 1 and 2 defects were getting worse, but the overall downward trend was due to severity 3 and 4 defects getting less frequent. (see figures 10 and 11). Furthermore, there is no attempt to normalise the data i.e. defects per FP. It may be that they are getting smaller numbers of defects simply because they are writing less or smaller volumes of code.

4.4 Normalising Data

A naive use of data is found when a metrics programme reports the number of requirements changes. Typically, the aim is to reduce the number of requirements changes during the project life-cycle. Again, this can be potentially meaningless, if not misleading, unless normalised data is used. For example, which is better, a project with 50 requirements changes or one with 50? How about a 1,000 FP project with 10 changes and a 1,000 FP project with 20?
changes? Even this is not sufficient information. What is important is not the numbers of change requests, but requested changes measured as a proportion of the original project size. For example a 1,000 FP project with 10 additional requirements totalling 150 FPs gives 15% requirements creep. Another 1,000 FP project with 20 changes that total 100 FPs gives only 10% requirements creep. This is fine for setting targets for project requirements creep e.g. 10% or less. However, if we are interested in determining the total costs of requirements creep for a particular context in which the data is to be used to determine the way the data is presented and used.
4.5 The Elusive Single 'Silver Bullet' Measure

When it comes to software development, there is no single measure that tells the whole story. Mah and Putnam [15] claim that the use of simple ratios such as productivity rates is a "Two dimensional flatlands view for a multi-dimensional terrain problem". For example, it is quite possible to improve productivity dramatically, but at the cost of lower quality and lower customer satisfaction. Similarly, compressing the schedule too much by adding more staff may well result in a quicker delivery, but a lower productivity rate as well as lower quality i.e. a higher defect rate. The use of ratios implies a linear relationship, which is far from the truth. For example, a project cannot be delivered in half the time by using twice the people.

The Business Scorecard is a good tool for implementing a metrics programme. A typical approach is described below. All scorecards have many indices, such as shown in figures 12 and 13. This ensures that a consolidated view of performance is considered, as opposed to attempting to assess performance from just one measure.

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial Index</td>
<td>&quot;Managing finances and cost and cost reducibility&quot;</td>
</tr>
<tr>
<td>Quality Index</td>
<td>&quot;Delivering products and services to agreed specifications&quot;</td>
</tr>
<tr>
<td>Productivity Index</td>
<td>&quot;Improving size for money and capability&quot;</td>
</tr>
<tr>
<td>People Index</td>
<td>&quot;Managing the skill base and staff satisfaction&quot;</td>
</tr>
<tr>
<td>Client Satisfaction Index</td>
<td>&quot;Managing client satisfaction&quot;</td>
</tr>
<tr>
<td>Process Index</td>
<td>&quot;Improving the way we do business&quot;</td>
</tr>
</tbody>
</table>

Figure 12: Scorecard Indices and their descriptions

![Figure 12: Scorecard Indices and their descriptions](image)

Each of the indices will be formed from one or more measures.

However, there can be some negative aspects to the scorecard approach. For example a senior manager stated to the author that "We cannot afford to spend months carrying out FPA on our systems as Productivity measures only contribute 2% towards the final score-card score". In one organisation, directors set an improvement target of 20 FPdSM for development projects. This might be fine for those parts of the organisation achieving less than 20 FPdSM, but what about those already achieving in excess of 20 FPdSM?

An alternative attempt to show the multi-dimensional nature of measuring software development is shown in figure 14.
<table>
<thead>
<tr>
<th>Objectives</th>
<th>Processes</th>
<th>Metrics/KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Efficiency</td>
<td>Estimating</td>
<td>Delivery</td>
</tr>
<tr>
<td>Bid/Supplier Management</td>
<td>Process Improvement</td>
<td>Staff Satisfaction</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>Decision Support</td>
<td>Cost Effectiveness</td>
</tr>
<tr>
<td>Benchmarking</td>
<td>Qualify</td>
<td>Client Satisfaction</td>
</tr>
<tr>
<td>Size</td>
<td>Effort</td>
<td>Soft Factors</td>
</tr>
<tr>
<td>Unit Defects</td>
<td>Schedule</td>
<td></td>
</tr>
</tbody>
</table>

4.6 Feedback

One sure way of speeding up the demise of a metrics programme is not to provide feedback. Many metrics programmes are "black holes" in that data is collected but nothing is ever returned to the data providers. One possible effect of this is that projects start providing lower quality data. In some cases there is evidence that would suggest that some metrics have been made up in order to satisfy management's requirements for measurement.

5. Conclusion

Organisations that want to implement a metrics programme should use the services of a reputable metrics consultancy who can provide reference sites of successful implementations of such programmes. They should not repeat the mistakes made by other organisations, but learn from them. Such consultancies should be able to provide the full range of services from designing, implementing and maintaining a metrics programme as well as metrics training for all levels of staff.

The authors own research based on the Times top 1000 companies in the UK [16] this area has revealed that senior managers and directors, as well as other staff lack sufficient training (figures 15 and 16) to make effective and informed decisions based upon the findings of their metrics programmes. Senior managers and Directors of IT organisations must be trained in the science of measurement. They must also understand that introducing measurements will have an impact upon behaviour and attitude. Some behaviours will be positive, others negative. They will need to assess the potential impacts of the measurement programme and be ready to manage any negative behaviours. For example, only providing data for 'good' projects, effort being under-stated for projects and refusing to provide data due to "pressure of work or commitments". The greatest risk to any measurement programme is the senior manager or IT director who is not trained to understand how to gather, interpret and use metrics properly.
To ensure that we are indeed comparing "Apples with Apples", the Validity of any potential benchmark comparisons must be ascertained before time and money is spent. Validity of the metrics can be considered to be composed of four components: accuracy, completeness, consistency and currency. Consider the following examples using the size and effort metrics:

1. Accuracy e.g. FPs counted properly according to IFPUG rules and all effort/time recorded by individual staff to nearest 15 minutes.
2. Completeness e.g. all end-user functionality within the counting boundary has been counted and all user effort and overtime effort included.

3. Consistency e.g. all Projects have been counted using IFPUG rules and the working day consists of 7 hours, the working week of 5 days and the working month has 18 working days.

4. Currency e.g. all projects were delivered this year.

Information can be misrepresented, suppressed or even altered for political reasons. DeMarco refers to "Limbahuing the Data", defined as "To choose selectively from a body of data those items that confirm a desired result and never mention any that might be construed to confirm the opposite" [17].

The following guidelines should be heeded by those wishing to avoid these pitfalls in their own metrics programmes and for those wishing to check the quality of external benchmark data prior to comparisons with their own data:

- Ensure the vision, goals and objectives of the metrics programme are shared by all staff.
- Use staff trained in the use and collection of metrics.
- Use staff trained in the use of statistical techniques to analyse and report metrics information.
- Ensure the audience for the metrics is trained to correctly interpret and use the metrics.
- Concisely define the metrics to be collected, who is to collect them, their source, precision etc..
- Do not use metrics information to appraise, rank or punish anyone.
- Do not measure too many things (trying to "boil the ocean").
- Do not measure to too low a level of detail.
- Ensure that you use the collected data and feed back the results to the data providers i.e. avoid the 'data black-hole'.
- Ensure the validity of the metrics or benchmarks by assessing accuracy, completeness, consistency and currency.
- Publish only 'cleansed' data e.g. remove outliers.
- When publishing software metrics, clearly identify the statistical methods used and any limitations.
- Continually demonstrate the usefulness of the metrics collected to reinforce the commitment of all staff.

Remember, “Statistics don’t lie, but liars use statistics”, so you must seek independent verification of the quality of your own data and any benchmark data that you use.
6. Appendix A: Backfiring

The main strengths of backfiring function points [9] are:

1. Backfiring is extremely quick and easy to perform.
2. Backfiring automation is commercially available.
3. Backfiring is supported by many software cost estimating tools.
4. Backfiring is used in many software benchmark studies.

The main weaknesses of backfiring function point metrics [9] are:

1. Backfiring is of lower accuracy than normal function point counting.
2. Backfiring is ambiguous if the starting point is physical lines of code (LOC).
3. Backfiring may be ambiguous for mixed-language applications.
4. Backfiring results may vary based on individual programming styles.
5. Backfiring is not endorsed by any of the major function point associations.

7. Appendix B: Defect Classification Systems

Scheme 1
- A Show stopper
- B Major
- C Medium
- D Minor
- E Cosmetic
- Q Query

Scheme 2
- 1 Severely disrupted
- 2 Seriously inconvenienced
- 3 General irritation
- 4 Minor problem

Scheme 3
- 1 Critical
- 2 Major
- 3 Minor
- 4 Cosmetic

Scheme 4
- Major
- Minor
- Negligible
- Documented
- Unknown

Scheme 5
- Catastrophic
- Critical
- Significant
- Minor

Scheme 6
- Critical
- Significant
- Minor
8. Appendix C: Averages

- The Arithmetic Mean = $\bar{x} = \frac{\sum x}{n}$
- The Mode = the value that occurs most often
- The Median of a set of numbers arranged in order of magnitude is either the middle value or the arithmetic mean of the two middle values.
- The Geometric Mean = $\sqrt[n]{x_1 \times x_2 \times x_3 \times \ldots \times x_n}$
- The Harmonic Mean = $n / \sum \frac{1}{x}$
- The weighted average is determined by summing the FPs of all projects and dividing by the sum of the Effort of all projects.

The Mode and Median are used to eliminate the effects of extreme values.

<table>
<thead>
<tr>
<th>#</th>
<th>FPs</th>
<th>Effort (mths)</th>
<th>FPs/SM</th>
<th>Mean</th>
<th>Mode</th>
<th>Median</th>
<th>Geometric Mean</th>
<th>Harmonic Mean</th>
<th>Weighted Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>8.00</td>
<td>10.00</td>
<td>mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>15.00</td>
<td>10.00</td>
<td>mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>180</td>
<td>12.00</td>
<td>15.00</td>
<td>median</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>210</td>
<td>7.00</td>
<td>30.00</td>
<td>geometric mean</td>
<td>17.35626</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>280</td>
<td>8.00</td>
<td>35.00</td>
<td>harmonic mean</td>
<td>15.21739</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totals</td>
<td>900</td>
<td>50.00</td>
<td>100</td>
<td>weighted average</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 16: Illustration of the various averages using fictitious data
9. References

4. Rule PG, UKSMA Inter-counter Consistency Checking Experiment, Guild of Independent Function Point Analysts, 1997-98
5. Longstreet David H, Longstreet, Function Point List Server (function.point.list@rim.ca), Subject: “Re: counting issue: boundary and application design”, 24 Apr 96 21:01:17, reference to a paper written by Lacity and Subramanian (both of University of Missouri, St. Louis), “Determinants of Variability of Function Point Estimates” and a paper written by Mahmood and Gemoets (both of University of Texas El Paso), “Factors Affecting the Reliability of Function Points”
9. Jones Capers; Strengths and Weaknesses of Software Metrics, Software Productivity Research; Burlington, MA, USA July 29, 1997
10. Jones Capers; Table of Programming Languages and Levels; Version 8.2; Software Productivity Research, Burlington, MA, USA; URL http://www.SPR.com.