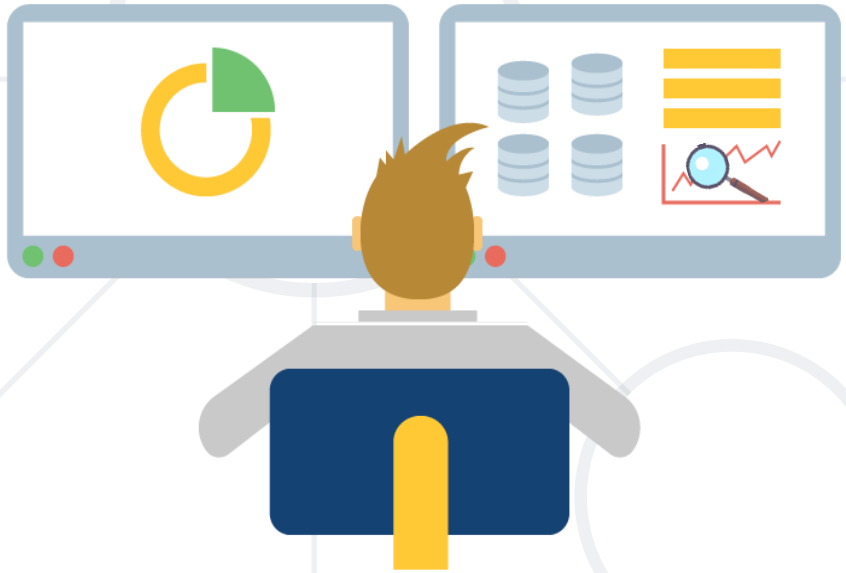
Test Monitoring and Control



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#QA-Fund

Table of Contents

Test Progress Monitoring

Test Progress Monitoring

Software University

- Testing work needs to be tracked
- **Test** progress **monitoring** is a test **management task**
  - Periodically **monitors the status** of a test project
  - Uses **metrics** for measuring actual **progress** against planned milestones
- Gives **visibility** and **feedback** on test **activities**

Status of Test Activities

- Defined by a number of **factors:**
  - The level of test plan and **test case completion**
  - The tested object and its **pass**, **fail** and **blocked** data
  - The quantity of **incomplete tests**
  - Number of **open defects**
  - Amount of **retesting** and **regression** testing required

Test Metrics and KPIs

Measuring Testing Effectiveness

Test Metrics and KPIs

- **Test metrics:**
  - Quantitative measurements
  - Gain insights into testing effectiveness, efficiency, and progress
  - Help to make informed decisions
  - Cover aspects like test progress, coverage, defect metrics and performance
- **Key Performance Indicators (KPIs):**
  - Critical metrics aligned with testing goals, providing actionable insights
  - Inform project management, risk analysis, and stakeholder communication
- Performed throughout software testing lifecycle **from planning to closure**

Common Test Metrics

- **Defect density:**
  - The number of defects identified in a component or system divided by the size of the component or system
  - Expressed in **standard measurement terms** such as lines of code, number of classes, or function points
  - The formula for the calculation of defect density is the **number of defects** divided by **some function points** tested

Common Test Metrics

- **Failure rate:**
  - It can be defined as the **ratio of the number of failures** of a given category to a given unit of measure
  - For example failures per unit of time, per number of transactions, and per number of computer runs
  - Test case failure rate is calculated as some failed test cases divided by the number of test cases executed

Common Test Metrics

- **Test coverage:**
  - Covers the extent of coverage achieved against requirements, risks or code
  - Higher coverage leads to better quality of testing

$$\frac{\text{Number of requirements tested}}{\text{Total number of requirements}}$$

Common Test Metrics

- Percentage of **test case preparation:**
  - Helps the manager identify the extent of preparedness for testing

$$\frac{\text{Test cases prepared}}{\text{Total number of planned test cases}}$$

Common Test Metrics

- Percentage of **test environment preparation:**
  - It is a helpful indicator to gauge the preparedness of testing effort

$$\frac{\text{Test environment preparation complete}}{\text{Total amount of preparation required}}$$

Common Test Metrics

- Percentage of **test case execution:**
  - This is an indicator of the amount of test execution progress achieved

$$\frac{\text{Number of test cases executed}}{\text{Total number of planned test cases to be executed}}$$

Common Test Metrics

- **Defect information:**
  - Defect density, defects found, open and fixed defects are useful metrics for evaluating the software stability and production readiness
  - Defect metrics are also used as an indicator to gauge the overall health of the development process

$$\frac{\text{total number of defects}}{\text{total number of modules}}$$

Common Test Metrics

- **The confidence level of Testers:**
  - The confidence level of Testers in the application or product can be captured through surveys or voting
- **Test milestones dates:**
  - Test milestones dates are set as a part of the test plan
  - These need to be monitored to measure any schedule slippages

Common Test Metrics

- **Testing costs:**
  - Testing costs include cost compared to the benefit of finding the next defect or to run the next test
  - The amount of cost spent in testing including the cost of manpower, environments and associated costs
- Costs should constantly be monitored to ensure that testing does not exceed the budget or to evaluate when to stop testing

Example: Test Metrics

Software University

QA Test Execution as on 11-Jan												
OVERALL Execution Status	Total # of Conditions	% Completed	Test Conditions Executed							% of Test Conditions Deferred	% of Test Conditions On Hold	% of Test Conditions On Failed
			As of Today									
			Planned	On Hold	Executed	Pass	Fail	In Progress	Deferred			
Underwriting	953	98.9%	953	9	923	898	25	1	20	2.1%	0.9%	2.6%
Billing	1438	98.1%	1411	0	1399	1397	2	0	12	0.8%	0.0%	0.1%
Claims	695	93.6%	653	0	611	608	3	0	42	6.0%	0.0%	0.4%
Finance	306	96.1%	296	0	272	250	22	1	23	7.5%	0.0%	7.2%
Printing	2776	76.2%	2379	0	1361	1307	54	27	991	35.7%	0.0%	1.9%
Conversion	671	80.7%	568	18	511	504	7	1	38	5.7%	2.7%	1.0%
Premium rater	1342	79.5%	1067	0	1067	1050	17	0	0	0.0%	0.0%	1.3%
Interfaces	792	84.7%	701	7	555	546	9	2	137	17.3%	0.9%	1.1%
Other Customizations	279	55.6%	197	0	109	93	16	5	83	29.7%	0.0%	5.7%
Underwriting End to End	1222	74.6	918	6	912	816	96	0	0	0.0%	0.5%	7.9%
Total	10474	84.6	9143	40	7720	7469	251	37	1346	13	0	2

Key Performance Indicators (KPIs)

- The concept of **KPIs** are **specific metrics** used to gauge the **performance** and **progress** of the testing process against defined objectives and goals
- **Benefits** of Test Metrics and KPIs:
  - Objective measurement of testing progress and effectiveness
  - Early identification of potential issues or bottlenecks
  - Data-driven decision-making for process improvement
  - Improved visibility and communication of testing outcomes to stakeholders

Most common KPIs

Test Reporting

Test Reporting

- **Test reports:**
  - Submitted for the testing period at the end of each phase or test project
- Test leaders generate multiple reports during the test planning, design and execution phase
  - They report the progress of test activities
- Reports:
  - Keep all **stakeholders informed**
  - **Help** the **Test Leader** get **attention** or **resources** to resolve project risks

Test Summary Reports

- Test team maintains different reports like **daily**, **weekly**, **monthly** and even **quarterly** status reports
- Test summary report should have **recommendations** and **decisions** for **future actions**, based on the metrics collected
- Include **lessons learned**, which helps in **preventing** repetitive mistakes for future phase

Example: Test Report

Test Report

Test Cycle System Test

EXECUTED	PASSED			130	
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				**130**
PENDING					0
IN PROGRESS					0
BLOCKED					0
(Sub-Total) TEST PLANNED (PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)					**130**

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	*High*	
Edit Customer	Check Customer can be edited	100%	100%	0	*High*	
New Account	Check New account is added	100%	100%	0	*High*	
Edit Account	Check Account is edit	100%	100%	0	*High*	
Delete Account	Verify Account is delete	100%	100%	0	*High*	
Delete customer	Verify Customer is Deleted	100%	100%	0	*High*	
Mini Statement	Verify Ministatement is generated	100%	100%	0	*High*	
Customized Statement	Check Customized Statement is generated	100%	100%	0	*High*	

Test Control

Test Control

Software University

- **Test control** is a test management task
  - It deals with development and application of a **set** of **corrective actions**
  - It is done to get a test project on track when monitoring shows a **deviation** from plan
  - Actions may **cover any test activity** and may affect other software life cycle activity or task

Example: Test Control

- In a software development project, there are automated **daily regression tests** to ensure the software's stability
- However, a **critical bug arises** that demands immediate attention from the development team
- To address this issue effectively, test control may involve temporarily **pausing** the **daily regression tests**
- The **resources** previously allocated for regression tests are **redirected** to **resolving** the **critical bug**
- Once the bug is fixed, the project can resume its regular testing routine, including the daily regression tests, to maintain software stability

Risk Management in Testing

The process for handling risks

Risk Management in Testing

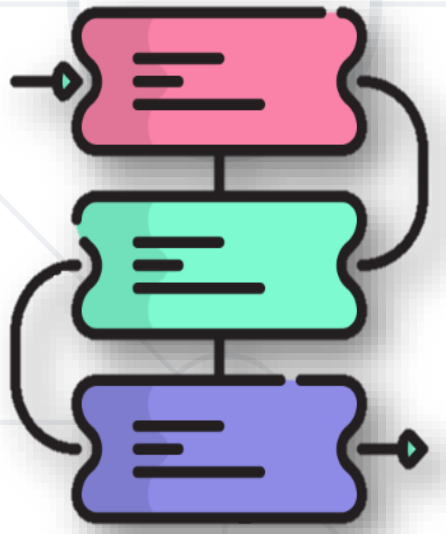
- Involves the following tasks, associated with testing activities:
  - **Identifying risks**
  - **Analyzing risks**
  - **Mitigating risks**
- Ensures that **potential issues** are addressed

Risk Identification

- Identifying risks early in the testing process
- Various sources **can introduce risks**
  - Project **complexity**
  - Requirements **volatility**
  - Resource **constraints**
  - Technology **dependencies**
- Significance of **involving stakeholders**, including business analysts, developers, and testers, in the risk identification process

Risk Analysis and Prioritization

- **Explain** the **process of analyzing and evaluating** identified risks
- **Discuss** techniques such as **probability and impact assessment**, risk matrices, and risk categorization
- **Emphasize** the **importance of prioritizing risks** based on their potential impact on the project and testing objectives



Test Closure

The formal process of completing testing activities

Test Closure

- **Involves**
  - Finalizing testing activities
  - Evaluating the test effort
  - Gathering valuable insights for **process improvement**
- Ensures that **objectives are met**, **risks are mitigated**, and **quality standards are achieved**

Objectives of Test Closure

- **The primary objectives** of test closure include:
  - Ensuring that all **planned test activities** have been **completed**
  - **Collecting and analyzing** test-related data and artifacts for **reporting and decision-making**
  - Assessing the **effectiveness** and **efficiency** of the **testing process**
  - Capturing **lessons learned** to improve future **testing efforts**

Test Closure Activities

- **Finalizing and documenting** test results - Consolidate and document test outcomes (test case execution status, defects, relevant metrics)
- **Analyzing test coverage** - **Review** test coverage achieved and assess if requirements and risks have been **sufficiently covered**
- **Conducting** a post-mortem review - **Evaluate** the testing process, identify strengths and weaknesses, gather **feedback** from team members through a retrospective meeting
- **Archiving** test assets - **Preserve** test documentation, scripts, data, and artifacts for future reference and audits

Test Completion Checklist

- **Comprehensive document** that verifies all essential testing activities have been completed before moving forward with the software release
- Typically **includes** the following items:
  - **All Test Cases Executed:** Ensuring that planned test cases have been executed and their results recorded
  - **Defect Resolution:** Verifying that reported defects have been fixed, verified, and closed appropriately
  - **Test Environment Cleanup**

Test Completion Checklist

- **Documentation:** All test-related documentation, such as test plans, test cases, and test reports, are complete and up to date
- **Test Closure Activities:** Verifying that all Test Closure activities have been performed

- The Test Completion Checklist acts as a **final validation** to ensure that all necessary testing tasks have been addressed and software is ready for deployment

Checklist Example

Software University

Exit Criteria	Status
100% Test Scripts executed	Done
95% pass rate of Test Scripts	
No open Critical and High severity defects	
95% of Medium severity defects have been closed	
All remaining defects are either canceled or documented as Change Requests for a future release	
All expected and actual results are captured and documented with the test script	Done
All test metrics are collected based on reports from HP ALM	
All defects are logged in HP ALM	Done
Test Closure Memo is completed and signed off	

Test Summary Report

- Formal document - **overview** of the testing effort and its outcomes
- Serves as a **communication tool** for stakeholders. Includes:
  - **Introduction:** An overview of the testing objectives, scope and timelines
  - **Test Progress:** A summary of test progress
  - **Defect Summary:** An overview of defects found during testing, including their severity, priority, and status

Test Summary Report

- **Test Coverage:** Details of test coverage - requirement, code and risk coverage
- **Test Metrics:** Key test metrics and KPIs used to evaluate the testing process's effectiveness
- **Recommendations:** For process improvement or further testing actions

- The Test Summary Report provides stakeholders with a **clear understanding** of the testing effort's results and helps in decision-making regarding the software's release and overall quality

Grafana

Grafana

Software University

- **Grafana** is a popular open-source platform for:
  - **Data visualization:** It creates beautifully interactive charts, graphs, and dashboards to help you understand your data at a glance
  - **Monitoring:** It keeps track of key metrics from various sources like databases, servers, applications, and infrastructure. This helps you identify issues and trends quickly
  - **Querying and exploration:** You can ask questions about your data through ad-hoc queries and explore it in detail
  - **Alerting:** Set up notifications to be sent to different platforms like Slack or PagerDuty when specific conditions are met

Grafana in QA

- **Test Execution Dashboards**:
  - Track test suite pass/fail rates, execution time, and test type distribution
  - Spot trends, regressions, or improvements in test results
  - Compare test performance across different environments
- **Test Automation Insights:**
  - Monitor automation framework metrics (e.g., test count, error rates)
  - Visualize flaky test occurrences
  - Assess code coverage by automated tests

Grafana in QA

- **Performance Testing Analysis:**
  - Visualize performance metrics (response times, throughput, errors) during load testing
  - Compare performance changes across software versions
  - Correlate performance data with infrastructure metrics
- **Defect Tracking and Analysis:**
  - Monitor defect trends, severity, and resolution time
  - Identify areas with high defect rates
  - Track the effectiveness of bug fixes

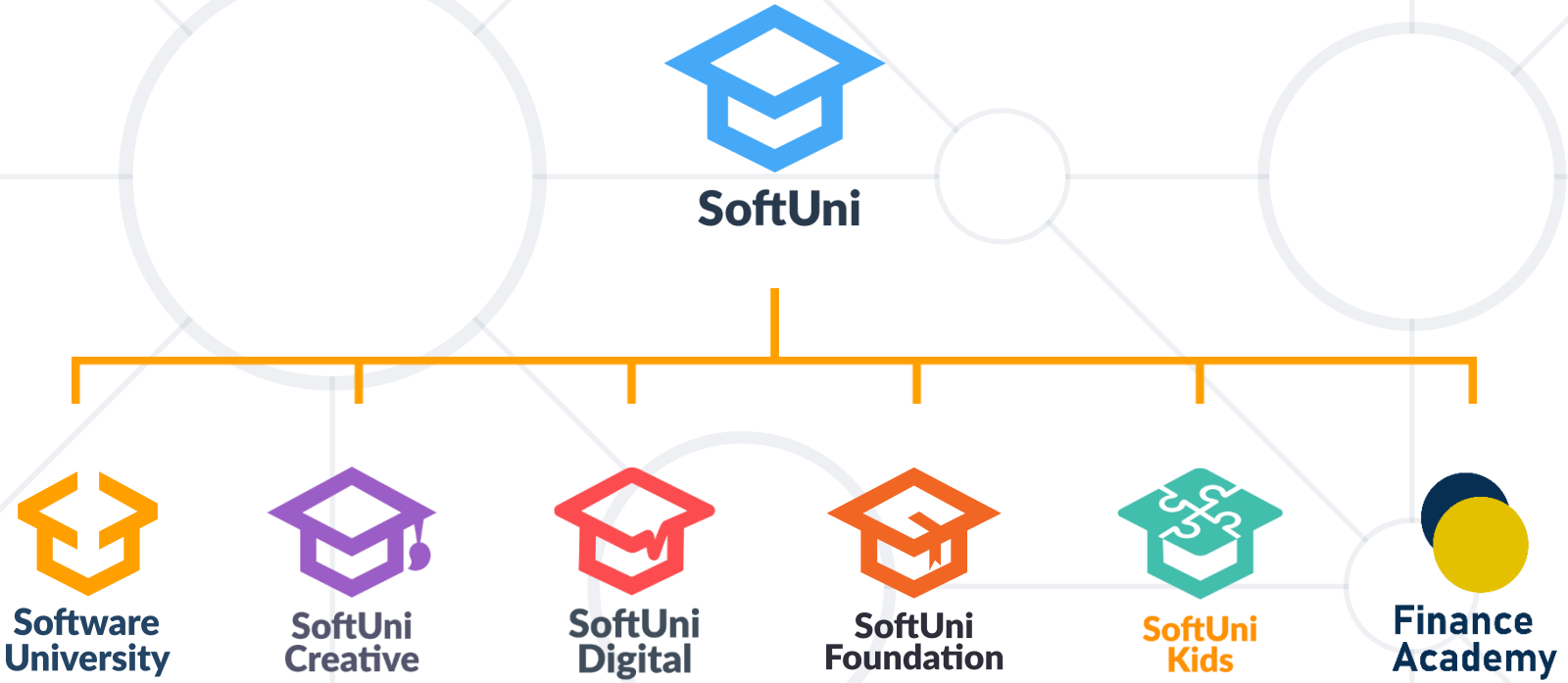
Explore Grafana

- Grafana is a powerful tool for anyone who wants to visualize, analyze, and understand their data better.
- Explore the **countless dashboards**:
  - https://grafana.com/grafana/dashboards/
- **Play** with Grafana:
  - https://play.grafana.org/d/000000012/grafana-play-home?orgId=1

Summary

- **Test progress monitoring** and **test reporting**
- Understanding **test metrics** and **KPIs**
- How to conduct **control over test process**
- What is **risk management**?
- **Test Closure** – checklist, summary, activities
- **Grafana** – powerful **analytics** and **monitoring** platform

Questions?



SoftUni Diamond Partners

Software University

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg