Project Report

Software Quality Engineering

Submitted by

Junaid Ul Hassan

Anique Khadim

Ubaid Ur Rehman

Rafiq Ahmad

BSSE-6E

Submitted to

Ms. Sumaira

Table of Contents

Title: Software Metrics

Abstract

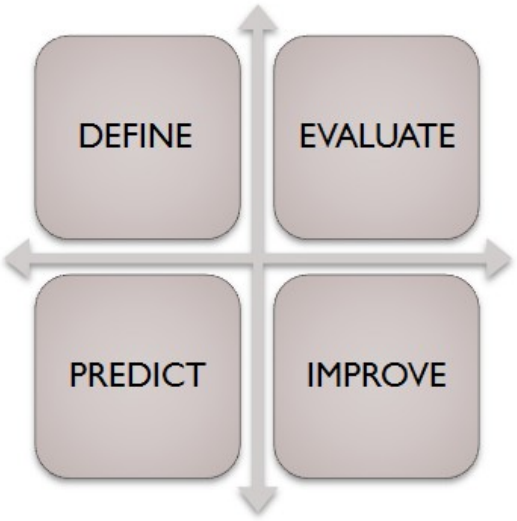
In the world of making software, think of software metrics as helpful tools, like guides that show us how we're doing and how we can do better. They're like measuring sticks that help us plan, make sure our software is really good, and fix any problems quickly. There are three kinds of metrics: Product Metrics (looking at the software blueprint), Process Metrics (helping us get better at making software), and Project Metrics (focusing on specific projects). These metrics act like maps, guiding us through the sometimes tricky journey of software development. In a world where making high-quality software is important, these metrics act like a compass, pointing us in the right direction to create software that's not just good but also efficient and effective. They're like our helpful friends, showing us the way to make the best software we can.

Introduction

A software metric is a measurement of quantifiable or countable software characteristics. Software metrics are essential for various purposes, including measuring software performance, planning work items, and measuring productivity.

- Industry standards (e.g., ISO 9000) and models (e.g., CMMI®) aid in understanding and improving software projects.
- Software metrics offer vital information for making technical decisions.
- Clear comprehension of metrics and their purpose is crucial.
- Metrics programs should be designed to provide precise data for project management and improving software processes.
- Goals for the organization, project, and tasks should be predefined, and metrics chosen accordingly.

Why We use the software metrics?



- To define and categorize elements in order to have better understanding of each and every process and attribute.
- To evaluate and assess each of these process and attribute against the given requirements and specifications.
- Predicting and planning the next move w.r.t software and business requirements.
- Improving the Overall quality of the process and product, and subsequently of project.

The Need for Software Metrics

- In our world, quality is crucial.
- Software metrics give us numbers to plan and predict software development.
- They help monitor and improve software quality easily.
- Everyone agrees that focusing on quality boosts productivity and encourages continuous improvement.
- By measuring problems and defects, we get information to control software products.
- The key metrics are Reliability, Usability, Security, Cost and Schedule, and Efficiency.

Type of Software Metrics

Product Metrics

Product metrics play a crucial role in assessing the current state of a software product and identifying potential risks and underlying areas of concern. These metrics are instrumental in evaluating the team's ability to maintain control over the product's quality. Several key indicators fall under product metrics:

1. Lines of Code

Code length refers to the total number of lines in a software program. This metric serves as a straightforward indicator of the software's size and intricacy. A larger code base generally implies increased complexity, potentially influencing factors like readability and maintenance.

2. Flow Complexity

Flow complexity is a metric quantifying the intricacy of a software's control flow. It evaluates how the program's logic branches and loops, providing insights into potential challenges in comprehension and error detection. High flow complexity may indicate convoluted logic, making it harder to understand and maintain.

3. Code Coverage:

Test coverage represents the percentage of code that automated tests exercise. It serves as a crucial metric in software testing, offering a glimpse into how thoroughly a code base is tested. A high test coverage percentage indicates a comprehensive testing strategy, while low coverage may leave areas untested.

4. Defect Density

Bug density is a metric that quantifies the number of defects or bugs per unit of code. This metric helps teams identify areas of the code base with a higher concentration of issues. By calculating bug density, developers can prioritize bug-fixing efforts, directing attention to sections of code that are more error-prone.

5. Code Maintainability Index

The maintainability score is a composite metric that combines various code attributes to assess how easily a software system can be maintained and enhanced over time. It takes into account factors like code readability, complexity, and modularity.

Example of product metrics

In a software development project, the team aims for a code coverage of 80%. After running automated tests, the code coverage tool reveals that only 65% of the code is being exercised. This metric highlights areas of the code base that lack proper testing, enabling the team to enhance test suites for better coverage and quality assurance.

Process Metrics

Process metrics focus on improving the long-term development and maintenance processes within a team or organization. These metrics are essential for optimizing software development and maintenance activities. Key process metrics include:

1. Checking our planning vs. Reality

Reality vs. Plan Discrepancy refers to the variance between the anticipated and actual effort invested in a project. This metric offers insights into the accuracy of effort estimations, allowing teams to assess the effectiveness of their planning processes.

2. Schedule Variance

Timeline Deviation measures the difference between the planned and actual project schedules. It serves as a key indicator of how well a project adheres to its timeline. Assessing schedule variance enables project managers to identify if a project is progressing ahead, on, or behind schedule.

3. Defect Injection Rate

This metric is crucial for identifying phases in the development lifecycle where defects are most likely to occur. By understanding when defects are injected, teams can implement targeted preventive measures and quality assurance strategies to minimize the overall number of defects, improving the software development process's reliability and efficiency.

4. Lead Time

Task Completion Time, or Lead Time, measures the duration from the initiation to the completion of a specific development task. It provides insights into the efficiency of the development process by identifying how swiftly individual tasks are accomplished. Analyzing lead times helps teams identify bottlenecks or areas where tasks tend to get delayed, enabling process optimization for enhanced productivity and timely project delivery.

Example of Process metrics

In an Agile development team, a user story is estimated to be completed in one sprint. However, the lead time for this story is consistently measured at two sprints. By analyzing lead time metrics, the team identifies inefficiencies in the development process, leading to improvements in task estimation, planning, and workflow.

Project Metrics

Project metrics provide insights into the characteristics and execution of a specific project. These metrics help in monitoring and controlling project activities. Examples of project metrics include:

1. Effort Estimation Accuracy

This metric serves as a critical evaluation of the team's ability to forecast project resource requirements effectively. Analyzing estimation precision helps teams refine their estimation techniques, enhancing the reliability of future project planning and resource allocation.

2. Schedule Deviation

Timeline Variance represents the difference between the planned and actual project schedules. Its purpose is to pinpoint discrepancies from the scheduled timeline, enabling timely corrective actions. Teams use this metric to identify areas where adjustments are needed to ensure that the project stays on track, meeting deadlines and delivering results according to the predetermined schedule.

3. Cost Variance

Cost Variance, is the variance between planned and actual project costs. This metric is vital for assessing the financial performance of the project and facilitating effective budget management. Understanding cost variance allows project managers to control expenditures, make informed financial decisions, and ensure that the project aligns with the approved budget.

4. Productivity

Team Efficiency, or Productivity, measures the development team's effectiveness in delivering outputs relative to the resources expended. This metric provides insights into the team's overall performance, highlighting areas of strength and identifying opportunities for improvement. Teams can use productivity metrics to optimize workflows, enhance collaboration, and ensure efficient resource utilization.

5. Number of Software Developers

This metric offers insights into team size and resource allocation, providing a foundation for understanding the project's human resource dynamics. Understanding the size of the development team is crucial for effective project management, task delegation, and resource planning.

6. Cost and Schedule

This comprehensive metric ensures that project expenses and timelines align with the planned budget and schedule. Regular monitoring allows for proactive management, enabling teams to identify potential issues early, make necessary adjustments, and ensure successful project delivery within established constraints.

Example of Project metrics

In a construction project, the original plan sets the completion date for a building as six months. However, due to unforeseen delays such as weather conditions and material shortages, the project experiences a schedule deviation of two months. This metric prompts project managers to reassess timelines, allocate additional resources if needed, and communicate revised schedules to stakeholders.

Advantages of Software Metrics:

1. Reduction in cost or budget.
2. It helps to identify the particular area for improvising.
3. It helps to increase the product quality.
4. Managing the workloads and teams.
5. Reduction in overall time to produce the product,.
6. It helps to determine the complexity of the code and to test the code with resources.
7. It helps in providing effective planning, controlling and managing of the entire product.

Conclusion

In conclusion, the careful consideration and analysis of product, process, and project metrics are paramount for ensuring the success of software development endeavors. Product metrics, such as lines of code and code coverage, provide essential insights into the size, complexity, and quality of the code base. By focusing on defect density and code maintainability index, development teams can pinpoint areas for improvement, enhancing the overall product quality. Process metrics, including effort and schedule variances, help in refining development and maintenance practices, leading to more accurate planning and timely project deliveries. Defect injection rates and lead time metrics offer valuable information for proactive issue resolution and process efficiency. Project metrics, encompassing aspects like effort estimation accuracy and productivity, contribute to the overall assessment of a project's success, enabling teams to make informed decisions.