GJCRT : Global Journal of Computer Science Research & Technology
ISSN (online): 2349- 0608

METRICS OF VARIOUS AGILE METHODOLOGIES

Ms. Jagpuneet Kaur Bajwa[1]*,Ifra*[2]
Department of Computer Science, Punjabi University, Patiala

Abstract-- Scrum Teams use lightweight tools like Story Points, the Burn down chart, and Team Velocity. While essential, these tools alone provide insufficient information to maintain a high energy state that
yields Hyper-productivity.

We describe the scrum metrics that can develop and sustain the Teams— Sprint burndown, Enhanced release burndown, Sprint interference, Remedial focus. We can also use agile management tools to reduce the problem of communication and for all the updates in the product backlog as well as Sprint backlog list.

***Keywords*—**Agile software development process, Software metrics, Testing, Scrum, Scrum metrics.

I. INTRODUCTION

Software metrics can be used to gain a wide variety of information about the quality of the product delivered to the customer, progress of a software project, cost estimation, and size/complexity of a software system. Measurements need to be closely monitored when the requirements of a software system change frequently. Changing requirements are one of the major problems arising in the software development process. Agile Software Development (ASD) process successfully handles the reality of change. Therefore, while selecting software metrics to measure ASD-based projects, it should be handled with a deeper understanding. The need of agile software development is growing rapidly as it allow to the change the requirements though development cycle and provide quick delivery of software to the customer and customer is a part of team. As testing is an very important part, without testing organization cant able to judge whether the software is properly working according to the requirements of customer and also help to find out the bugs in the coding. In Agile there are multiple releases so organizations use automation testing to cut the testing time and quick delivery of project.

Given that, adaptation into the ASD process is rapidly increasing, it is imperative to identify a set of metrics that is more suitable for the ASD process.

"*What are the important software metrics and their usage in projects based on the ASD process?*"

To identify a suitable set of metrics for the ASD process. We identified ten metrics that are suitable for the ASD process. Among those the top five metrics include **Delivery on Time,**

Work Capacity, Unit Test Coverage for the developed code, percentage of Adopted Work, and Bug Correction Time from new-to-closed state. Thumbs-up Rule, Noncompliance Index, and Top Hill View are three new metrics identified through the study. Thumbs-up Rule can be used to measure customer satisfaction at the end of each sprint. Non-compliance Index is used to check a project's compliance as per their company standards. Top Hill View is used to track the project progress.

SCRUM

SCRUM is a method to manage or develop a project using agile methodology. It is an iterative process and delivers a small part of the software at the end of iteration. The basic idea of SCRUM is to make system flexible enough to adapt the changes in requirements, resources; technology etc. to achieve the desired results. SCRUM includes many activities to develop software or an application. It is helpful for management and development team to handle the project in a better way. It builds a Rapid prototype. As the initial requirements gathered by the customer may be incomplete and can be changed during development process. The team took the overview of the software application.

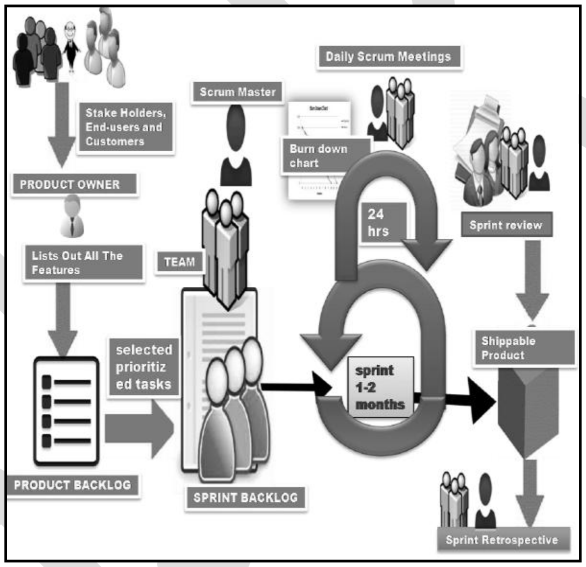

Fig 1.SCRUM Process [5]

After team has gathered the requirements, then planning phase starts and then simple design is built for the software. Planning and designing is finalized in a short meeting. After planning and designing the whole project is divided into small iterations, known as Sprint and comes into development phase. Each Sprint has a Sprint backlog which contains the goals to be achieved during that Sprint and accordingly team members distribute their work among themselves and review it at the end .At the end of every Sprint there is a meeting in which team member discuss what they have done and what needed to be done as shown in figure 2.

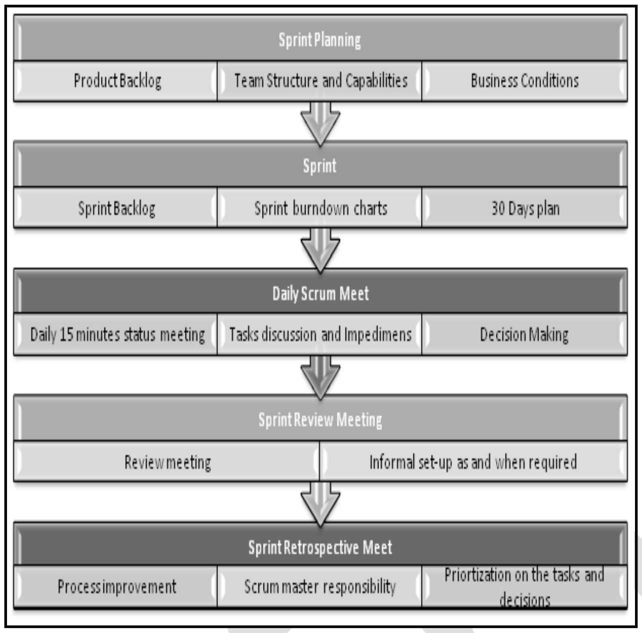
Sprint Planning
Product Backlog
Team Structure and Capabilities
Business Conditions
Sprint
Sprint Backlog
Sprint burndown charts
30 Days plan
Daily Scrum Meet
Daily 15 minutes status meeting
Tasks discussion and Impedimens
Decision Making
Sprint Review Meeting
Review meeting
Informal set-up as and when required
Sprint Retrospective Meet
Process improvement
Scrum master responsibility
Priortization on the tasks and decisions

Fig 2. Sprint Process [5]

Customer tests the part of the software and if there are any bugs/ errors, or they need any changes, will be added in product backlog list and be implemented in next Sprint.

II. RELATED WORK

Oza and Korkala [2] classified the metrics used in the ASD process as **Code level, Productivity/effort level, and Economic metrics.** Code-level metrics try to provide visibility into the code quality. Productivity and Economic metrics support the decision making process.
Downey and Sutherland [1] identified ten essential metrics which are meaningful and can be used for managerial decision making. Those metrics include **Velocity, Work Capacity, Focus Factor, Percentage of Adopted Work, Percentage of Found Work, Accuracy of Forecast, Targeted Value Increase (TVI+), Success at Scale, and Win/Loss Record.**

Manila [3] came up with a set of customized metrics by analyzing a selected organization. These metrics include **Fault Correction Time to Closed state, Delivery on Time, Technical Debt, Unit Test Coverage for the developed code, Smoke Test Cycle Time, and Regression Test Cycle Time.**

Gustafsson [4] classified ASD metrics into five categories as **Quality, Predictability, Business value, Lean, and Cost**. He
further described three metrics each under Quality and Lean, namely Defect Count, Technical Debt, Faults Slip Though,
Lead Time, Work in Progress, and Queues

III. RESEARCH METHODOLOGY

Scrum Metrics

I look at grouping metrics into two main categories:

- Metrics for Scrum projects (the focus of this article)
- Metrics for broader Scrum rollouts (the focus for another article)

Four meaningful metrics

The following sections will step you through a selection of four project related metrics that I find particularly helpful including:

- Sprint burndown
- Enhanced release burndown
- Sprint interference
- Remedial focus

1.Sprint Burndown

The sprint burndown is a forecasting metric to assist in tracking progress throughout the current sprint.

How is it generated?

1. For each day in a sprint, plot the sum of the ‘remaining times’ for all tasks in the sprint backlog.
2. Draw a connecting line between the current day’s total and the previous day’s

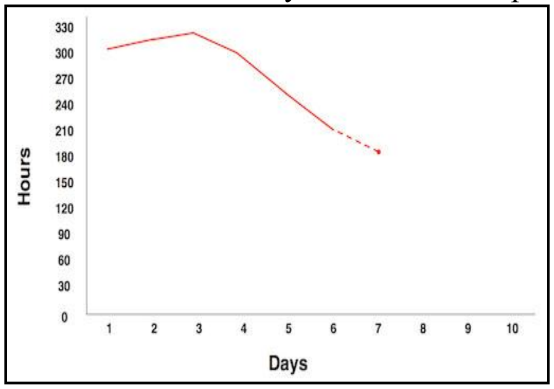

Figure 1 – A sprint burndown chart, updated on a daily basis. [9]

When it is generated?

The sprint burndown is generated at the end of each day of a sprint (excluding the final day, which is dedicated to the sprint review, retrospective, and planning for the subsequent sprint).

What is it telling you?

The sprint burndown metric acts as a daily gauge for the Scrum team to help manage its workflow and gauge progress.

If the chart is trending behind schedule, it could be reflecting the fact that:

- New tasks were added to the sprint backlog (that weren’t anticipated during sprint planning).
- Some of the task estimations were incorrect.
- Team members had taken some unplanned time off.

- Impediments had hampered progress.

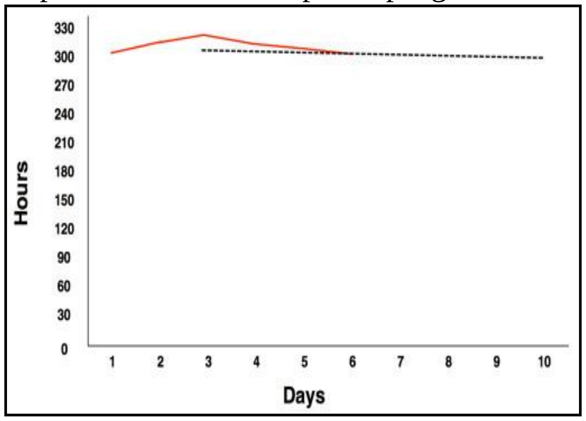

Figure 2 – The team is clearly behind schedule; time to speak to the product owner about possibly decreasing scope. [9]

Of course, it is possible that all four factors had come into play causing the now expected delay.

Following sprint planning, many teams will draw a straight, diagonal (theoretical) line from the top of the y-axis values to end of the x-axis values and this is used as a benchmark for the actual burndown line. I warn against using this approach as it can easily create an inaccurate perception of progress. The problem with this line is that sprint progress will rarely mirror the theoretical line on a day-to-day basis. Many sprint burndown lines will actually burn up for the first few days due to new discoveries before beginning it's downward descent as the team gathers momentum. By including the theoretical line, a curious stakeholder may get the misleading impression that the team has fallen behind after only just one or two days.

How can you act on it?

If the sprint burndown clearly indicates that the team is not going to reach the sprint goal then apart from doing everything in your power to help remove any impediments, it should prompt a discussion with the product owner to assess whether any scope can be removed. If the slip is due to inaccurate estimating of tasks, it can be helpful to analyze why the estimates were wrong to try and improve the sprint planning accuracy for the next sprint.

The team is clearly behind schedule; time to speak to the product owner about possibly decreasing scope.

Sprint burndown charts can also paint a rosier picture (believe it or not) if they trend steeply towards an early completion of the sprint backlog. If this is the case, the burndown should prompt the product owner to prepare the next sprint-ready product backlog items for additional consumption prior to the forthcoming sprint planning session.

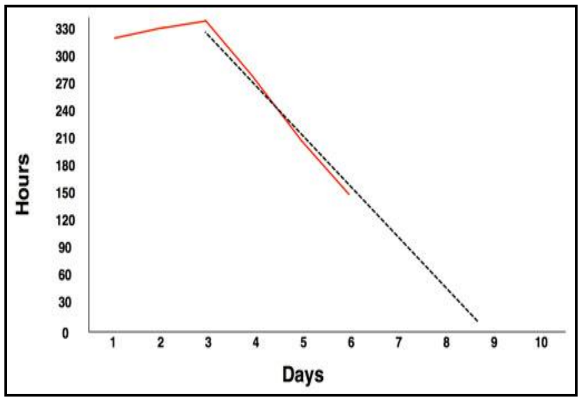

Figure 3 – The team is clearly ahead of schedule; time to speak to the product owner about adding scope. [9]

2.Enhanced Release Burndown

The enhanced release burndown is inspired by Mike Cohn’s ‘Alternative Scrum Release Burndown Chart’.

How is it generated?

1. For each sprint, plot the sum of the ‘remaining points’ for all product backlog items in the product backlog designated for the next release.
2. Draw a trend line relating to the data points in step 1.
3. For each sprint, plot (as negative y-axis values) the sum of the story points for any product backlog items added to the product backlog after the start of the project (if applicable).
4. Draw another trend line that relates to the data points in step 3.

When is this generated?
The enhanced release burndown metric is generated at the end of every sprint.

What is it telling you?
This metric signals what the development team’s rate of progress is relative to the scope’s rate of change. The point where the two trend lines (hopefully) intersect indicates roughly how many sprints will be required to complete the release. If the trend lines run parallel to each other (or diverge), it is an ominous indication that the release will theoretically never see the light of day.

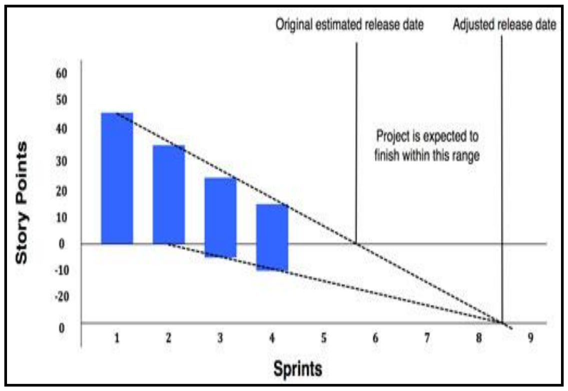

Figure 4 – An enhanced burndown chart, updated after each sprint. [9]

How can you act on it?
If the two trend lines do not intersect or the expected release duration is intolerable, then either the rate of progress needs to increase (by improving practices and / or removing impediments) or the scope needs to be reduced.

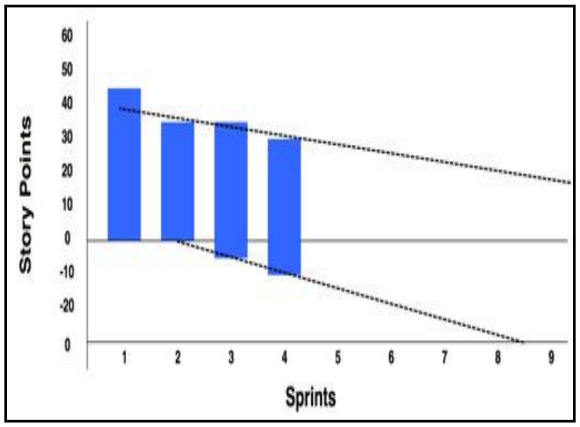

Figure 5 – Ominous signs that this release might never see the light of day. Better improve practices, remove impediments, or decrease scope. [9]

3. Sprint Interference

Sprint interference is productivity metric to assist teams with their sprint capacity planning.

How is it generated?

1. For each sprint, plot the sum of the time spent by any of the developers for any non-sprint backlog tasks.
2. Draw a trend line that relates to the data points in step 1.

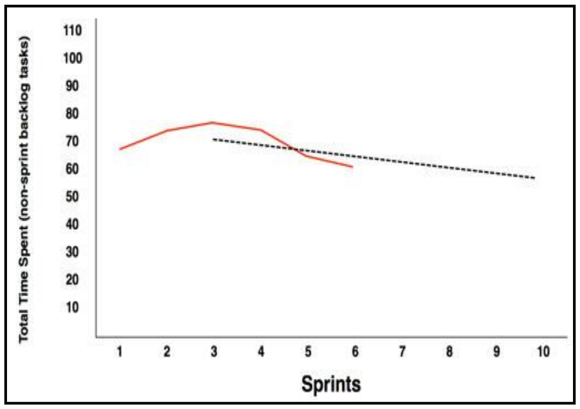

Figure 6 – Be sure to note the trend in this graph to assist in estimating your team's capacity at the next sprint planning session. [9]

Be sure to note the trend in this graph to assist in estimating your team's capacity at the next sprint planning session.

When is this generated?

The sprint interference metric is generated during sprint planning.

What is it telling you?

By providing visibility on the time spent handling historical sprint disruptions, this metric will help you to estimate the potential sprint capacity for the forthcoming sprint (the amount of time that the development team should allocate to sprint backlog tasks). This is especially helpful if you have adopted commitment-based sprint planning.

How can you act on it?

In any given sprint, there will be a range of external organizational disruptions that simply can't be avoided. This metric assists in quantifying these disruptions and can also indirectly help to identify what are unavoidable disruptions (such as company meetings) and what are avoidable impediments (such as constantly having to fix inadequate equipment).

4. Remedial Focus

How is it generated?

For each sprint, plot the total velocity (the sum of the points for all product backlog items including both new functionality and bugs.)

1. For each sprint, plot the sum of the points for all bug related work.
2. Draw a trend line that relates to the data points in step 2.

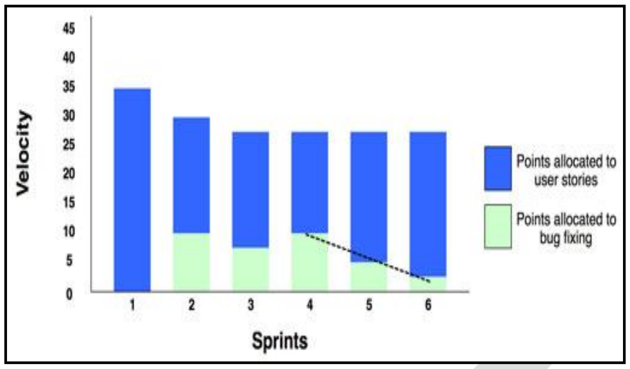

Figure 7 – This indicates that quality is improving. The effort spent on bug-fixing is going down while velocity remains consistent. [9]

When is this generated?

The remedial focus metric is generated at the end of each sprint.

What is it telling you?

This metric monitors the fluctuations in product quality by measuring the percentage of each sprint that is spent working on bugs as opposed to new functional requirements.

In addition, by quantifying the make up of the 'total velocity', additional insight can be garnered. For example, the total velocity may be trending upwards, which, on the surface would typically indicate positive improvement. However, if the amount of bug related work is also trending upwards this is an indication that the level of quality is slipping. As such, the increase in total velocity could in fact be indicating that the team is just getting faster at fixing its own bugs – somewhat of a back-handed compliment…

How can we act on it?

If the time spent on bugs isn't trending downwards, it is a clear indication that the level of inherent quality is insufficient. This should prompt the Scrum team to revisit the definition of done to tighten up the quality requirements.

Another Scrum Metrics

1. Actual Stories Completed vs. Committed Stories – the team's ability to understand and predict its capabilities. To measure, compare the number of stories committed to in sprint planning with the stories identified as completed in the sprint review.

2. Technical Debt Management – the known problems and issues delivered at the end of the sprint. It is usually measured by the number of bugs, but may also include deliverables such as training material, user documentation and delivery media.

3. Team Velocity – the consistency of the team's estimates from sprint to sprint. Calculate by comparing story points completed in the current sprint with points completed in the previous sprint; aim for +/- 10 percent.

4. Quality Delivered to Customers – Are we building the product the customer needs? Does every sprint provide value to customers and become a potentially releasable piece of the product? It's not necessarily a product ready to release but rather a work in progress, designed to solicit customer comments, opinions and suggestions. This can best be measured by surveying the customers and stakeholders.

5. Team Enthusiasm – a major component for a successful scrum team. If teammates aren't enthusiastic, no process or methodology will help. Measuring enthusiasm can be done by observing various sprint meetings or, the most straightforward approach, simply asking team members "Do you feel happy?" and "How motivated do you feel?"

6. Retrospective Process Improvement – the scrum team's ability to revise its development process to make it more effective and enjoyable for the next sprint. This can be measured using the count of retrospective items identified, the retrospective items the team committed to addressing and the items resolved by the end of the sprint.

7. Communication – how well the team, product owner, scrum master, customers and stakeholders are conducting open and honest communications. Through observing and listening you will get indications and clues about how well everyone is communicating.

8. Team's Adherence to Scrum Rules and Engineering Practices – Although scrum doesn't prescribe engineering practices—unlike XP—most companies define several of their own for their projects. You want to ensure that the scrum team follows the rules your company defines. This can be measured by counting the infractions that occur during each sprint.

9. Team's Understanding of Sprint Scope and Goal – a subjective measure of how well the customer, product team and development team understand and focus on the sprint stories and goal. The goal is usually aligned with the intended customer value to be delivered and is defined in the acceptance criteria of the stories. This is best determined through day-to-day contact.

IV. COMPARATIVE STUDY

Authors(s)	**Year**	**Paper Name**	**Results**
Scott Downey and Jeff Sutherland	2012	Scrum Metrics For Hyper-Productivity Teams:How They Fly Like Fighter Aircraft.	Provide subtle control that maintains safe and consistent growth.
Nilay Oza and Mikko Korkala	2012	Lessons Learned In Implementing Agile Software Development Metrics	Need of inducing a reference framework for metrics program.
Johan Gustafsson	2011	Model Of Agile Software Measurement:A Case Study	Conduct the types of performance and process optimization measurement.
K.V. Jeeva Padmini, H.M.N. Dilum Bandara and Indika Perera	2015	Use Of Software Metrics In Agile Software Development Process	Analyze the usage and benefits of software metrics.

V. FUTURE WORK

We analyze the usage and benefits of the scrum metrics and then propose a suitable set of metrics to be used within projects based on ASD process. We identified four recommended metrics to be used in ASD process which focuses on the product quality and team productivity. We are currently in the process of evaluating how the use of metrics correlates to the success or failure of Agile-based projects.

REFERENCES

[1] S. Downey and J. Sutherland, “Scrum Metrics for Hyperproductive Teams: How They fly like Fighter Aircraft,” in Proc. 46th Hawaii International Conference on System Sciences, Hawaii, 2013, pp. 4870-4878.
[2] N. Oza and M. Korkala, “Lessons Learned In Implementing Agile Software Development Metrics,” in Proc. UK Academy for Information System Conference, 2012.
[3] J. Mannila, “Key performance indicator Agile software development,” Satakunta University of Applies Sciences, 2013.
[4] J. Gustafsson, “Model of Agile software measurement: A case study,” MSc Thesis, Chalmers University of Technology, Sweden, 2011.
[5] Gaurav Raj, Komal Yadav and Arunima Jaiswal, “Emphasis on Testing Assimilation Using Cloud Computing for Improvised Agile Scrum Framework,” in Proc. 1st International conference on futuristic trend in computational analysis and knowledge management, 2015.

[6] K.V. Jeeva Padmini, H.M.N. Dilum Bandara and Indika Perea, “Use of Software Metrics in Agile Software Development Process,” University of Moratuwa, Sri Lanka, 2015.

[7] K.E. Emam, “A methodology for validating software product metrics,” Technical Report NRC 44142, National Research Council, Canada, 2002.

[8] M. Kunz, R.R. Dumke and N. Zenker, “Software metrics for Agile software development,” in Proc. 19th IEEE Australian Conference, 2008, pp.673-678.

[9] http://www.axisagile.com.au/blog/planning-and-metrics/scrum-metrics-and-reporting-measure-what-you-manage/.