Software Measurement

- Measurement
- Need of Measurement
- Types

Metrics

- Characteristics
- Classification
- Agile metrics
- Application monitoring

Software Measurement

A measurement is a manifestation (presentation, clear) of the size, quantity, amount, or dimension of a particular attribute of a **product or process**. Software measurement is a titrate (evaluation) impute of a characteristic of a software product or the software process. It is an authority within software engineering. The software measurement process is defined and governed by ISO Standard.

Need for Software Measurement:

Software is measured to:

1. Create the quality of the current product or process.
2. Anticipate (expect) future qualities of the product or process.
3. Enhance the quality of a product or process.
4. Regulate the state of the project in relation to budget and schedule.

Types of Software Measurement:

There are 2 types of software measurement:

Direct Measurement:

- In direct measurement, the product, process, or thing is measured directly using a standard scale.
- Direct measures include software processes like cost and effort applied and products like lines of code produced, execution speed, and other defects that have been reported.

Indirect Measurement:

- In indirect measurement, the quantity or quality to be measured is measured using related parameters i.e. by use of reference.
- Indirect measures include products like functionality, quality, complexity, reliability, maintainability, and many more.

Metrics (Standard of measurement)

- A metric is a measurement of the level at which any impute (assign, credit) belongs to a system product or process.
- A software metric is **a measure of software characteristics that are quantifiable or countable**.

- Software metrics are important for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.
- Software metrics will be useful only if they are characterized effectively and validated so that their worth is proven. There are 4 functions related to software metrics:

1. Planning
2. Organizing
3. Controlling
4. Improving

Characteristics of software Metrics:

1. **Quantitative:** Metrics must possess quantitative nature. It means metrics can be expressed in values.
2. **Understandable:** Metric computation should be easily understood, and the method of computing metrics should be clearly defined.
3. **Applicability:** Metrics should be applicable in the initial phases of the development of the software.
4. **Repeatable:** The metric values should be the same when measured repeatedly and consistent in nature.
5. **Economical:** The computation of metrics should be economical.
6. **Language Independent:** Metrics should not depend on any programming language.

Classification of Software Metrics:

There are 3 types of software metrics:

1. **Product Metrics:** Product metrics are used to evaluate the state of the product, tracing risks and undercover prospective problem areas. The ability of the team to control quality is evaluated.
2. **Process Metrics:** Process metrics pay particular attention to enhancing the long-term process of the team or organization.
3. **Project Metrics:** The project matrix describes the project characteristic and execution process.
   - Number of software developer
   - Staffing patterns over the life cycle of software
   - Cost and schedule
   - Productivity

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.

Agile Metrics Important for Your Project

Agile metrics measure different aspects of project development. Here are some agile metrics important for your project.

1. Sprint Burndown Report

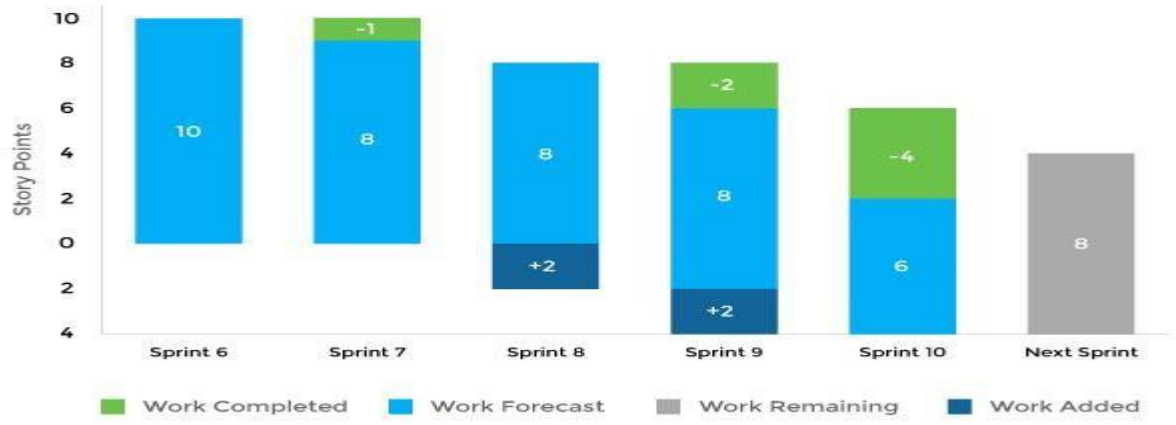
An agile framework comprises scrum teams. They organize their processes into sprints. Since a sprint is time-bound, it's important to track task progress frequently. A sprint burndown report is for tracking the completion of different tasks during a sprint. **Time and work left** to complete are the 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 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.

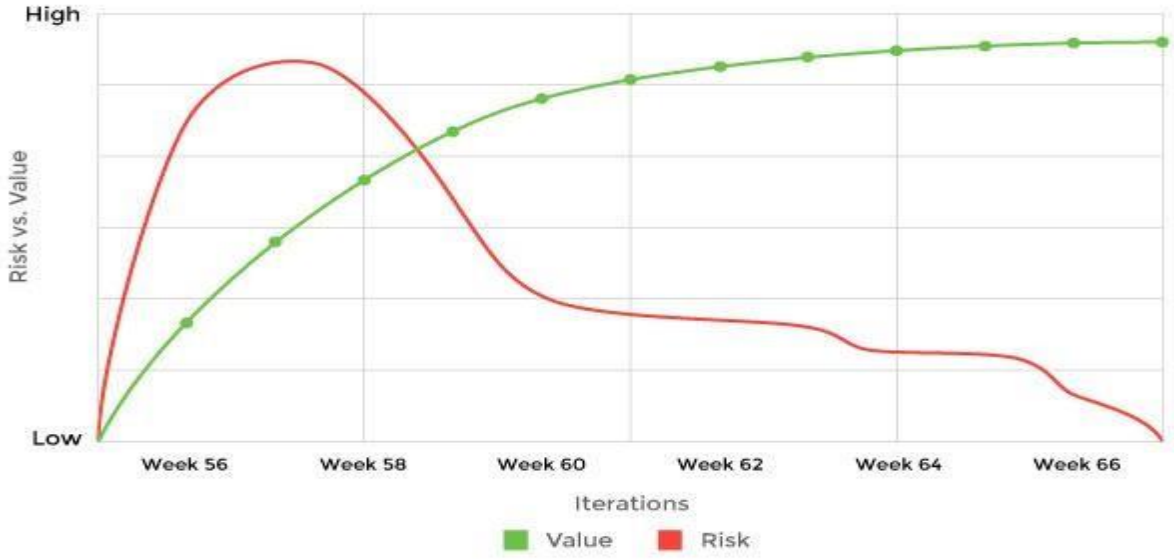
The CFD measures the state of the work in progress. With that, you can take measures to speed up the workflow. The diagram provides a clear visual representation of bottlenecks. After that, the team can take steps to eliminate them and make improvements.

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.

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. 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. You can get a better overview of the capacity of your team.

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.

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.

14. Code Coverage

Code coverage measures the percentage of code unit tests cover. You can run this metric with every build. This metric gives a decent perspective on progress.

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. It makes the team aware of when they should invest more time in testing.

Application monitoring

- Application monitoring is the process of collecting log data in order to help developers track availability, bugs, resource use, and **changes to performance in applications** that affect the end-user experience (UX).
- Application monitoring can be accomplished by dedicated tools to monitor apps, or by collecting and analyzing logs using log management tools. With application monitoring, the end goal is to maximize availability and give customers the best experience.

The main functions of application monitoring tools are:

- **To observe app component**s - Components may include servers, databases, and message queues or catches.
- **To provide app dashboards and alerts** - Dashboards give an overview, alerts drive attention to specific problems.
- **Anomaly detection** - Can vary from simple threshold detection to advanced machine learning pattern recognition.
- **Distributed tracing** - Tracking how one event connects across multiple nodes to detect the origins of errors.
- **Dependency & flow mapping** - A visual representation of how requests travel between services