## **Epics:** Epics represent large bodies of work that can be broken down into smaller, manageable pieces called Stories. They serve as high-level containers for related Stories and provide a comprehensive view of project progress and goals.

## 

## **Stories:** Also known as User Stories, they represent user-facing requirements and provide context for the development team. Stories articulate how a software feature will deliver value to the end-user and serve as the building blocks of Epics.

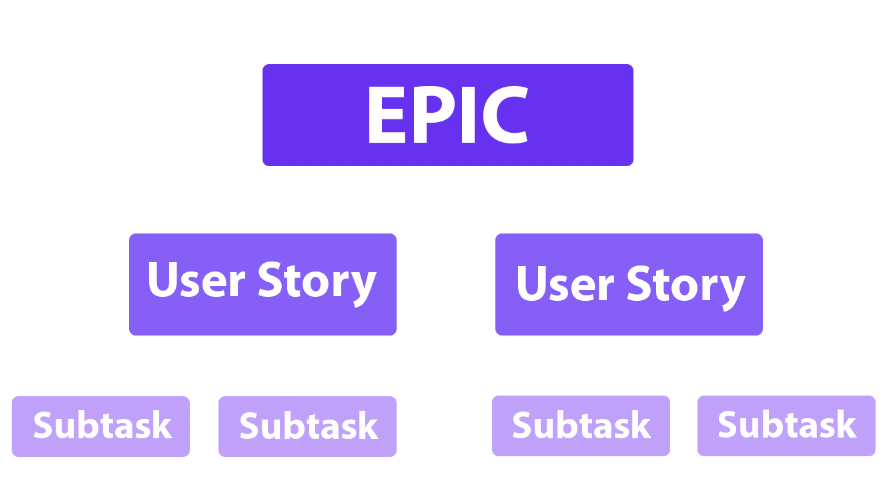
## 

## **Tasks:** Tasks are detailed units of work that must be completed to achieve the objectives outlined in the Stories. They represent the individual steps and actions required to implement a feature or complete a Story.

## **Epics**

"An Epic is a large body of work that can be broken down into a number of smaller stories, or sometimes called “Issues” in Jira. Epics often encompass multiple teams, on multiple projects, and can even be tracked on multiple boards."

In short, an Epic is a large User Story that can not be always delivered within a single Sprint.



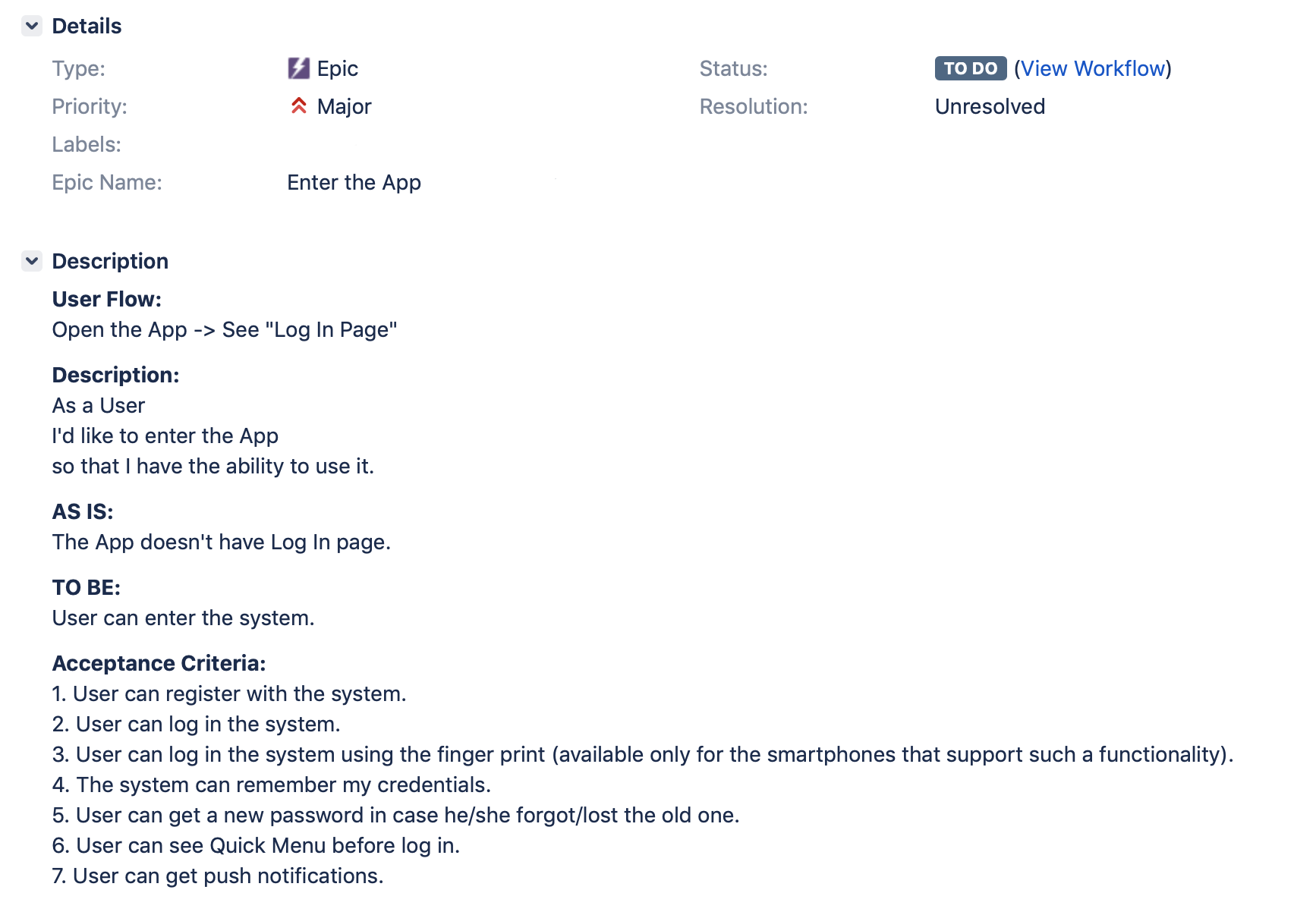
Example of the Epic:

"Bank's Users have the ability to log in with the System."

### **How to describe an Epic?**

It's really important to keep in mind that the team doesn't have enough time for reading the Project Manager's elaborate tasks.. Be accurate and short. It should be important and useful information inside only.

The purpose of an Epic is to articulate how a piece of work will deliver a particular value back to the customer. But as it's mentioned above Epic = big User Story. So we can describe it the same way.

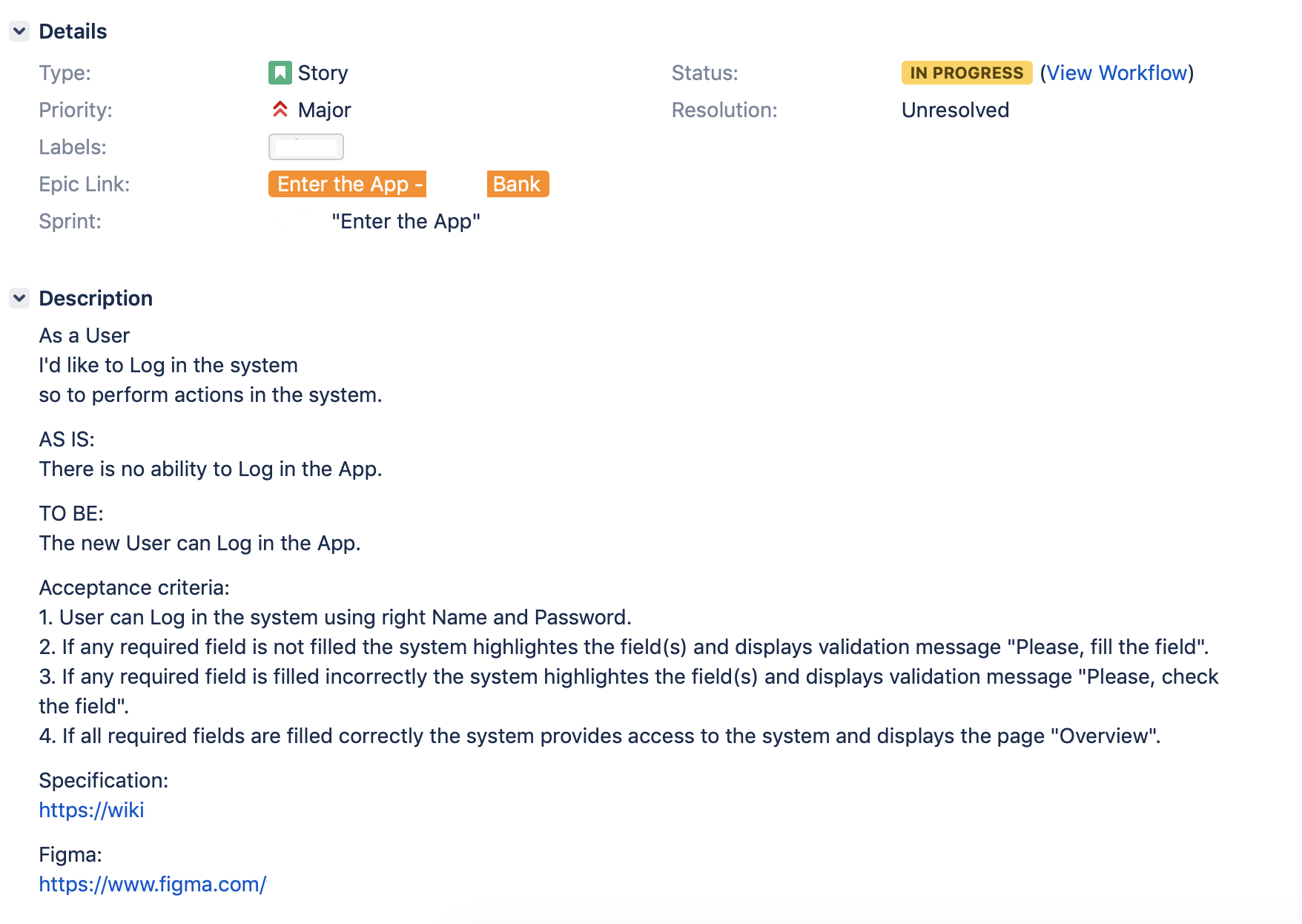


## **User Story.**

User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

### **How to describe a User Story in Jira?**

It is a good practice to have no more than 9-12 acceptance criteria in one User Story.

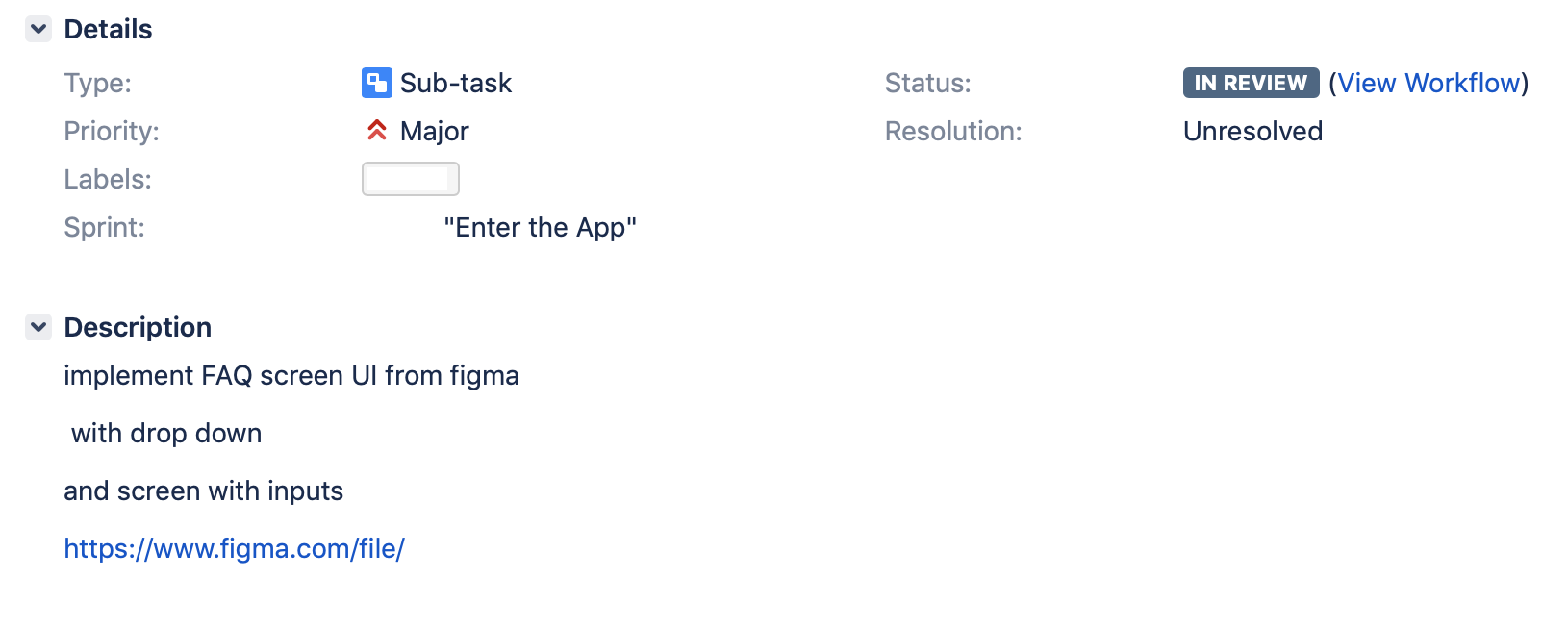


## **Subtasks.**

What concerns Subtasks - there are no specific requirements for creating them. The essential principle - to make subtasks SMART:

* S — (specific),
* M — (measurable),
* A — (achievable),
* R— (relevant),
* T — (time-boxed).

So, the description should have enough information for anybody who'd like to test the dev process.



To summarize all the information above I have to mention that your management principles should be easy to understand. Every time before starting a new Project I do my best to simplify the management process. And as a result I see grateful teams with high KPI levels.

KPI stands for Key Performance Indicator. KPIs for engineering teams can range from general performance metrics like velocity and burndown to more development-specific measurements like code coverage and time to market.

Each of these metrics provides a unique perspective into your team’s performance. If you combine them properly, they can give you detailed and actionable insights to help improve your project management skills and techniques.

**Agile Metrics:**

Agile metrics are standards that help a software team in monitoring how productive a team is across the different phases of the SDLC.

Agile metrics are an essential component of the development process. For companies or teams that work on the agile framework, agile metrics help in assessing software quality.

By measuring how productive a team is, agile metrics help keep the team performance in check. If there are any loopholes, they expose them at the initial stages. Since the data and its usage are measurable, it’s easier to work on the shortcomings with the help of these metrics. For example, velocity metrics can help you track your team’s output.

## **Importance of Agile Metrics**

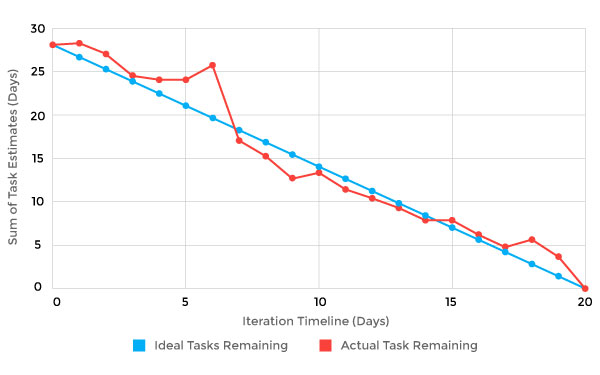
Now that we know what agile metrics are, let’s break down how they work. The entire concept of agile rests on continuous improvement (CI). But this isn’t something that you can impose on teams. It has to come from within. In short, self-improvement (SI) is a must. So, it’s safe to say that CI isn’t possible without SI.

Immediate delivery is an important component of agile. But one shouldn’t overlook SI in this case. Teams practicing SI give better results than those who don’t. But having a sustainable and effective SI is not child’s play. It’s a long-term process and needs a management framework. By tracking software quality and team performance, agile metrics support SI. In a way, these metrics directly impact CI.

Apart from improving continuity, delivering a high-quality product is also a vital part of agile. However, striking the balance between these two can be challenging. This gives rise to the need for metrics against which teams can measure progress. All in all, agile metrics help teams become self-managing. They also help firms in delivering value. At the same time, CI becomes a part of the workflow without much effort.

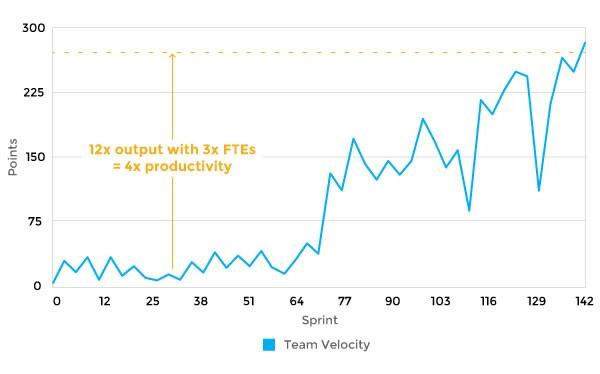
### **1. Sprint Burndown Report**

An agile framework comprises [scrum](https://www.plutora.com/blog/scrum-vs-agile) teams. They organize their processes into sprints. Since a sprint is time-bound, it’s important to track task progress frequently. A sprint burndown report is for tracking the completion of different tasks during a sprint. Time and work left to complete are the two main parameters of measurement in this case. The X-axis refers to the time. The Y-axis represents the work left. The unit of measurement is hours or story points. The team forecasts the workload at the beginning of a sprint. The target is to complete the workload by the end of the sprint.



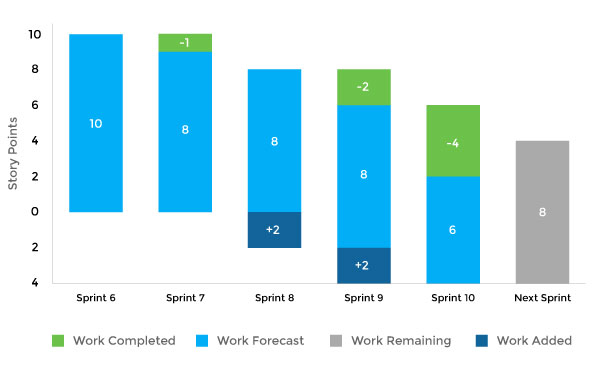
### **2. Velocity**

Velocity measures the average work a team does during a sprint. The report, in this case, contains several iterations. The accuracy of the forecast depends on the number of iterations. The more iterations, the more precise the forecast. The unit of measurement is hours or story points. Velocity also determines the ability of a team to work through backlogs. As time passes, velocity tends to evolve. To ensure consistent performance, it’s important to track velocity. If the velocity declines, it’s a sign that the team needs to fix something.



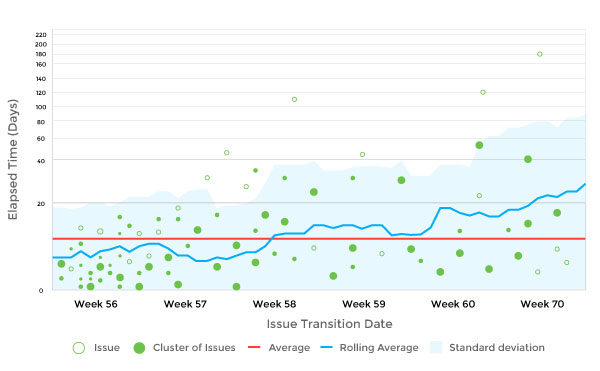
### **3. Epic and Release Burndown**

Unlike a sprint burndown, epic and release burndown focus on the bigger picture. They track progress over a large work body. There are many epics and versions of work in a sprint. So, it’s important to track their progress as well as each sprint. The entire team has to be aware of workflow in the epic and version. Epic and release burndown charts make that possible.



### **4. Control Chart**

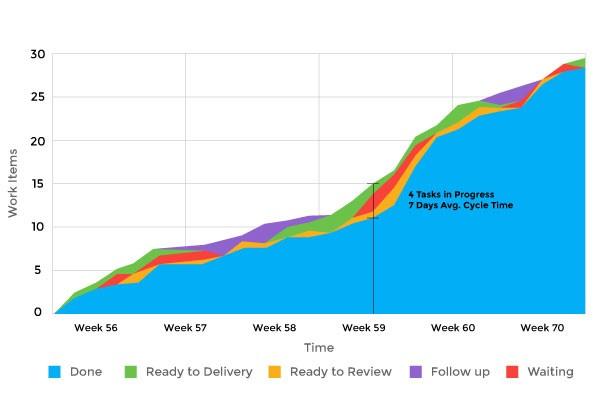
In agile, control charts focus on the time duration from the “in progress” to “complete” status of tasks. Their purpose is to check the cycle time of a single issue. Teams with consistency in cycle times have predictable deliveries. Besides this, teams with short cycle times have a high throughput. When teams measure cycle times, they improve the flexibility of their processes. For instance, in the case of changes, you can discern the results instantly. As a result, team members can make the necessary adjustments. In general, a short and consistent cycle time is the target to achieve in every sprint.



### **5. Cumulative Flow Diagram**

The cumulative flow diagram (CFD) ensures consistency in workflow across the team. The X-axis represents time. The number of issues is on the Y-axis. Ideally, the diagram should be smooth from left to right. Smoothen out the color bands in case of uneven flow. The band narrowing means throughput is higher than the rate of entry. If the band widens, this means that your workflow capacity is greater than required, and it can be moved elsewhere to smoothen the flow.

The CFD measures the state of the work in progress. With that, you can take measures to speed up the workflow. The diagram provides a clear visual representation of bottlenecks. You can analyze how bottlenecks formed in the first place. After that, the team can take steps to eliminate them and make improvements.



### **6. Lead Time**

Lead time is the period between the moment of making a request for delivering a product and the actual delivery. All the processes to bring a product to completion come under lead time. It also includes developing a business requirement and fixing bugs. Lead time is an important metric. The reason for this is it provides the exact time calculation for every process.

### **7. Value Delivered**

Here, project managers assign value to every requirement. This metric uses either dollars or a points system. Implementing features with high value should be the top priority. An upward trend in this metric shows that things are on track. On the other hand, a downward trend isn’t a good sign. It means the implementation of lower-value features is going on. If that’s the case, the team should make amends. Sometimes, you might even have to stop product development.

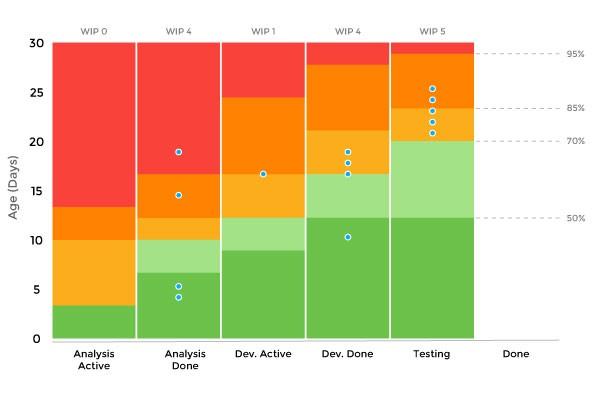


### **8. Net Promoter Score**

Net Promoter Score measures how much the customers are willing to recommend the product or service to others. It’s an index that ranges from -100 to 100. Customer loyalty is an important factor to determine the success of a firm. You can use the Net Promoter Score as a proxy for this purpose.

### **9. Work Item Age**

Work item age is the aging work in progress. This metric indicates the time that passes between the start and completion of the current task. The use of work item age is to detect the timeline for unfinished tasks. By using this metric, you’ll realize how your present tasks move forward. You can also compare your previous performance in the same context as the current scenario. The measurement tool, in this case, is the aging work in progress chart.



### **10. Throughput**

Throughput measures average tasks processed in each time unit. You can also call it a measure for story points per iteration. It represents a team’s productivity level. Throughput helps you understand the effect of workflow on business performance. You can get a better overview of the capacity of your team. However, it doesn’t show the starting point of tasks.

### **11. Blocked Time**

This metric assigns a blocker sticker to a task. It means that due to some reason, the assignee can’t proceed with a particular task because of some dependency. As soon as the dependency is fulfilled, you should move the blocked card to the right on the task board. Count the number and duration of blocked cards for measuring the number of blockers. Resolving the blockers will allow you to finish your “in progress” task quickly.

### **12. Escaped Defects**

When there are bugs in production, it causes a lot of unexpected damage. They pose problems, and the team needs to address them. Escaped defects metrics help in bug identification when a release enters production. You can assess the quality of the software in a raw form. For more information on automated defect minimization

**13. Failed Deployments**

Failed deployments is a useful quality metric. It helps in assessing the number of overall deployments. Moreover, teams can determine the reliability of the testing and production environment. This metric also determines whether a sprint is ready to enter production.

### **14. Code Coverage**

Code coverage measures the percentage of code unit tests cover. You can run this metric with every build. It represents the percentage of code coverage in raw form. This metric gives a decent perspective on progress. But it doesn’t cover other kinds of testing. Thus, high code coverage numbers don’t necessarily represent high quality.

### **15. Quality Intelligence**

The quality intelligence metric is a must if you’re looking for clarity on software quality. It helps in identifying recent code changes. Suppose there are new codes that the team has developed but testing is yet to be done. Maybe there are instances where the quality declines in those codes. Quality intelligence helps determine the same. It makes the team aware of when they should invest more time in testing.

Agile delivers value to both employees and customers. The metrics that we discussed help to optimize the development process in an agile framework. Agile metrics are development centric. But to optimize the production processes, lean is the way to go. Both agile and lean empower teams to deliver better value. They also focus on quick delivery of value to customers. Let’s discuss lean metrics. We’ll get to know what they are and discuss some lean metrics that are musts to include in your development cycle.