Assessing Agility

Dr Peter Lappo, Henry C.T. Andrew

Systematic Methods Research Ltd
Greylands House, Mallory Rd, Hove, BN3 6TD, UK
peter.lappo@smr.co.uk,
Box River Ltd
Greylands House, Mallory Rd, Hove, BN3 6TD, UK
henry_andrew@yahoo.com

Abstract. A technique is described that uses goals to assess the agility of software development teams and shows how it can be used with some examples. The agile assessment can be used to make investment decisions and process alterations. Value stream mapping is seen as an important technique in analysing processes.

Keywords. Agile Goals, Assessment Technique, Value Stream Mapping, Net Benefit, Measuring Agility

1 Introduction

There seems to be a general feeling in the agile community that if you follow all the practices associated with your chosen method then you are by definition agile. While this may be true of agile methods such as XP [1] or Scrum [2] which have defined a set of practices that have emergent properties such that the team becomes agile as a result of the process. It is still possible to use XP or Scrum without gaining much in terms of agility.

There is much talk in the agile community of improving the software development process but most of the improvements are anecdotal. There have been attempts at measuring and proving the efficacy of agile software development methods versus traditional methods [3, 4]. These studies have shown that agile methods are at least as good as traditional methods. There is even some talk of metrics [5, 6] which some people unfortunately frown upon [7] because of the political connotations, but nevertheless metrics don't measure how agile you are. Williams et al [5] proposed an interesting set of agile metrics, but the metrics defined where not formalised.

This paper defines a technique to assess agility through goals and using some examples shows how to create agile goals. The technique won't compare you with other people, at least not directly, rather it is a means to measure your relative performance.

1.1 Comparative Studies Between Traditional And Agile Methods

Comparative studies [3, 4] between traditional and agile methods are very difficult to do on a small scale because it is very hard to devise a controlled experiment that anyone trained in the theory of science can qualify as a valid scientific experiment. The problem is small scale studies are not repeatable primarily due to the human element.

The repeatability problem only disappears when large numbers of projects are compared and statistical techniques are used to correlate process features with outcomes. However, at this moment in time there is not enough data available to prove that agile methods are better than traditional methods. Although anecdotal evidence suggests that agile methods are more effective at delivering working solutions.

1.2 Relative Comparisons

The only practical way of determining whether agile methods actually make a difference to your software development process is by measuring your own process and seeing whether agile methods actually make a difference. The question is how to make the measurements and what to measure. Once these questions have been answered it is possible to have some clarity about what makes a difference in your environment.

While it is possible to use the technique proposed in this paper to compare yourself with other teams it is difficult as often you are not comparing like with like. However, other peoples performances are useful as a guide to what can be achieved.

1.3 Metrics Versus Measurable Goals

There is a large body of work concerning software metrics. Most of this work is useful for the long term analysis of trends and comparative studies. Metrics are sometimes used during planning and bidding. However, metrics are not much use for assessing agility. This paper won't concern itself much with metrics as it is the belief of the author that most metrics are measuring artifacts of the process such as lines of code or code complexity.

Most environments the author has worked in collect some sort of metrics even if it is only hours spent on project tasks, requirements tested or defect rates. While these may be useful, especially the last two, the collection of hours on tasks is often a fantasy of the developer or manipulated by political necessities.

Rather than just gather metrics such as lines of code, code complexity, function point or quality metrics we need measurements that are related to business needs with an agile perspective. For instance, while code complexity analysis tells you how complex your code, it does not give you any idea whether this code is easy to change in practice and hence having the potential to be agile. You may also have code that is not complex, but you still may have difficulty being agile because of your process or because of the attitude or experience of the people working on the project. The point

being that low level measurements of process artifacts don't necessarily mean anything at a higher process level.

Using the measurable goals described below it is possible to define a set of goals for your team that are directly related to agile principles such as frequent delivery of software. Goals differ from metrics principally in that they attempt to be free from the details of the process, so that a goal to be responsive to change, for example, doesn't care about metrics like lines of code or code complexity. The other reason to differentiate goals from metrics is because a goal implies thought about where you want to be rather than where you are now.

2 Technique for Defining and Achieving Goals

2.1 Goal Setting and Implementation

We simply define a set of measurable goals (see 2.3 below) for the process, environment, tools and software quality in conjunction with the project stakeholders (this includes developers). Then we determine what the current state of these goals are, agree a future value with the project stakeholders, and takes steps to achieve the agreed values.

The process of achieving the goals should of course be iterative, with regular reviews on progress and the goals themselves. The cost and benefit of change should also be considered thereby preventing over or under investment. You may of couse find that you are sufficiently agile using your current process.

The goals, ideally, should be method agnostic, that is, we shouldn't define goals in terms of particular practices used to achieve agility or in terms of the artifacts of the process as this will stop method innovation and cause a lot of argument about favourite best practices. But it should be possible to assess a particular practice in terms of the impact it has on agility. This however, this is beyond the scope of this paper.

2.2 Techniques for Achieving Goals

Numerous management techniques exist for improving processes, but perhaps the most interesting one to use at an early stage when you are investigating possible improvements is value stream mapping [8] as used by the lean community.

Value stream mapping produces a timeline for a complete process and determines those steps which add value to the process. Subsequent work entails eliminating steps that don't add value and eliminating process delays.

For example, the production release process is always an interesting process to examine. It may have a number steps that cause unreasonable delays which could easily be eliminated or automated.

2.3 Categorisation of Goals

It is useful to categorise goals to help define them and focus the mind on what goals are necessary. This paper proposes four categories as follows.

Process
Goals associated with the software development process and the process practices used.

Environment
Goals associated with the environment the process runs in. These are mainly organisational and people oriented goals.

Tools
Goals associated with the tools used to develop the software.

Software
These are goals associated with the design and quality of the software. How the software has been designed can have a big impact on agility and of course if the software is full of bugs or only manual testing is performed then again agility will be constrained.

2.4 Goal Definition

Goals are defined using the technique described by Gilb [9] and are described by the following attributes.

- Name
- Test
- Benchmark
- Now
- Worst
- Planned
- Net Benefit
- Planned Date
- Owner
- Notes

These are detailed as follows.

Name
This a short name for the goal to make it easy to remember and discuss. It is also used for cross referencing to other goals. For example, “Rate Of Change”. Names are preferred over numbers as they are easier to remember and have more meaning.

However, some people prefer to use numbers. We don't care what convention you use as long as it works.

Test
The goal needs to be measured in some way. This defines the test to measure the goal and its scale. The test is the most important parts of defining a goal.

Tests should be quantitative when possible, but it is appreciated that some things are difficult to measure, such as knowledge transfer, so qualitative assessments can be used.

For example, “Rate Of Change” could be measured by running a query on your change management system to determine how many changes have been released to production over a given period. The scale could be changes per month.

Benchmark
This is an actual measurement taken in the field. It could be data from within your own organisation but is more likely to be a measure taken from the best organisation in your line of business. In other words, it is the benchmark to compare yourselves against. This field is optional as the data may not be available or you have taken the “lean” approach [10] which is to strive for perfection and ignore benchmarks. However, you may find some data which is relevant to your situation.

Now
Now simply states what the current measurement is. For example, for the “Rate Of Change” goal could be 1 change released per month. This field is optional if data is unavailable. However, we don't recommend this because you won't know if you are making progress, so a rough guess is better than no data at all.

Worst
It is recognised that some goals may be difficult to achieve so this defines the lowest expected improvement in the goal. For example, the “Rate Of Change” goal could have a worst case improvement of 2 changes per month.

Planned
This is the planned level of the goal. For example, the “Rate Of Change” goal could have a planned value of 20 changes per month.

Planned Date
The planned date defines when you expect to achieve your planned or worst case goal.

Net Benefit
We'd rather you didn't implement any change to your organisation unless you have some idea of the net benefit of the goal. Where net benefit is the potential value of the goal minus its implementation cost.

Value is a difficult thing to define and measure and even more difficult to predict. It also dangerous as you may oversell the benefit of a goal and raise expectations too high. Some goals may have intangible values. In this case simply list the benefits and costs.

The cost of course is only an estimate as it is difficult to predict what the costs will be as you may incur unexpected costs. For instance, your new environment may not be suitable for some people and they may leave, forcing you to replace them and train their replacements.

You may find that some goals don't add much value or the cost of achieving the goal is prohibitive in which case the goal should be dropped. The net benefit serves as a means of checking whether its worth implementing this goal.

While this attribute is optional we recommend you attempt to quantify the benefits to your organisation. If nothing else it will help you justify what you are trying to achieve. One surprising result may be that the goal may cost virtually nothing. For example, implementing a daily build may simply require an entry into a Unix cron table, which, assuming you have you have the correct environment, may only take 10 minutes to implement.

For example, the “Rate Of Change” goal may bring the following intangible benefit: Ability to implement changes that previously had to be ignored because they weren't of a sufficiently high priority.

Owner

All goals must be owned by someone or if you prefer sponsored by someone, preferably this person should be in the management team or it could a steering committee. The owner is responsible for ensuring the goal is achieved but not necessarily implementing the goal, as this may be carried out by someone else.

Notes

This is simply further notes of explanation which can include a reference to further information that may be relevant. It is optional.

3 Agile Goals

This paper hasn't the space available to define a set of goals for a team as goals are dependent on business objectives and available investment so we shall just present a few sample goals to give you an idea of how to define them for yourselves.

3.1 Example Goals

If you read the agile manifesto one of its principles is to value working software over artifacts such as documentation. This leads us directly into the most obvious goals for your agile team, the **Frequent Delivery Of Working Software**. Of course your current process may be incapable of delivering this so you may set yourself another

goal which could be **Quick Releases**, i.e. the ability to integrate, build and release your software in a timely manner.

On the environmental front you know key application knowledge is in the heads of a two or three individuals which is preventing the rest of the team from being productive, so you define a goal to **Share Knowledge**. How you do this is irrelevant to the goal, but some means of measuring is not be.

The tools you use are of course perfect and you are perfectly happy with vi (or so you think), so you don't define any tool goals. On the other hand you don't actually get your hands dirty with code, but when you hold a review with the project stakeholders your developers come up with their own goals, namely **Refactoring Support** as they know they are steadily creating an unmaintainable mess.

When it comes to the actual software you do know you have trouble, but you are convinced the QA testers are a bunch of slackers. Surely they didn't mean four weeks to regression test the system for such as small change? **Automated Acceptance Testing** seems the only way forward.

3.2 Formalised Goals

Rather than attempt to squeeze all the goals and their attributes on a single table we'll just look at a couple.

Name	**Frequent Delivery Of Working Software**
Test	Record the date when software is released to pre-production on a graph and measure the number of working days between each release measured in frequency in days.
Benchmark	10
Now	90
Worst	20
Planned	10
Planned Date	June 2004
Net Benefit	Each piece of automation will reduce the manual effort in processing the invoices and provide valuable feedback on how the users are adapting to the software and whether the software is meeting business needs. Reducing the delivery cycle to 10 days will require a large investment in test automation amongst other things. The expected net benefit is difficult to determine as it depends on the value of the changes being introduced at each cycle. However, if the system goes into operation earlier the company will start getting a return on its investment sooner.
Owner	Director of IT
Notes	None.

Name	**Share Knowledge**
Test	Amount of time spent pair programming in minutes per hour pre day.
Benchmark	unknown

Now	0
Worst	30
Planned	50
Planned Date	June 2004
Net Benefit	The team should be more productive as a whole as less experienced members won't have to waste time finding out things for themselves. Some reduction in key personnel productivity is expected. The actual net benefit is difficult to quantify.
Owner	Team Leader
Notes	This is difficult goal to measure and it is possible it should be divided into sub-goals.

4 Conclusion

With a little thought it is possible to define a number of measurable goals which will help you achieve greater agility, where of course agility is defined by your goals! Any number of management techniques can be used to achieve your goals with value stream mapping being particularly useful during analysis.

No longer will you have sleepless nights worrying whether you are doing all the recommended XP practices in order to be agile (whatever they are at the time). If your agile goals satisfy the project stakeholders then you are agile. You can of course look around and see what kind of agility scores your competitors are achieving and attempt to better them or you could take the lean approach [10] and simply aim to be the best.

The point is by measuring what you are doing and setting goals for the future you have an opportunity to achieve those goals. Without objective measurements you are in the same state as early philosophers that conjectured about our universe. You are guessing.

References

1 Kent Beck . Extreme Programming Explained – Embrace Change, Addison-Wesley, (1999)
2 Ken Schwaber. The Scrum Development Process (OOPSLA'95 Workshop on Business Object Design and Implementation (1995)
3 John Noll and Darren C. Atkinson. Comparing Extreme Programming to Traditional Development for Student Projects: A Case Study. In Proceedings of the 4th International Conference of Extreme Programming and Agile Processes in Software Engineering, May 2003.
4 Francisco Macias, Mike Holcombe, Marian Gheorghe. A Formal Experiment Comparing Extreme Programming with Traditional Software Construction. In Proceedings of the Fourth Mexican International Conference on Computer Science September (2003)
5 L. Williams, G. Succi, M. Stefanovic, M. Marchesi. A Metric Suite for Evaluating the Effectiveness of an Agile Methodology. In Extreme Programming Perspectives. Addison Wesley (2003)
6 William Krebs, Laurie Williams, Lucas Layman. IBM / NC State University XP Study Metrics. Workshop submission to XP Agile Universe, http://sern.ucalgary.ca/eeap/wp/bk-position-2003.html (2003)

7 Tim Bacon, Steering With Numbers. XDay http://xpday3.xpday.org/slides/SteeringWithNumbers.pdf (2003)
8 Mike Rother and John Shook, Learning to See. Lean Enterprise Institute (1998)
9 Tom Gilb. Principles of Software Engineering Management. P133-158, Addison-Wesley, (1988)
10 James Womack and Daniel Jones. Lean Thinking: Banish Waste and Create Wealth in Your Corporation, Revised and Updated, Free Press (2003)