QA Metrics
The Value of Testing Metrics Within Software Development

WHITEPAPER

ZEPHIR
Real-Time Test Management
# Table of contents:

3 Introduction

5 Project-level metrics
   1. Requirements and requirement coverage
   2. Defect distribution by status and phase
   3. Defect open and close rates
   4. Execution trends by status and by user

8 Putting project-level metrics together

9 Department-level metrics
   1. Mean Time to Detect and Mean Time to Repair
   2. Defect removal efficiency
   3. Overall testing trends
   4. Defect trends
   5. Burn down chart

12 Putting department-level metrics together

13 Company-level metrics
   1. Number of issues reported by customers
   2. Defect severity index
   3. MTTR and MTTD
   4. Number of system outages and downtime
   5. Effort expended in fixing a problem and releasing it to the customer during post release

17 Optimizing metrics through test management software

18 Testing metrics point the way towards success

References
Introduction

Software testers face a great deal of pressure to get their products released under budget and on time. In recent years, businesses have raised their expectations for development projects, cutting back financial resources while demanding ever shorter production cycles. Everyone is expected to do more with less. That means developers and testers need to take every opportunity to effectively manage the quality of their product or risk releasing flawed software.

Competition within the software market has placed additional strain on project teams to get their latest products released as soon as possible, increasing the risk of a defect-heavy release. The issues experienced with the US Federal Government’s rollout of the HealthCare.gov website is a noteworthy reminder of how a rushed delivery schedule and faulty quality assurance practices can blow up in an organization’s face. If developers and testers do not take more strident steps to improve quality management practices, and managers and supervisors do not bother to keep track of software quality at various stages in the software cycle, they may become the latest media punching bag and poster child for sloppy software development practices. In addition to public focus and attention towards quality issues, it will also, very likely, cause those who would otherwise be interested in purchasing or using the application to shy away from purchasing it. Bad press and an association with shoddy quality practices would require quite an uphill climb to rectify, and that doesn’t include the loss of sales while resolving the issues and restoring a sense of "good faith" to your customers.

To prevent such embarrassing and costly software releases from occurring, project teams need to maintain a high level of accountability and oversight during the development process. Without the capacity to accurately monitor testing measures, team leaders will be unable to effectively manage the quality of software while it is in development. Insights regarding the state of the development cycle must be freely shared between testers, managers and executives in order to ensure that companies are able to identify issues, analyze progress and take corrective action before it is too late.

Introduction
In addition, QA managers need to be able to gauge the effectiveness of their organizational and testing methods over the long haul, as an inefficient approach could affect schedules and quality down the road. Executives similarly require greater oversight capabilities in order to identify systemic flaws in the QA process that could prevent their organizations from releasing high-quality software.

Businesses looking to address these concerns should consider expanding their QA department’s use of software testing metrics. These metrics can provide a great deal of insight into the quality of in-development products as well as the effectiveness of current test methods. Enterprise members from executives to testers can view these metrics and gain a greater understanding of how quality assurance efforts are operating and to what extent they are actually succeeding.

These insights are not simply relegated to the tester level. Quality test metrics will provide a complete visibility up and down the chain of command regarding the effectiveness of software development efforts. This way, everyone from QA managers to C-level executives can glimpse how operations are proceeding and if anything needs to be done to improve the development process.

Project teams should take note, however, that metrics are, at times, most valuable when taken as a whole. Individually, each measure is useful to determining the effectiveness of certain aspects of development, but together, they can provide snapshots and trends of the current project. This gives team members and company leaders a big picture view of the development progress. Otherwise, zeroing in a single metric may wind up misleading company officials, as they are not getting the full story.

Additionally, metrics are irrelevant if they are taken out of context, or do not take into account the unique aspects of the organization that is using them, and what they are looking to measure. There are no universal "best practices". Likewise, there are no "universal metrics" that will work in every capacity all of the time. It is critical that any organization that looks to make any measurements and assess any metrics do so with the goals and focus of their own organization, market, and environment.

At their core, testing metrics can be separated into three basic tiers based on what they are reporting on and where their impact is most felt: project-, department- and company-level. Understanding which high-quality metrics are available to testers and how they can support QA processes is essential to ensuring that developers and testers are effectively managing a company’s standards of quality.
Project-level metrics

Some of the most insightful testing metrics can be gathered at the software project level. They can be collected across an individual development cycle, or over the course of multiple releases and projects. QA members can use this information to gain a better idea of how effective their efforts have been in any specific project. When knee-deep in the production process, it can be difficult for testers to see the forest for the trees.

If testers make a concerted effort to gather software testing metrics at the project level, they can create a clearer snapshot as to where they currently stand. Combining the following metrics into a single project report will allow testers, managers and executives to see how day-to-day QA operations are progressing. Many times, it’s only after software has been released that QA professionals see the bigger picture, and can identify opportunities for improvement. If appropriate project metrics have been collected, it becomes a lot easier to run retrospectives or post-mortems to identify such areas of improvement. Some of the more widely used metrics at the Project-level are:

1. Requirements and requirement coverage

What is it?

Requirements should be clearly outlined at the start of any new development project, so team members understand what goals they should be striving toward.

Why is it important?

The quality of a finished piece of software is often defined by its ability to meet the requirements laid out by developers at the outset of production. If these guidelines are not clearly defined or present significant challenges for them to be met, testers may stumble when conducting their work. Monitoring the effectiveness of these stipulations is absolutely essential to ensuring that a development project gets off on the right foot and maintains its trajectory. Additionally, testers can get a head start on any project by effectively "testing the requirements" and provoking further thought, which itself could lead to fewer "missed test cases".
**How can it be calculated?**

Requirement coverage looks at the cross section between the business requirements and actual processes/workflows. Instead of only focusing on each atomic requirement at the configuration and execution level, using a workflow models can define stronger and more robust test coverage, with an emphasis on the workflows users actually follow.

When creating coverage calculations, each instance of the specific criteria is counted separately. To simplify this, we can consider Requirements Coverage as a percentage to be:

\[
\text{Requirements Coverage} = \left( \frac{\text{Number of requirements covered}}{\text{Total number of requirements}} \right) \times 100
\]

Note: Coverage for each workflow needs to take the following into account:

Flow coverage of a "business process" test is calculated for the entire workflow, not the individual criteria. If flows are run independently, outside of a business process test, then calculate the individual criteria within the workflow. If looking at multiple business process workflows, and each workflow has multiple iterations, calculate each iteration of each configuration.

Reference:

---

2 **Defect distribution by status and phase**

**What is it?**

As a project progresses, both from initial start and as it nears completion, it’s expected that the raw number of identified defects will decrease as the quality of the software improves. However, some areas of the program may still be rife with flaws. Defect distribution rates identify these defect hotspots.

**Why is it important?**

A persistently problematic section of code or unit within the program may indicate some deeper concerns regarding the functionality of the overall product. Identifying these areas through defect distribution rates gives team members time to address them.

**How can it be calculated?**

Though somewhat simplistic, a basic way of looking at defect distribution is to use the following:
Running a defect distribution report will give testers a rundown on the location and severity of all flaws discovered during the course of the project’s run. QA managers can also use their requirement-based metrics to compare how they are meeting stated goals with the number of defects that they continue to encounter.

Defect Distribution = \( \left( \frac{\text{Total number of defects}}{\text{Functional area(s)}} \right) \times \text{Status} \times \text{Phase} \)

Defect open and close rates

What is it?

While there are numerous defect statuses which indicate the current state of an identified flaw, they can typically be categorized as either open or closed.

Why is it important?

The sheer number of defects that are encountered during a project’s run can make them difficult to keep tabs on. If team members are not diligent about measuring the current status of their program’s flaws, certain defects could slip through the cracks and show up in the finalized release. Furthermore, comparing the frequency of open defects with close rates will also provide insight into the ability of testers and developers to work together to identify and address software issues.

How can it be calculated?

Defect Open and Close Rate = \( \frac{\text{Defects found before delivery}}{\text{Defects found before delivery} + \text{Defects found after delivery}} \times 100 \)

Again, a defect report will provide clear insight into the nature of all found flaws. A quality report will break these down by status, giving personnel an opportunity to view trends regarding defect closures throughout the development cycle.

Reference:

4  Execution trends by status and by user

What is it?

These metrics identify which tests have been executed by a given member of the QA team as well as indicate trends related to the status of found defects.

Why is it important?

QA managers can use these metrics to quantify the effectiveness of individual team members and the project unit as a whole. Trends across a single development cycle or multiple productions offer insight into the ongoing ability of a given team to deliver on its promises.

How can it be calculated?

This is not as simple as making a specific equation, but by creating a chart of each tester and comparing the previous three example metrics on a per person basis. Even then, this will not necessarily give an accurate assessment of execution trends, unless each tester is examining the same area. Different contexts will provide different results, and they may not correlate.

\[
\text{Execution Trend} = \left( \frac{\text{Execution Status}}{\text{Execution Rate}} \right) \times \left( \frac{\text{Functional Area}}{\text{Iteration}} \right)
\]

Test execution reports can be created with information detailing these trends intact. QA leaders can then look them over to identify patterns that may indicate changes that need to be made.

Putting project-level metrics together

When project level metrics are paired with a coverage burn down chart, QA members will have a better idea as to how their project has progressed thus far, as well as how much more work needs to be completed. A burn down chart provides a clear view for testers, managers and executives on the current state of development, as well as how much more time and effort must be expended to meet stated goals. Burn down charts allow team members to see where their efforts have been applied thus far, and what will be needed to carry the project to its completion. The burn-down chart also allows all groups to see the various elements in their proper contexts. By gathering and examining these project-level metrics on a regular basis, a clearer picture as to how much time has been spent to meet milestones can be observed, and projections for future milestones can be predicted.
Department-level metrics

Project-based metrics offer insight into the day-to-day operations of a QA team. Department-level metrics can help provide oversight across multiple projects and groups, and give insights as to the operation of a QA group as a whole. These metrics are an aggregate measure, and can be a vital tool to help improve testing efforts down the road.

1 Mean Time to Detect and Mean Time to Repair

**What is it?**

MTTD measures how long it takes QA professionals to find a problem, while MTTR demonstrates the amount of time needed to effectively address it.

**Why is it important?**

To gain a wider view of how QA teams are addressing defects, department leaders can leverage the mean time to detect and remove metrics. Between these two tools, managers will be able to conclude with some degree of accuracy just how effective their team members are.

**How can it be calculated?**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
</tr>
</thead>
</table>
| MTTD         | \[
|              | \frac{\text{Number of issues detected}}{\text{Total execution time}} \]
| MTTR         | \[
|              | \frac{\text{Number of issues fixed}}{\text{Total coding time}} \] |
The detection of any given defect should be recorded and time stamped within a software testing portal. Likewise, it should be noted whenever a particular bug has been fixed.

Comparing these two durations will give QA managers a better idea of how effective their project teams are.

2 Defect removal efficiency

What is it?

Defect removal efficiency is the rate at which team members have been able to adequately address and fix identified flaws in the program.

Why is it important?

Defect-related metrics can provide greater insight into the effectiveness of the production team’s efforts to not only identify flaws but properly address them as well. A QA manager can use the department’s defect removal efficiency rate to determine if personnel are doing enough to ensure that flaws are removed relatively promptly.

How can it be calculated?

\[
\text{Defect Removal Efficiency} = \left( \frac{\text{Number of Pre-release Defects}}{\text{Number of Total Defects}} \right) \times 100
\]

By comparing the raw number of defects identified during the course of a development cycle with the number of flaws actually repaired or eliminated, QA management can discern a team’s overall defect removal efficiency.

3 Overall testing trends

What is it?

When taken as a whole, these various department-level metrics can highlight trends regarding the overall efficacy of a company’s QA efforts.

Why is it important?

Individual metrics can sometimes be misconstrued as a firm descriptor of a project team’s effectiveness, but other factors may play a part as well. Compiling all defect-related metrics and looking at
the bigger picture will give QA management and C-level officers better visibility regarding the quality of the department’s work. Furthermore, these trends can be extended across multiple projects, giving company officials insight into the development of their QA employees as skilled testers.

**How can it be calculated?**

This is a combination of several metrics, including MTTD, MTTR and DRE. Use these metrics as a snapshot to determine testing trends at a particular point in time.

Getting an overall idea of the quality of current and past tester performance requires QA management to compile various metrics regarding defect detection and repair. Once these measurements have been gathered, officials can then analyze them with a longer view.

### Defect trends

**What is it?**

Similar to defect-related metrics highlighted earlier, these measurements show trends in the frequency of software flaws, and the propensity of team members to identify and resolve them. The key difference is, these trends are carried over multiple production cycles.

**Why is it important?**

Viewing the effectiveness of QA teams over the course of several projects will help department and company leaders better determine if any changes need to be made to their testing approach. Trends that perpetuate across numerous development cycles indicate they accurately reflect the team’s capabilities and are not a one-time fluke.

**How can it be calculated?**

This is a combination of several metrics including Defect Distribution, Defect Open and Close Rates, and DRE. Use these metrics as a snapshot to determine defect trends at a particular point in time.

Calculating defect trends requires QA management and the C-suite to gather many of the previously covered metrics and view them from the vantage point of several years. Granular insight is useful but so too is a wider view of QA activity.
Burn down chart

What is it?

Typically completed during the course of an individual project, a burn down chart provides a visualization of the amount of work that has yet to be completed to meet stated requirements.

Why is it important?

Burn down charts offer the opportunity to take a step back and view the progress of development and allows officials to ascertain the effectiveness of their QA employees.

How can it be calculated?

An effective burn down chart can be easily created even within basic spreadsheet software suites if officials have access to the corresponding data. QA managers can take the principles of project-level burn down charts and extend them across multiple projects to get a better idea of how effectively teams are conducting their required work.

Putting department-level metrics together

The total sum of these department-level metrics should help managers and executives determine whether or not their development processes are on the right track. Companies should be able to see some measure of improvement from their production teams, and if not, they can make the necessary changes to foster more effective testing efforts. These metrics allow key decision makers to determine if QA initiatives are progressing, becoming more efficient, and if QA team members are developing their skills. This oversight capability can help managers make better personnel decisions and craft a highly talented QA team.
Company-level metrics

The broadest tier of software testing metrics, company-level measurements, offer executives a clear view as to how QA efforts are translating into the delivery of high-quality software. This information is vital to recognizing how effective a company’s overall software development and testing operation is, and which steps would need to be taken to address or retool processes.

Number of issues reported by customers

What is it?

These metrics outline the number of quality problems that have been experienced by the end user and brought to the company’s attention.

Why is it important?

The end goal of all QA efforts is to facilitate the release of products that adequately meet the needs of a customer or other end user. Tracking the number of issues reported by customers allows businesses to determine the effectiveness of their QA departments. The number of issues reported by customers will not only demonstrate the types and number of defects lingering in released products, but it will also show the breadth of the problem. A widespread issue would indicate that there may be some critical issues with current QA processes.

How can it be calculated?

To get a simple calculation of the percentage of customer reported issues:

\[
\text{Customer Reported Issue Percentage} = \left( \frac{\text{Total number of defects found by customers}}{\text{Total number of product defects}} \right) \times 100
\]
As these issues are directly reported to company officials by the end user, collecting this data isn’t especially difficult. Getting value out of this metric requires businesses to diligently compile this information and reference it against past trends.

Defect severity index

What is it?

Company leaders can measure the degree to which defects are disruptive across multiple projects.

Why is it important?

A company whose projects frequently encounter major defects will be more likely to release flawed products to customers and end users. Looking at the severity of bugs and other flaws across multiple projects can shine some light on the effectiveness of development efforts.

How can it be calculated?

After compiling records regarding defects found in numerous projects, company leaders can analyze these figures to discern how much time and energy was needed to correct them, as well as the extent to which these flaws affected the functionality of the software.

MTTR and MTTD

What is it?

Similar to the concept of MTTR and MTTD at a department level, these metrics show how much time is needed for project members to identify and address defects expanded over the course of numerous development cycles.

Why is it important?

Deploying MTTR and MTTD metrics over multiple projects can help company leaders determine how effective their teams have been and if they have shown improvement in their capacity to quickly and effectively remove flaws from software.

How can it be calculated?

\[
\text{MTTD} = \left( \frac{\text{Number of issues detected}}{\text{Total execution time}} \right) \times \text{Number of projects}
\]
MTTR = \left( \frac{\text{Number of issues fixed}}{\text{Total coding time}} \right) \times \text{Number of projects}

If QA managers have been diligent about recording MTTR and MTTD figures on a department-wide level, executives should have a wealth of information to work with. Using these figures, they can then identify overarching trends in this area among project teams.

4 Number of system outages and downtime

What is it?

Department leaders and C-level officers can track the frequency with which their products experience operational disruptions as well as the duration of each downtime event.

Why is it important?

The quality of a release cannot always be ascertained by the number of customer complaints alone. Measuring outages and tracking downtime periods provides insight into the long-term quality of a product. The number of system outages and length of downtime experienced by the end user will also provide insight into the severity of a software bug.

How can it be calculated?

Mean Time between Failure = \frac{\text{Total time of software system operation}}{\text{Number of critical software system failures}}

Department heads must make it a point to record disruptive incidents as much as possible including outlining the severity of the outage and its duration. By looking at not only the number of downtimes events, but their length as well, companies can better assess their product’s quality.

Reference:

Effort expended in fixing a problem and releasing it to the customer during post release

What is it?

This metric measures the amount of resources needed to identify a flaw following release, repair the defect and get the updated program in the hands of the end user.

Why is it important?

QA efforts rarely end at a product’s release date. Given the complexities of software, issues are bound to crop up once a program has been finalized and pushed out the door. That is why officials must be diligent about identifying and repairing defects as quickly as possible.

How can it be calculated?

Similar to department-level metrics, this asset can be measured by recording and time-stamping any reported end user issues and tracking the amount of personnel and time needed to address them.
Optimizing metrics through test management software

Even when developers have successfully identified which metrics can provide the greatest value to their production processes, they may struggle to get the most value out of their use. Many times, this is because QA team members do not have the tools in place to conveniently share these metrics with one another, or are unable to access them when needed. A comprehensive test management system can address these concerns by providing a single portal through which all metrics can be stored and made available to the relevant parties.

When metrics are used within a partitioned environment, developers will not be able to fully realize their benefits, as some personnel will only have access to certain assets. Furthermore, testers, QA managers and executives may have different figures available to them regarding the quality of ongoing production efforts, preventing organizations from coordinating their resources to meet operational challenges.

A single repository will provide all interested parties with direct access to a wide range of useful metrics. Testers can share information with one another regarding the team’s day-to-day operations and long-term goals. QA managers can evaluate their department’s success on a team-wide or individual level. Executives and other upper management officials, meanwhile, will have access to hard data regarding the quality of their finished products and the progress of their company as a whole. At every level of software development, test management software serves as a convenient platform to provide insight into the production process.

Another benefit offered by this technology is the ability to accurately track development processes at a granular level, including who wrote and executed a particular test script. In the event that executives decide to audit the effectiveness of their QA department, this level of detail will be extremely helpful.
Tracking the quality of software development and testing efforts is by no means an easy endeavor, but it is absolutely critical to accurately gauge the current state of an organization’s health and success. At every level of development, from QA staff members to executives and individual tests to company-wide productions, reliable data is needed to ascertain the quality of current processes and whether changes need to be made to improve operations. Implementing the right testing metrics can help an organization remove inefficiencies and position itself for success. By incorporating a sophisticated software test management system, developers can optimize the value of these assets, facilitating the availability of metrics across the entire enterprise.
References


About Zephyr

Zephyr is a leading provider of quality management solutions, powering intelligent DevTestOps for more than 11,000 global customers across 100 countries. Project teams and enterprises of all sizes use Zephyr’s products to enable continuous testing throughout their entire software delivery pipeline to release higher quality software, faster. Zephyr is headquartered in San Jose, CA with offices in King of Prussia, PA, Europe and India. For more information, please visit www.getzephyr.com.