Towards a Successful Software Metrics Programme

Ray Dawson
Loughborough University, UK
R.J.Dawson@lboro.ac.uk

Andrew J. Nolan
Rolls-Royce, Derby, UK
Andy.Nolan@rolls-royce.com

Abstract

Based on the authors' 43 years of combined experience in industry, this paper describes a number of ways to ensure a metrics programme is considered successful. Experiences of a number of industries provide lessons on the planning of a metrics programme, the motivation of employees collecting the metrics, embedding metrics collection into everyday processes, presenting the metrics in financial terms and using metrics that already exist. It is acknowledged that metrics collected in industry can prove very little, but they are useful if used with other data or as a pointer for further investigations.

The lessons learned from these experiences form guidelines which, if followed, should give valuable assistance in achieving a successful software metrics programme.

1. Introduction

This position paper was presented at the 2003 conference on Software Technology and Engineering Practice, at the workshop entitled "Where is the Evidence? - The role of empirical practices in Software Engineering." It is based on the combined experiences of the two authors (eg. [1,2,3,4]). Andrew Nolan has worked in software engineering in a number of departments and businesses within the Roll-Royce company. Ray Dawson previously worked as a software engineer for Plessey Office Systems before moving to Loughborough University where he has since worked with a number of companies researching software engineering methods. Between them they have over 43 years' experience of software methods as practiced in UK industry.

The paper is anecdotal in nature, covering the social, managerial and organisational reasons why metrics programmes in software engineering have been successful or otherwise. While this paper offers little more than "story telling" as evidence, it does cover many practical experiences and draws lessons from them. The authors believe the lessons from this paper will be valuable to any practitioner attempting to use metrics for software process improvement, decision making, validating previous action or simply seeking to gain a better understanding of their processes. The paper does not attempt to cover the technical aspects of statistical analysis. While it is important to recognise the need for statistical validity for the analysis and presentation of metrics results, this topic is covered elsewhere [5].

2. Metrics and Motivation

Many metrics programmes in industry fail. Often it is because the metrics programme has been put in place for the wrong reasons. The authors have been asked on a number of occasions "How can I implement a metrics programme". This question, itself, can be a strong indication that the proposed programme will fail. A metrics programme is often desired simply because it is considered to be good to have one. The higher levels of the Capability Maturity Model [6], for example, require proper measurement of processes to be able to fully manage and optimise them. However, metrics themselves are not helpful unless they are the right measures to tell you what you need to know.

An example of this was given by an engineering company who had developed machine test software. The tests are carried out many times over on similar machines. In the interests of "good management" a metrics programme had been implemented, and eventually, after collecting metrics over many tests it was decided to use the metrics collected to see if changes made to the software produced a significant improvement. So, for the first time, the metrics were analysed.

The analysis very quickly showed:

1. The metrics were incomplete
2. The metrics were not accurate
3. The wrong metrics had been taken
4. It was impossible to produce the information required

Investigation into the reasons behind these problems showed that employees had to take some extra action over

Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'04)
0-7695-2218-1/04 $20.00 © 2004 **IEEE**

and above their core task of testing the machines to record the metrics. Although this action took little time (less than a minute) the employees could see no reason why they should bother. Consequently, many employees did not record the data consistently. Furthermore, when pressed by their managers to do so they usually had to fill in a backlog of missing data, which was then done from memory or even by guesswork, which was far from accurate.

This analysis had been made difficult because the data itself was not necessarily in the right form. Timings were required for various tests but the recorded metrics often grouped several tests together making it necessary to use average values, which further reduced the accuracy. Finally, the whole exercise proved to be fruitless as the majority of improvements made to the software had, for the convenience of the users, been introduced between a series of tests on one machine type and a series of tests on another type. The before and after data, therefore, were not comparable.

Had it been known what the metrics were to be used for from the outset, the data could be collected in the right form, the process improvement implementation could have been timed to give the necessary data and the employees may have been better motivated to collect data in a reliable and accurate way. This last point is particularly significant. Even if the right data and timings had been achieved it is still important for employees to feel there are benefits to be obtained from the extra work put in. Whether the employees would have been properly motivated by this exercise is questionable as the information sought has no obvious benefit to the company or to the employees. It is important, therefore, that a metrics programme is motivated by some clear business benefit if there is to be any prospect of motivating the employees.

There are a number of lessons to be drawn from this example. Asking how a metrics programme is to be implemented is focusing too much on the mechanics of metrics collection. It is also necessary to consider, what do you want to know, what are you going to do with the results and what will be the business benefit, otherwise there may be no benefit at all.

3. Embedded Metrics Collection

Engineers are generally busy people, they are often working to tight deadlines on high priority tasks. The example in the previous section shows they will resist any add-on process for metrics gathering as they will see it as diverting them from their more important tasks. In general, unless employees can collect metrics without trying, or it is easier to cooperate with metrics collection than it is to avoid it, the collection of metrics will haphazard even if the engineers are well motivated. To make metrics collection as painless as possible it must be embedded into everyday processes.

An example of embedded metrics collection comes from Rolls-Royce who implemented an electronic timesheet recording system. By putting the recording of time spent on tasks online, it became easy for the employees to access the timesheet than it was for the previous paper version. The introduction of pull-down menus made the timesheet quicker to fill in and the automatic calculation of totals required less input and made the input easier to check than before. All employees knew that they had to fill in a time recording system anyway so, because the new system saved time and effort, it was welcomed by all users. It could be said that the new system gave the company no new data that they had not been recording before, but the new system, being electronic made all the data available for analysis. The greater accuracy and consistency of the new system was an added bonus.

4. Metrics for Decision Making

One reason for collecting metrics is for managers to be able to make better, more informed decisions. However, managers in industry are normally driven by financial considerations. When given a choice of alternatives their bottom line is which will give the best value for money. If considering an investment, the bottom line is whether it will be worth the financial outlay. Given this financial orientation, it is perhaps surprising that metrics are, in the authors' experience, rarely expressed in monetary terms. Metrics may be gathered in terms of numbers of errors, volume of throughput, size of code or execution times and the engineers will try their best to explain what these imply for productivity, quality or systems support but there remains a huge communications gap. If metrics analysts were to go one step further to translate their metrics into financial costs and benefits the metrics would achieve far greater significance for the decision makers.

An example of the advantage of expressing metrics in terms of money comes from a training centre at a large electronics company. The company had two software lecturers who were struggling to keep up with the demand for their courses. The courses were for the companies own employees so there was no direct income resulting from the courses. Three times the senior lecturer put forward a case for a third lecturer. The first occasion was based on the workload of the existing members of staff. The management made sympathetic noises but no action was taken. On the second occasion metrics were used to make a case based on the number of extra people that could be trained and the backlog of engineers waiting for training. Again there was interest from the management but no action resulted. On the third occasion, metrics were turned into costs with the cost of man hours wasted through lack

of training and the cost of external training being compared with the costs of providing training. It was shown that the two existing lecturers each saved the company £180,000 every year and a third lecturer would bring similar savings. On this occasion the decision to recruit the extra employee was made immediately. The lesson from this is simple, present your metrics in monetary terms if you want management to take any notice of what you are trying to say. Fail to do so and your metrics are likely to become irrelevant!

When expressing metrics in costs it is important to consider *all* costs. If only a few costs or benefits are taken into account, these costs tend to take on an importance greater than they really deserve, giving a distorted view. This is illustrated by an example from a manufacturing company in the maintenance of their desktop computers. The company policy had been to replace each computer after four years of use. However, when the service agreement costs were considered, the fourth year was considered too expensive so the service contracts were only arranged for the first three years of each computer's life. Based on the simple costs of computers and service contracts this policy seemed sensible. However, a later analysis of the full costs involved took into account the cost of wasted manpower dealing with computer repairs in the fourth year, the costs of the computer unavailability during breakdowns and even the costs of users waiting for a response from older, slower computers. It was found that keeping the computers for four years with a three year service contract was one of the least cost effective options, it being better to keep each computer for less than three years and have a service contract effective throughout its company life.

This is relatively straight forward for tangible costs such as licences, hardware or manpower but it can be difficult to evaluate intangible costs such as customer dissatisfaction or loss of opportunity. However, a manager will inevitably need to make a judgement on the value of all these costs, no matter how intangible, in some form of "is it worth it?" decision. This means that however difficult and vague it may be, a fully analysed estimation of costs is bound to be better than a less informed judgement. Techniques used by the authors have included analogy with past experiences, finding substitute metrics that are measurable (eg. customer reorders as a measure of satisfaction) or working with upper or lower bounds to try and put a value to intangible costs, but no matter how vague these estimates are they have still proved to be better than no estimate at all.

5. Finding and Using Sources of Evidence

One possible means of eliminating the overhead of metrics gathering is to use the metrics the company already has. Many companies have mountains of data that have never been fully analysed. Error logs, change requests, time sheets, project spending records, project schedules and actual timings are all metrics that companies normally record. As stated earlier, there is the problem that for a particular purpose these metrics may not be in the right format or be adequate. However, before embarking on any new metrics gathering programme it is worth checking to see if the available data would be useable. For example, a manager in a retail services company did a statistical analysis of task estimates compared with actual timings to spot any anomalies[2]. In doing so he was able to identify areas that needed closer investigation and this, in turn, identified problems early enough in a project life to take remedial action. These simple metrics are available in most companies, yet this manager was able to use them for troubleshooting and, as a result, regularly finished his software projects on time and within budget.

The lesson here is that the problem may not be the lack of metrics for decision making and process improvement but a lack of analysis of the data that already exists. If no attempt has been made to analyse and learn from the data that is available is there any purpose in collecting yet more?

6. Expectations for Software Metrics

Software engineering metrics from industry pose a problem for the academic community - can they prove anything? For example, multiple tests on the same project are not practical as industry will rarely consider employing two teams to perform the same task. Test on different projects are never identical and nothing is ever repeatable under identical conditions. Staff turnover means that the same team cannot normally be used again with exactly the same personnel, and even if the same team members are available the learning from one test would affect subsequent tests. If a new development methodology is being tried, for example, all the metrics can prove is that it can work well, but there are too many unique factors about any industrial trail to be able to use statistics to show that it worked better than any other methodology, or that it would also work well elsewhere.

If a large number of multiple trials are carried out it may be possible to apply statistical analysis to prove the merits of a new methodology, but practical considerations will still make it difficult to eliminate other factors that could distort the analysis. Furthermore, when a sufficient number of trials have been carried out, in the fast moving software industry the methodology in question is unlikely to be considered new any more, and in reality it will probably have evolved and changed as experience in the methodology grew.

One method that university academics have used to produce software engineering metrics is to use tests on

students. A student body can produce a large number of tests that can be undertaken in parallel under identical, consistent and stable conditions. Unfortunately it is this very consistency and stability that means the tests are significantly different to the real industrial world where change is inevitable.

So, if the metrics obtained from industrial case studies can prove very little, does this mean the metrics programme used is unsuccessful? Whether it is successful or not depends on the expectations. Clearly, if the objective of a metrics programme was to obtain some form of proof that was unobtainable, then it was unsuccessful, but this does not mean that the metrics programme is not useful. Every piece of information can be a useful influence in decision making even if there is acknowledged uncertainty. A manager commiting to a contractual deadline would use his or her past experience of a similar project to decide whether to take on the commitment, even though statistically a single metric can guarantee nothing. In practice the manager will use all their experience and take into account a wide variety of considerations in making such a decision. The lesson here is that while metrics may not be able to provide any certainty, they can still provide useful evidence with or without any other information to enable a more informed decision than would otherwise have been possible.

Another illustration of the usefulness of metrics is shown by the retail services manager described in the last section. This manager did not use the metrics to prove there was a problem in any particular process. Instead, he used the metrics to give pointers as to where there could be problems. Further investigation was then needed to see if a problem existed or not. The manager used the metrics to give the starting point of the problem analysis, not the end result. In the experience of the authors, metrics analysis, if used as a focus for further investigation, can be an valuable tool for the software engineering manager.

7. Conclusion

There is a continual demand in industry for metrics to provide evidence of productivity, quality or costs, yet the implementation of many metrics programmes means that they fail to produce the data required. This paper has described a number of ways that can help a metrics programme to be considered successful.

1. Firstly, determine what the metrics are for and what will be done with them. This allows for better planning of the metrics programme giving a better chance of success.
2. Motivate employees with a metrics programme that has clear business value so that they will collect accurate and complete metrics.
3. Wherever possible, embed the metrics into everyday processes, making the data collection automatic or no extra effort, to enable complete and accurate metrics collection.
4. If the metrics are to be used by managers then express the results in financial terms. This means the results are more likely to be acted on.
5. Make sure all costs are considered, intangible as well as tangible costs, to avoid misleading results.
6. Check to see if metrics already exist that could serve the purpose required. It could be the lack of analysis, not the lack of metrics that is the problem.
7. Be realistic in your expectations on what you will gain from the metrics programme. The metrics on their own may not be able to show anything with any certainty. Be prepared to use the metrics with other data or as a pointer for further investigation to achieve your objectives.

The above guidelines have been based on the authors' many years of experience and, if followed, should greatly increase the chances of achieving a successful metrics programme.

8. References

[1] Nolan, A.J. (1999), Learning From Success, IEEE Software 16(1), 97-105

[2] Dawson, R.J. and O'Neill, W.P. (2003), Simple Metrics for Improving Software Performance and Capability: A Case Study, Softw. Quality J., 11,(3), 243-258

[3] Bradley, M. and Dawson, R.J., (1999), 'Whole Life Cost: The Future Trend in Software Development', Software Quality Journal , 8(2) ,121-131

[4] Jackson, T.W., Dawson, R.J. and Wilson, D., (2001), "The Cost of Email Interruption" , The Journal of Systems and Information Technology , 5(1), 81-92

[5] Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K. and Rosenburg, J. (2002), Preliminary guidelines for Empirical research in software Engineering, IEEE Trans. Softw. Eng., 28(8), 721-734

[6] 'Capability Maturity Model for Software', Technical report CMW/SEI-91-TR-24, Carnegie-Mellon University, 1991.