Introduction

This document seeks to provide a small yet useful set of metrics that may easily be implemented within an organisation transitioning to Agile.

As Agile has moved from the *innovators* through *early adopters* and now (possibly) into the *early majority* of adopters, the demand for more formal measurement has increased. Some Agile practitioners suggest that all metrics are bad. We believe it is untrue and unhelpful.

There are certainly additional Agile metrics that can be put in place. In particular, we have not addressed enterprise level metrics (for portfolio management). We have also not dealt with a number of secondary metrics that teams might want to employ on a temporary basis to make specific issues visible in their journey to becoming Agile.

Metrics and Myths

The problem with measurement, or more accurately how it is used or perceived to be used, is that it drives behaviour. Research has shown that many traditional metrics drive dysfunctional behaviour that is aimed at optimising ('gaming') the metric, but may be sub-optimal for the organisation. The sad fact is that it is often management's misuse of the metric that drives this behaviour.

The challenge therefore is to establish Agile metrics that do a good job of avoiding gaming and other dysfunctional behaviour.

So we may divide all possible measurements into 'metrics', which we define as those that yield helpful results and 'myths', which we define as those that drive dysfunctional or sub-optimal behaviour.

We also think it is helpful to consider metrics in four categories:

- Productivity
- Quality
- Predictability
- Value

Evidence drawn from field work suggests that the usefulness increases from the first to the last of these categories. For example at least one Agile expert has suggested that productivity is impossible to measure and that attempts to do so result in sub-optimal behaviour changes by those being measured. By contrast, predictability metrics can yield helpful results without much risk of gaming.

Good Agile Metrics

We have endeavoured to choose metrics that live up to the following standards. A good Agile metric:

1. Reinforces Agile principles (reinforces desired behaviour)
2. Measures results not output (e.g. work *not* done)
3. Measures trends (measures up)
4. Belongs to a small set of metrics (just enough)
5. Is easy to collect (duh!)
6. Reveals its context (no obfuscation)
7. Fuels meaningful conversation (people talk about what they *learned*)

8. Provides frequent and regular feedback (amplifies learning)
9. May measure *value* or *process* (both may be needed)
10. Encourages good enough quality (from a business perspective)

The One Metric

The Agile Manifesto tells us that: “our highest priority is to satisfy the customer through early and continuous delivery of valuable software”.

Both *Lean* and Scrum tell us that the primary measure is return on investment (ROI). It offers the advantages of being a high-level measure that cannot be gamed by sub-optimal team behaviour.

Unfortunately most organisations are not initially in a position to measure ROI. The cost side is normally easy, but measuring value that results from delivery of software features often is not.

It is one of key challenges of the Product Owner role to find ways to attribute value to product releases and features in order to choose which are the most valuable features to implement.

We need to follow this up with measurement of this value in order to establish ROI as our key metric.

Metrics We Can Implement Now

Happily there are metrics that we can establish quite early in an Agile implementation with relative ease. In this section we set what we believe to be a small yet sufficient number of metrics that have these characteristics.

We do this with the caveat that most (all?) metrics will drive behaviour. The extent to which this happens will be influenced by how they are used by management, or perceived to be used.

Customer Satisfaction Survey [value]

A customer satisfaction survey contains a number of questions aimed at assessing the customers’ view of the team and the work the team is delivering. While the questions are mostly qualitative and individual answers subjective, surveys taken regularly and across a variety of participants will yield useful trends.

It is desirable to conduct such surveys before or at the start of the Agile implementation to establish a baseline against which to assess progress.

Recommended frequency is with or after each release of the product. If releases are infrequent then perhaps each sprint, but too often will become annoying for the participants. Around every 6 weeks to 3 months feels about right.

Another question is who should participate. Our feeling is it should include all stakeholders who have regular interaction with the Scrum Team and those who are impacted by what the team delivers. (Hopefully those who are impacted are also those who interact, but who knows?)

Team Satisfaction Survey [value]

Similar to the customer satisfaction survey, but obviously directed towards the Scrum Team members. Remember that the Scrum Team comprises the Product Owner, (development) team members and the ScrumMaster.

Again, conduct the first survey as Agile implementation starts.

The ideal place to conduct these is during or linked to the Sprint Retrospectives. Judgement will need to be used whether to do this every Sprint if the Sprint length is short. It feels right to do these at least monthly, as they are less invasive than customer surveys.

Business Value Burnup Chart [value]

Part of the process of 'grooming' the Product Backlog is assigning a measure of business value to each backlog item. Mike Cohn describes some techniques for doing this in his book "Agile Estimating and Planning". A difficulty exists in assigning value to individual, fine-grained stories. To overcome this Cohn advocates grouping stories in 'themes' for the purpose of assigning value.

Until the Product Owner is able to assign real ($) values to stories or themes, a relative measure can be used such as 'value points' or 'utils'.

The Burnup chart is then simply a measure of business value delivered over time (Sprints). If the value can be measured in monetary terms, then a proper ROI calculation can also be done.

Velocity Burndown Chart [predictability]

The Burndown Chart reflecting team velocity is one of the standard artefacts mandated in the Scrum Framework. This chart reflects the rate at which the team can turn Product Backlog items into potentially shippable product increments (or "Running Tested Features" as Ron Jeffries like to call them).

It is important for management to understand that the Burndown Chart is a predictability metric and its purpose is *not* to drive output (productivity). Its primary purpose is to inform management using empirical evidence how many features are likely to be delivered within a given time period. This aids management to do better release planning.

The visibility offered by the Burndown Chart is intended to counter the historical obfuscation of the truth by project managers driven to this dysfunctional behaviour by their managers in the mistaken belief that this will lead to higher productivity. In practice, such behaviour only leads to a diminution of quality, which leads in turn to technical debt and product brittleness. Over time the cost of this is far higher than the (possible) short-term gain in output from 'driving' the project manager and team.

The standard Burndown Chart is fully described in the standard Scrum literature. We prefer the 'alternative' Burndown Chart as described by Mike Cohn. This provides additional visibility into changes in scope over time and adds fidelity to release planning. We prefer sizing of Backlog items in 'story points'.

The Burndown Chart must be updated by the Product Owner at the end of every Sprint. It is normal (and desirable) to do this during the Sprint Review. While tools may be used, we recommend a big, visible, hand-drawn chart, possibly fed from an Excel spreadsheet!

Running Automated Tests [quality]

Within limits, the more running (i.e. passing) automated tests a team has in place is a positive measure of quality. Beyond a certain level, this will cease to be true, but we have not yet met a team that has reached this point. (We hope to!)

The metric simply tracks the cumulative number of passing tests against time. Obviously a higher value is better.

Anecdotally, this was one of the prime metrics that salesforce.com put in place during its big-bang transition to Agile.

Defect Count [quality]

In organisations that use formal tools for tracking defects (or incidents), we recommend two metrics to track quality improvement:

- Post-Sprint Defect Arrival (*leading indicator*)
- Post-Release Defect Arrival (*lagging indicator*)
- Defect Resolution (root causes fixed)

These numbers are plotted against time (Sprints). The trending of these curves independently and relative to one another can tell us a great deal about the effect of the team's attempts to improve quality *ab initio* and about it's ability to drive down the open defect count.

It may be worth noting the prescription for dealing with a large initial defect list (database). If the count of open defects exceeds 100, simply cull (delete!) all defects logged more that 6 months ago. If the remaining number still exceeds 100, delete those whose severity (impact) is not 'critical' or 'significant'. The wasted effort in managing large backlogs of defects vastly exceeds the value in retaining them. Any deleted defect that was important will re-surface and be logged again soon enough!

Technical Debt [quality]

Technical debt is 'undone' work. In other words work that will have to be done in the future in order to bring the code base to the required quality level. It might include refactoring of code to simplify the design or remove 'dirty' code. It generally occurs during periods where the team is driven to produce output at a faster pace than it can actually sustain.

Technical debt is always added to the Product Backlog and is prioritised by the Product Owner and team in relation to all the other work. The idea is to specifically track the quantum of technical debt and whether this is growing or shrinking. This is a good metric for the CIO (or equivalent) to track.

Items to track:

- Technical debt added
- Technical debt removed

The units are story points (as for other Backlog items) and these are tracked against time (Sprints).

Work-In-Process [productivity]

Items (stories) in-progress is a productivity metric. It seeks to help a team track whether they are working collaboratively or not. The idea in an Agile team is for the whole team, as far as is reasonably possible, to collaborate on a single work item until it is 'done'. This increases the rate of output, quality and cross-learning. It decreases the risk of unfinished items at the end of the Sprint, which results in waste.

Simply by tracking on a daily basis how many items the team has in-progress will make visible the extent to which they are collaborating. The chart tracks stories in-progress against days. It is agnostic of Sprint boundaries. It should trend towards 1 over time. Any value higher than 2 is cause for action by the ScrumMaster.

Story Cycle Time [productivity]

This measures how long each story is in-progress. This metric therefore measures much the same as the previous one. A team working on several stories in parallel will display longer cycle times. A team with fewer, larger stories will also show longer cycle times.

Cycle time per story is tracked against time (Sprints). A lower value is better, with around 2 days being a good result.