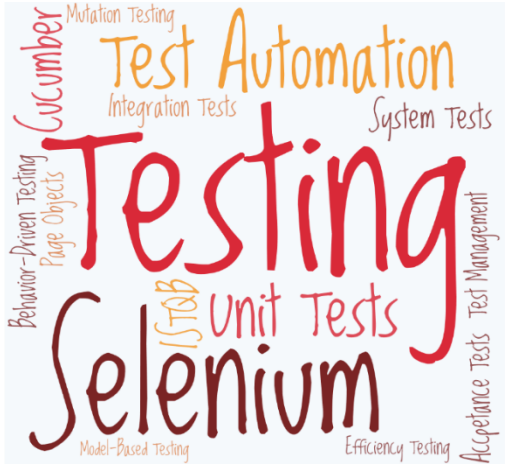
183.290 Software Testing



peso PERFECTING SOFTWARE

Risk-Based Testing
Test Documentation
Test Management Tools

WS 2020

Christina Zoffi

peso@inso.tuwien.ac.at
https://peso.inso.tuwien.ac.at

Content

Risk-Based Testing – Risk

- **Definition**
  - The chance of an event, hazard, threat or situation occuring and resulting in negative consequences or a potenial problem
- **Expected loss** due to failures that result from errors that remain in the software
- **Risks are determined based on**
  - Likelihood of occurrence
  - Failure Impact/Consequence
  - **R**(isk) = **P**(robability) * **C**(ost)
- Separation into **project** and **product risks**

Risk-Based Testing – Project Risks

- Risks that are related to **management and control** of the project and that threaten the **project‘s capability** to **deliver** its **objectives**:
  - **Organisational**
    - Lack of skills/training, shortages of staff
    - Communication conflicts/Political issues
  - **Technical**
    - Requirement issues (wrong, missing, too complex)
    - Test environment/data not ready
    - Poor defect management
  - **Supplier**
    - Failure of third party
    - Contractual issues
    - Change in legal aspects/regulations

Risk-Based Testing – Product Risks

- All risks that are **directly related** to the **test object** (software)
- Also called **quality risks**

- **Examples for product risks:**
  - Non-conformity to the specification (verification)
  - Non-conformity to stakeholders expectations (validation)
  - Non-functional requirements not met by the system architecture
  - Bad response times
  - Bad usability

Risk-Based Testing

- **Definition**
  - An approach to testing to reduce the level of product risks, starting in the initial stages of a project
- It is **not feasible/reasonable** to test a system with **all combinations** of **input/output/environment** parameters
- **Risks are used to prioritize**
  - … where to start testing
  - … where to test more
- **Risk-based testing allows to reduce either**
  - the likelihood of occurrence
  - the impact/consequences

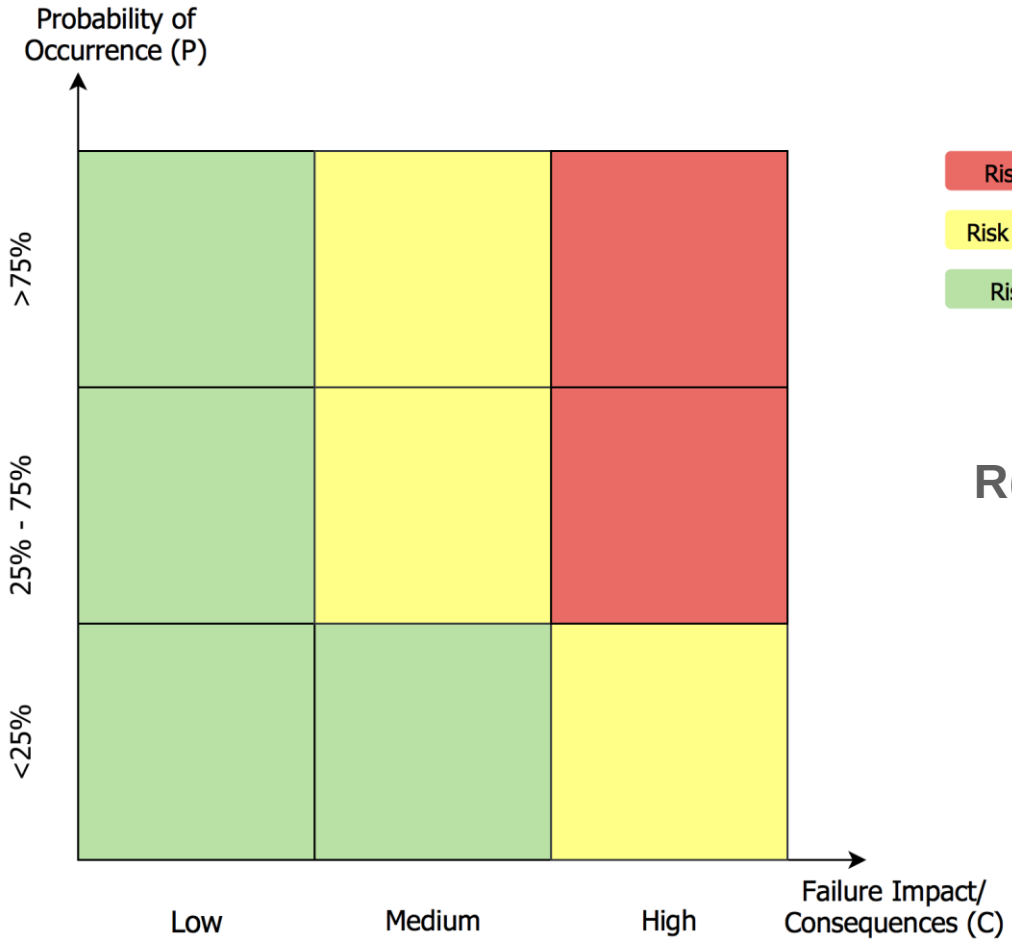
Risk-Based Testing

- Involves planning the **test approach** in order to **minimize risks**
- Includes the **identification/classification/monitoring** of product risks
- → Close connection between **risk** and **test management**

- **Identified risks may be used to**
  - Select the test techniques and test levels/types to be applied
  - Determine the scope/extent of testing
  - Prioritize testing with the goal to find critical defects as early as possible
  - Determine whether non-testing activities can help to reduce risks (e.g. trainings for staff)

Risk-Based Testing – Risk Matrix

- **Focus on testing high-priority features**



R(isk) = **P**(robability) * **C**(ost)

Risk-Based Testing – Risk Priority Table

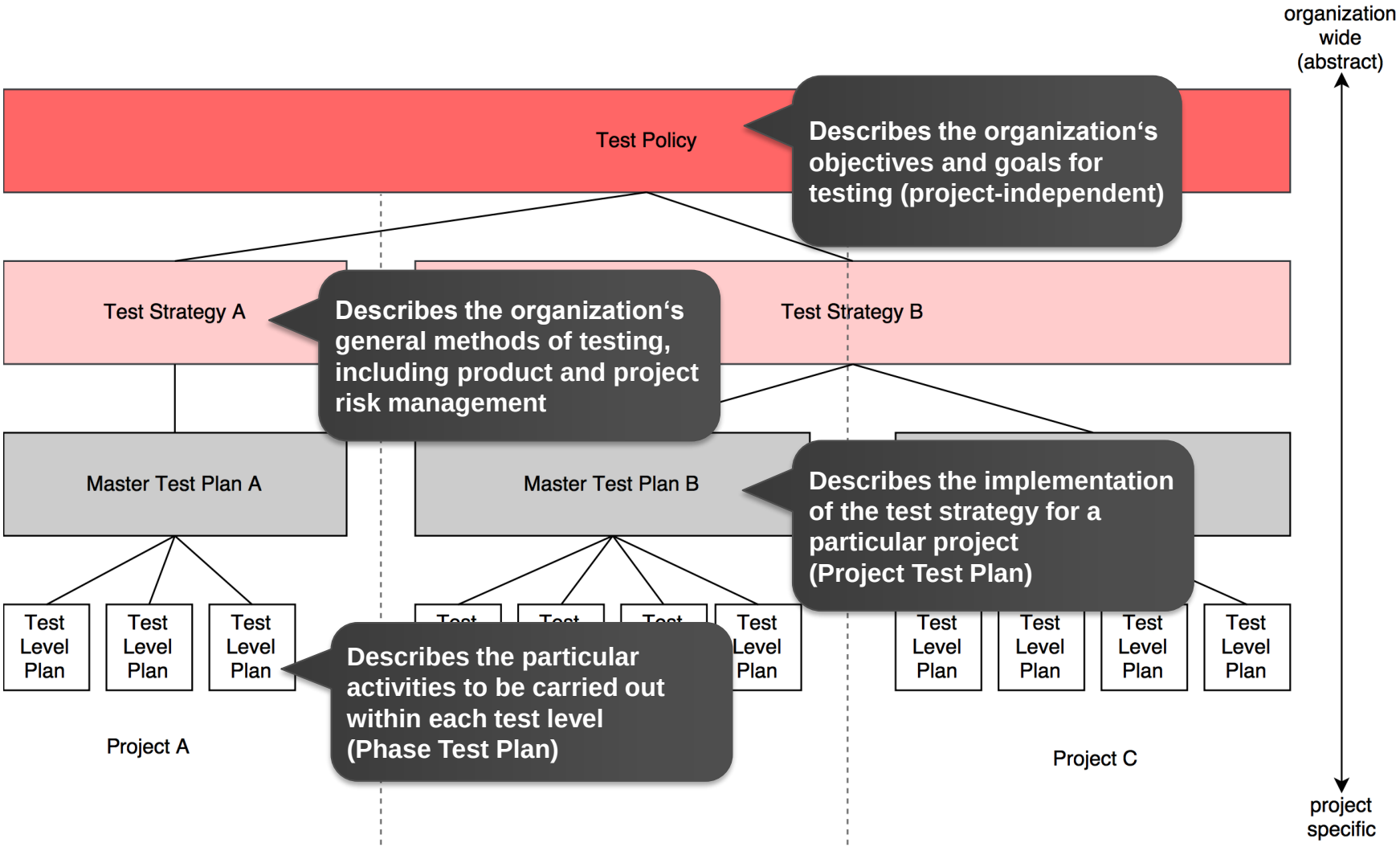
- **Example ATM Machine**

Features/Attributes	Probability (P)	Cost (C) Failure, Consequence	Priority (P * C)
Withdraw Cash	High (3)	High (3)	9
Transfer Money	Medium (2)	Medium (2)	4
Read Balance	Low (1)	Low (1)	1
Make Payment	Low (1)	High (3)	3
Security	Medium (2)	High (3)	6

- **→ Focus on testing high-priority features**

Content

Test Documentation



Test Documentation – Test Policy

- **Definition**
  - The test policy is a high-level document which describes the principals of testing and the general testing approach
  - Within this document all major organizational goals related to software testing are defined
  - Policy is valid for a whole organization

- **Content**
  - Definition of a test process
  - Evaluation of the effectiveness/efficiency of testing
    - Cost of testing vs. cost of errors
  - Definition of quality targets
    - Reliability/Usability/…
  - Approach for test process improvement

Test Documentation – Test Strategy

- **Definition**
  - Describes the project and product risks and how the test process manages these risks.
  - Provides the generic test requirements for one or more projects

- **Based on the test policy**
- **Tailored to the organization‘s needs and means**

- **2 Objectives**
  - Identification/Description of **all risks** which need to be covered in testing
  - Identification/Description of **all test levels** and **test activities** in order to cover these risks

Test Documentation – Test Strategy II

- **Typical test strategies may include:**
  - Test entry and exit criteria
  - Test approach
  - Desired level of independence of the test team
  - Mandatory and optional standards/legal requirements
  - Test environment specification
  - Test automation (desired level of automation)
  - Defect management
  - Regression testing strategies

Test Documentation – Master Test Plan

- **Definition**
  - Describes the implementation/realization of a test strategy for a particular project
  - Addresses multiple test levels
- **Complements the project plan or operations guide**

- **Typical contents:**
  - Items to be tested/excluded
  - Quality attributes to be tested/excluded
  - Time management from test perspective
  - Resource management from test perspective (budget)
  - Test execution cycles and their relationship to release plan
  - Specific test entry/exit criteria

Test Documentation – Level Test Plan

- **Definition**
  - Describes the activities to be carried out within a particular test level.
  - Expands the master test plan

- **Typical contents**
  - Schedule/Task/Milestone details
  - Standards and templates for specification of tests

- **In smaller projects/organizations master and level test plan are often combined to single document**

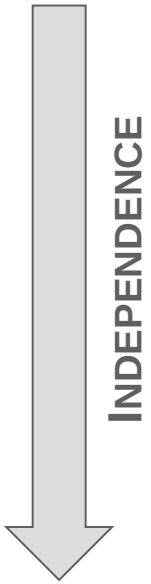
Content

Test Management – Independence of Testing

- How is testing installed within an organization?

- **Different levels of independence:**

1. No test team, developers test their own code
2. Independent test team within project team
3. Independent test team within organization
4. Independent external test specialists
5. Independent external/outsourced test team



- **Independence varies depending on test level**
- Independent test teams can **improve effectiveness** of testing

Test Management – Independence of Testing

- **Benefits of independent test teams**
  - Higher level of objectivity
  - Unbiased
    - Verify/Question assumptions made during design/implementation
  - Different point-of-view
  - Different technical perspective
  - Find different type of defects

- **Drawbacks of independent test teams**
  - Increasing level of independence → increasing isolation from development team
  - Less responsibility for quality by developers
  - Seen as bottlenack/blamed for delays
  - Lack in important business information

Test Management – Roles in Testing

- **Test Manager**
  - Also referred to as **test coordinator**
  - **Overall responsibility** for **management** and **monitoring** of **test activities** and **resources**
  - Position is often taken by project/quality/development manager

- **Tester / Test Engineer**
  - A skilled professional who is involved in testing a component or system
  - Different people can take the role as a tester
    - Unit/Integration testing → developer
    - Acceptance testing → user/business expert
    - Usability/Security/… testing → specialists
    - System testing → independent testing team

Test Management – Test Manager

- **Tasks involve…**
  - Definition/Maintenance of the **organization‘s test policy**
  - Definition/Review of **test strategies**
  - **Planning** of test activities (test objective, risks, effort estimation, cost, resources, dependencies, …)
  - **Writing and coordinating test plans**
  - **Monitoring** of **test acitivites** and **test progress**
  - Definition of actions to compensate **problems/risks**
  - Decision for **test automation**
  - Selection of **test tools**
  - **Reporting**
  - Setup and maintenance of **defect management system**

Test Management – Tester

- **Tasks involve…**
  - Review/Extension of **test plans**
  - Analysis and review of **requirements** for testability
  - Definition and implementation of **test cases**
  - Preparation of **test data**
  - Preparation of **test environment**
  - **Implementation and execution** of tests
  - Evaluation/Documentation of **test results**
  - **Test automation**
  - Usage of **test tools**
  - Evaluation of **non-functional requirements**

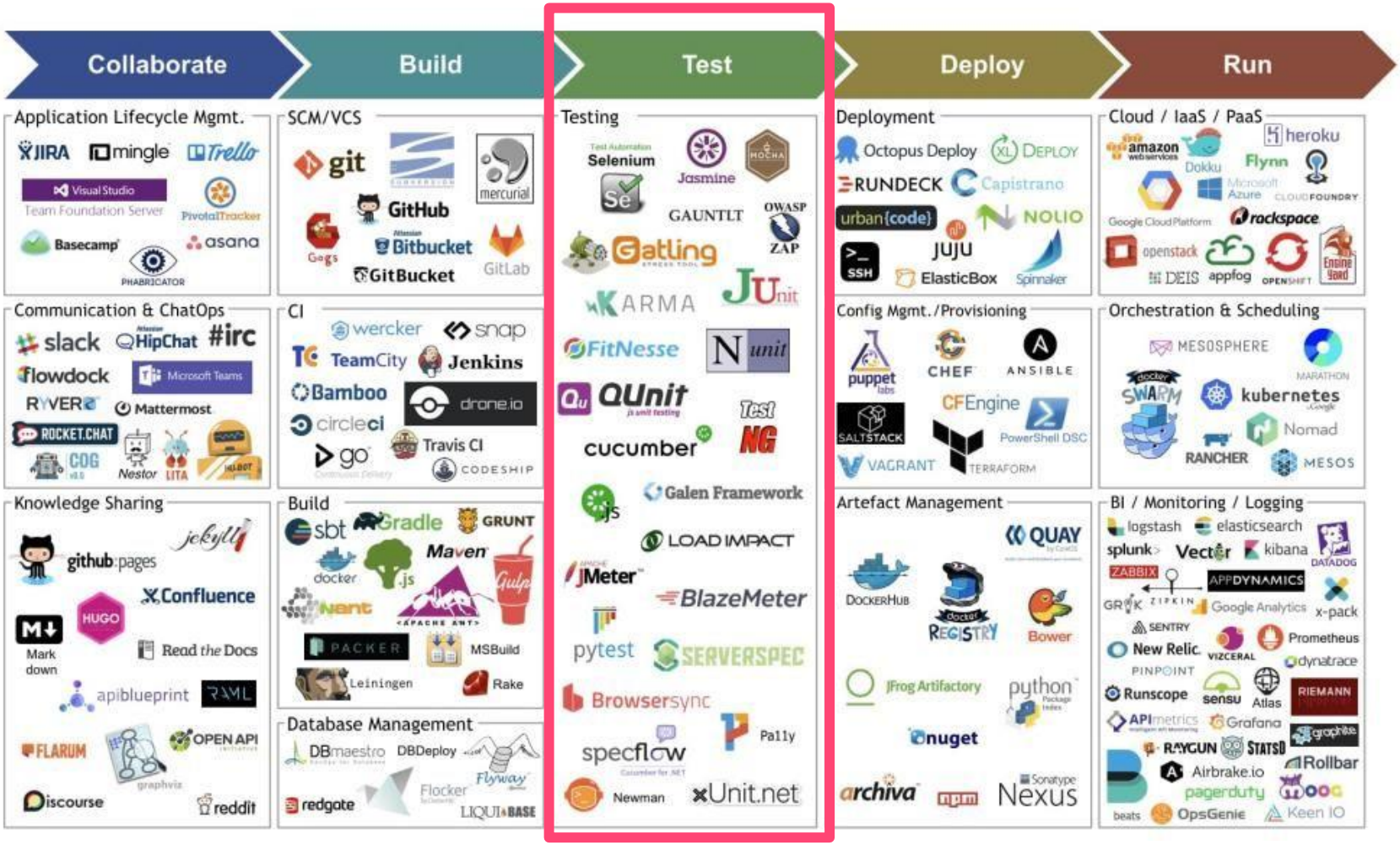
Test Management – Agile Development

- **Independence**
  - Testers as part of the development team
  - In addition, testers as part of a larger independent testing team
  - Product owner performs acceptence testing at the end of each iteration

- **Roles**
  - No strict line between test manager and tester
  - Certain tasks of the test manager may be handled by the team
  - Agile testers are concerned with testing within the project
  - Test managers are concerned with topics that span multiple projects and coordinate resources between them

Content

Test Tools


Collaborate
Build
Test
Deploy
Run
Application Lifecycle Mgmt.
JIRA
mingle
Trello
Visual Studio
Team Foundation Server
PivotalTracker
Basecamp
asana
PHABRICATOR
SCM/VCS
git
mercurial
GitHub
Bitbucket
GitBucket
GitLab
Gogs
Testing
Selenium
Jasmine
MOCHA
GAUNTLT
OWASP ZAP
Gatling
JUnit
KARMA
FitNesse
Nunit
QUnit
Test NG
cucumber
Galen Framework
LOAD IMPACT
JMeter
BlazeMeter
pytest
SERVERSPEC
Browsersync
specflow
Pally
Newman
Unit.net
Deployment
Octopus Deploy
DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
SSH
JUJU
ElasticBox
Spinnaker
Cloud / IaaS / PaaS
amazon web services
heroku
Dokku
Flynn
Microsoft Azure
CLOUD FOUNDRY
Google Cloud Platform
rackspace
openstack
DEIS
appfog
OPENSHIFT
Engine Yard
Communication & ChatOps
slack
HipChat
#irc
flowdock
Microsoft Teams
RYVER
Mattermost
ROCKET.CHAT
COG
Nestor
LITA
HUBOT
CI
wercker
snap
TeamCity
Jenkins
Bamboo
drone.io
circleci
Travis CI
go
CODESHIP
Config Mgmt. / Provisioning
puppet labs
CHEF
ANSIBLE
SALTSTACK
CFEngine
PowerShell DSC
VAGRANT
TERRAFORM
Orchestration & Scheduling
MESOSPHERE
MARATHON
SWARM
kubernetes
Nomad
RANCHER
MESOS
Knowledge Sharing
github pages
jekyll
Confluence
HUGO
Mark down
Read the Docs
apiblueprint
RAML
FLARUM
OPEN API
graphviz
discourse
reddit
Build
sbt
Gradle
GRUNT
docker
Maven
.js
Gulp
Nant
APACHE ANT
PACKER
MSBuild
Leiningen
Rake
Artefact Management
QUAY
DOCKERHUB
REGISTRY
Bower
JFrog Artifactory
python
nuget
archiva
npm
Sonatype Nexus
BI / Monitoring / Logging
logstash
elasticsearch
splunk
Vector
kibana
DATADOG
ZABBIX
APPDYNAMICS
GROK
ZIPKIN
Google Analytics
x-pack
SENTRY
New Relic
VIZCERAL
Prometheus
dynatrace
PINPOINT
Runscope
sensu
Atlas
RIEMANN
APImetrics
Grafana
graphite
RAYGUN
STATSD
Airbrake.io
Rollbar
pagerduty
beats
OpsGenie
Keen IO
Database Management
DBmaestro
DBDeploy
Flyway
redgate
Flocker
LIQUIBASE

Test Tools

- **Test tools support one or more test activities**
  - Planning and Control
  - Test Design
  - Test Data Generation
  - Text Execution
  - Test Reporting

- **Purpose of test tools**
  - Improve efficiency
  - Automate repetitive tasks
  - Automate activities that require significant resources
  - Automate activities that cannot be done manually
  - Increase reliability of testing
  - Allow traceability and transparency

Test Tools – Benefits/Risks

- Tool support does **not** automatically **guarantee success/ improvement**

Benefits	Risks
• Reduction of repetitive tasks (e.g. regression tests) • Improved consistency and repeatability • Objective assessment (metrics, coverage, …) • Improved access to information/reports (charts, defect rates, …)	• Unrealistic expectations • Underestimating time/cost/effort for initial setup and training • Underestimating time/effort to gain significant benefit • Poor integration/maintenance of the tool and its artifacts • Neglection of dependencies/ interoperabilities between test activities or tools • Vendor dependence (poor support, bankruptcy, tool retirement, …) • Lacking ownership

Test Tools – Management Tools

- **Support test activities over the entire software life cycle**
  - Test Management Tools
  - Requirements Management Tools
  - Defect Management Tools
  - Configuration Management Tools
- **Interaction with other tools required to exchange information**

- Examples
  - Squash TM
  - TestLink
  - Redmine
  - Trac
  - GitLab / GitHub
  - Mantis Bug Tracker
  - JIRA (commercial)
  - MicroFocus HP ALM (commercial)
  - YouTrack (commercial)

Test Tools – Static Testing Tools

- **Support identification of defects at an early project stage without execution of the software**
  - Review Tools
  - Code Analysis Tools

- **Examples**
  - Gerrit
  - GitLab (integrated)
  - UpSource (commercial)
  - SonarQube
  - SonarLint
  - MetricsReloaded
  - FindBugs

→Course „Software Quality Management“ in summer term

Test Tools – Test Execution Tools

- **Enable (semi-) automated test execution**
  - Unit Testing Frameworks
  - UI Testing Frameworks
  - Coverage Measurement Tools
  - Performance Testing Tools

→ **Lecture 4 – Test Automation**

- **Examples**
  - xUnit (e.g. JUnit)
  - TestNG
  - Selenium
  - JaCoCo
  - Clover
  - Squash TA
  - Tosca (commercial)
  - HP Unified Functional Testing (commerial)
  - IBM Functional Tester (commercial)

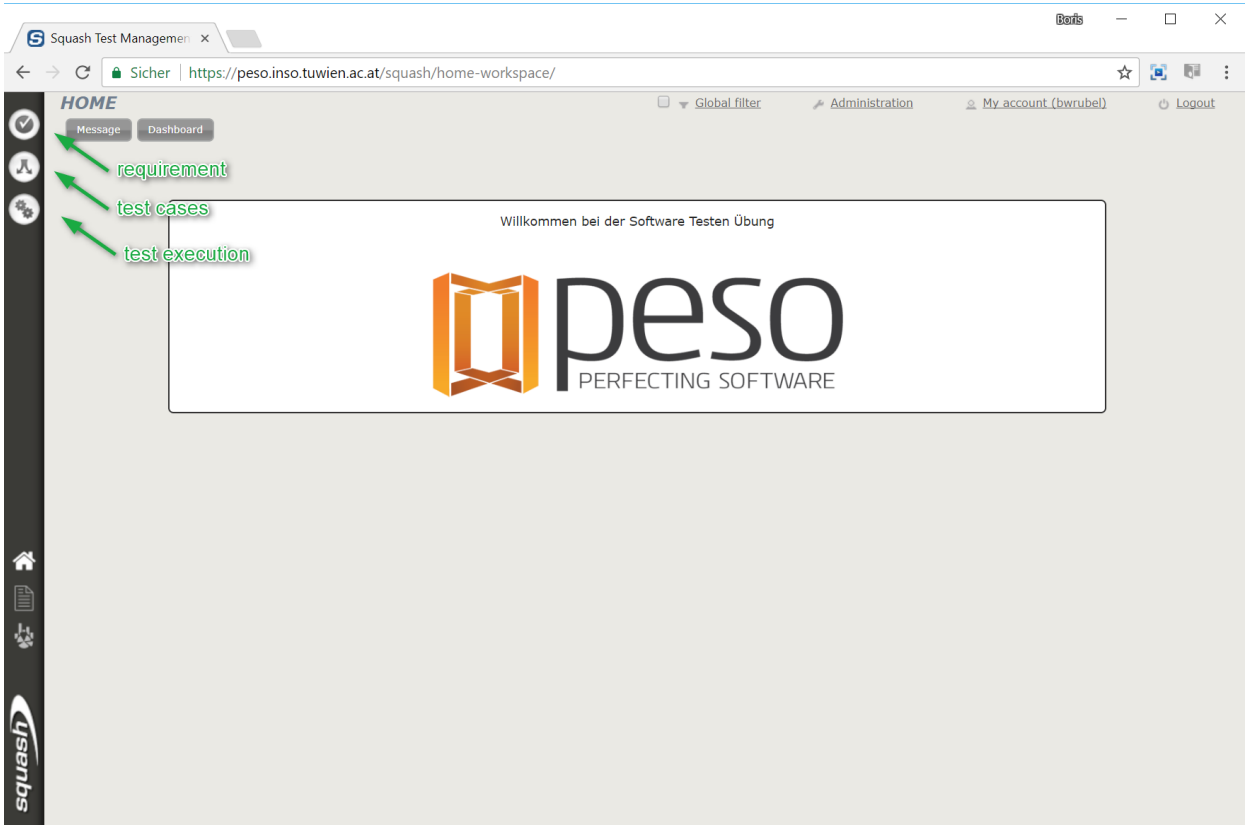
Test Tools – Squash TM

- **Open Source Test Management Tool**

  http://www.squashtest.org/

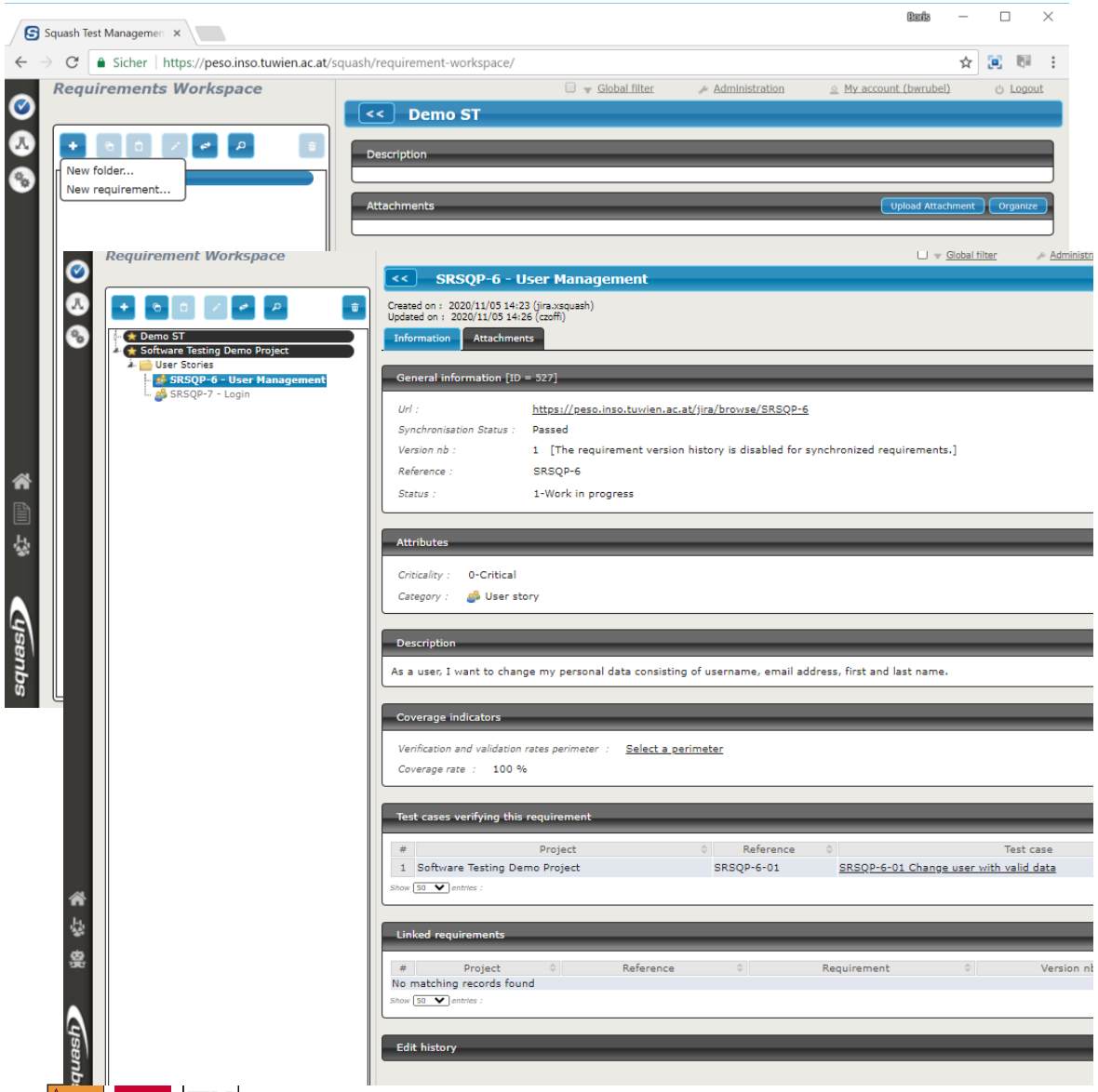
- **Web-based**
- **Will be used at group project for**
  - specification of test cases
  - manual test execution
  - test planning
  - traceability between test cases and requirements

Squash Test



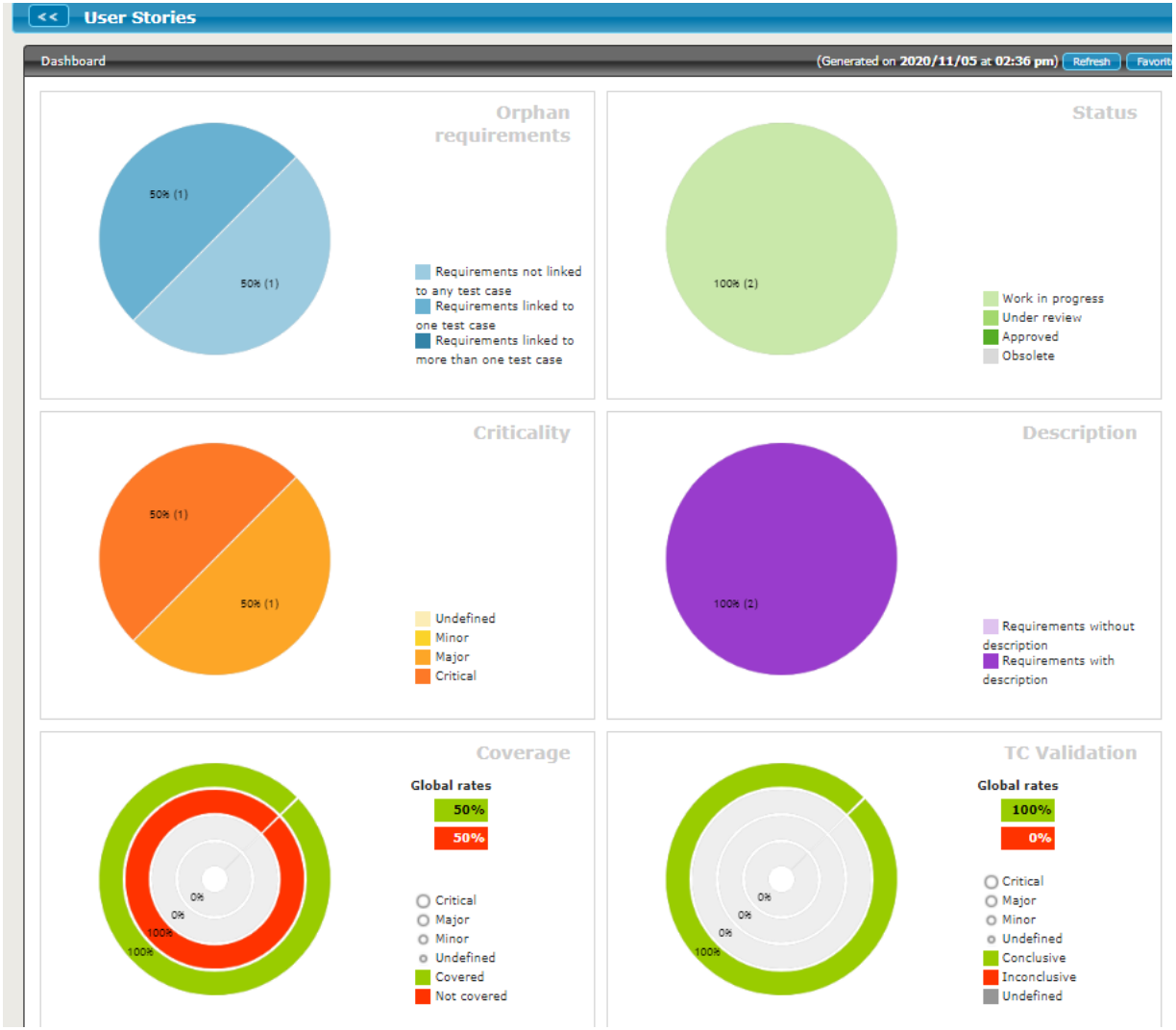
- **Squash TM** is divided into different workspaces:
  - Requirements
  - Test Cases
  - Test Execution

Squash Test



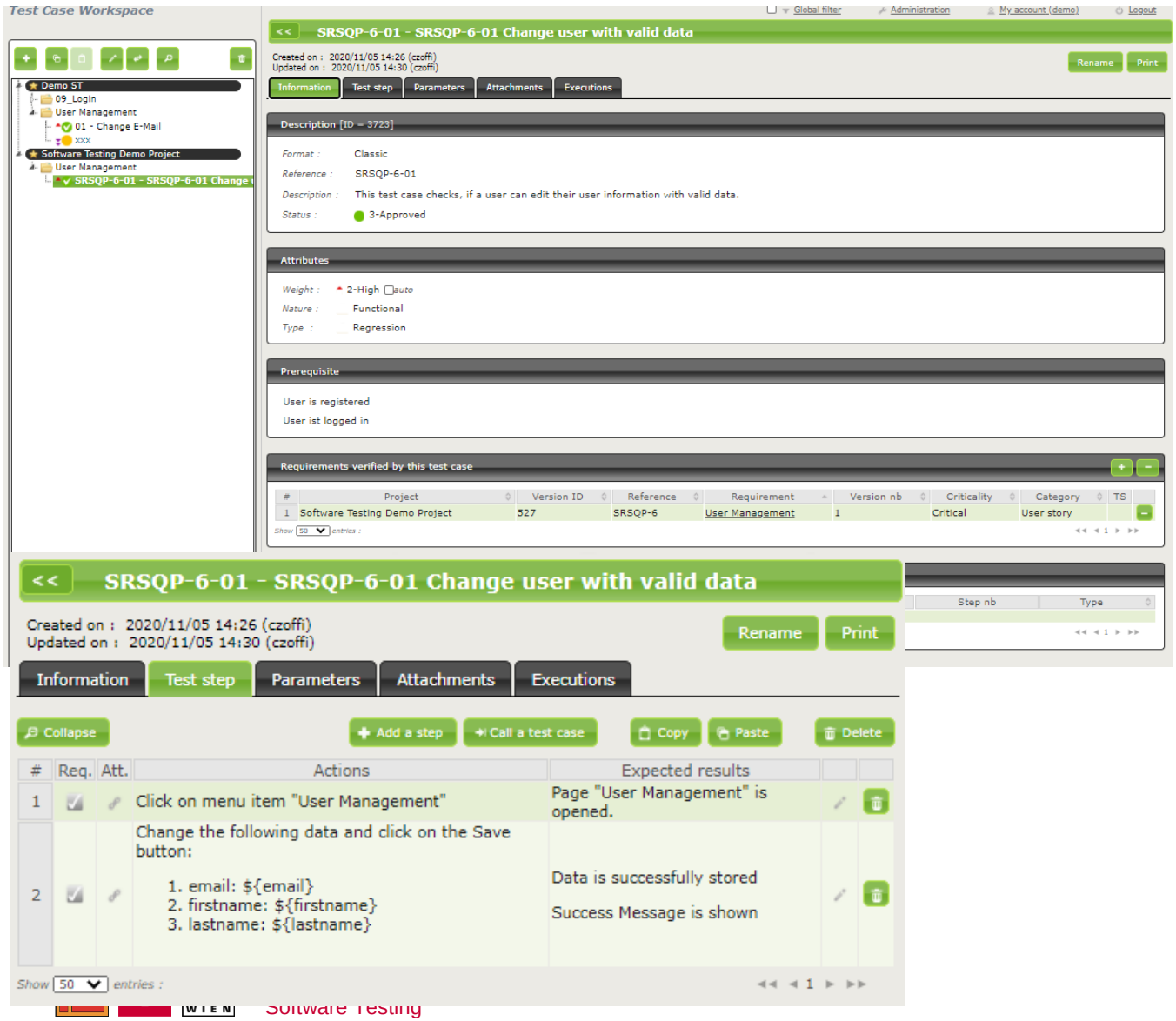
- Folder structure in workspaces
- Create folders and requirements by selecting parent folder and +
- Synchronize existing requirements from issue tracking systems, e.g. JIRA

Squash Test



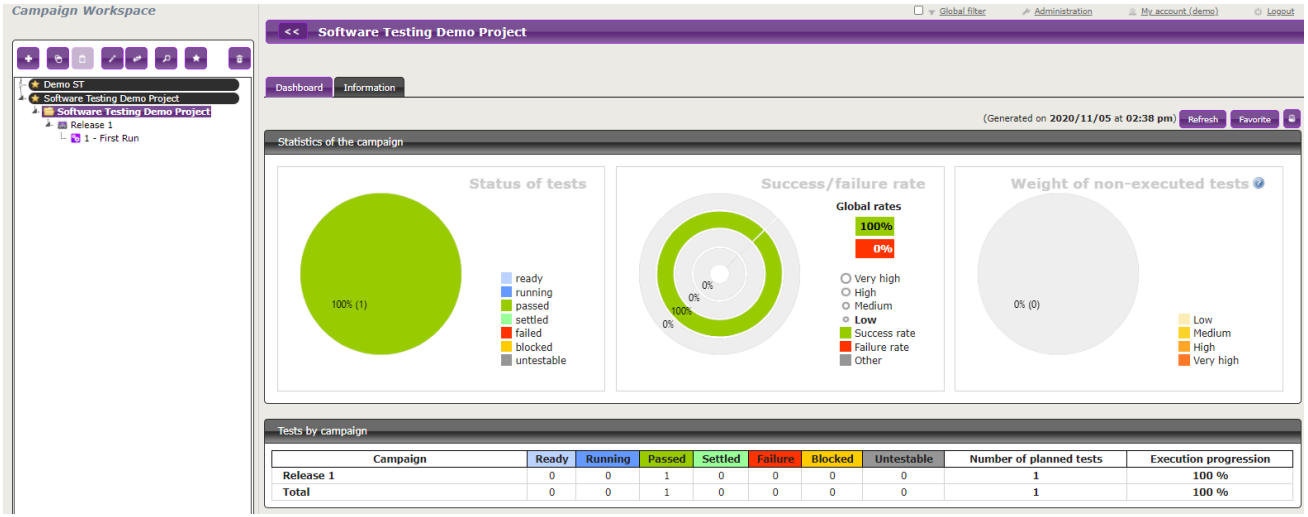
- Visualize the status and progress
- Visualize the coverage of requirements by tests
- Traceability between requirements, test cases and test execution results

Squash Test



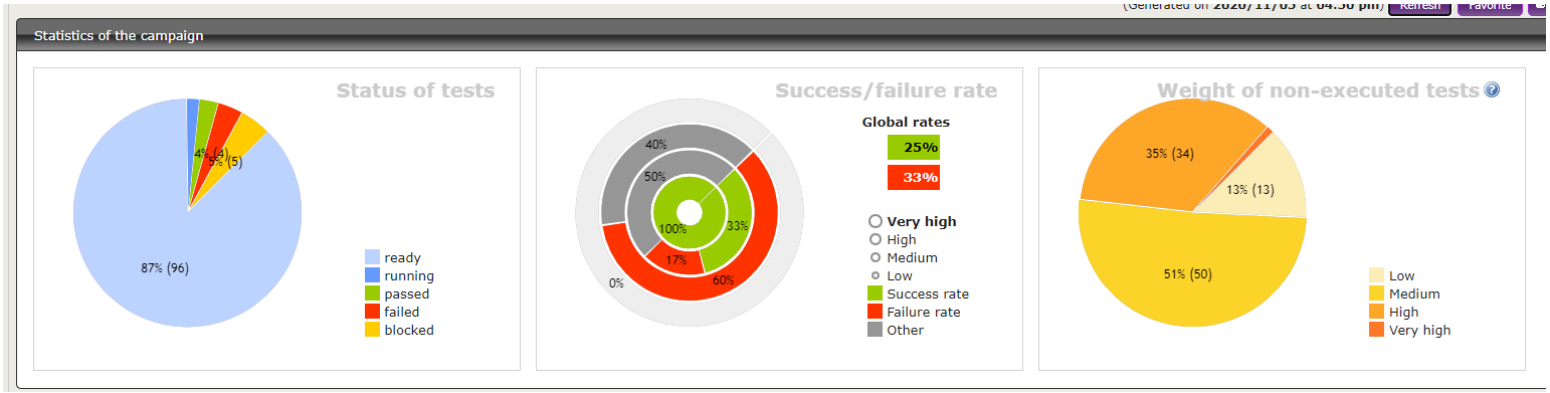
- **Similar creation in test cases workspace:**
  - Information
    - Including link to requírements
  - Test Steps
    - Step by step description
  - Parameters
  - Attachments
    - Files
    - Screenshots
    - …
  - Executions
    - Trace executions
    - results

Squash Test

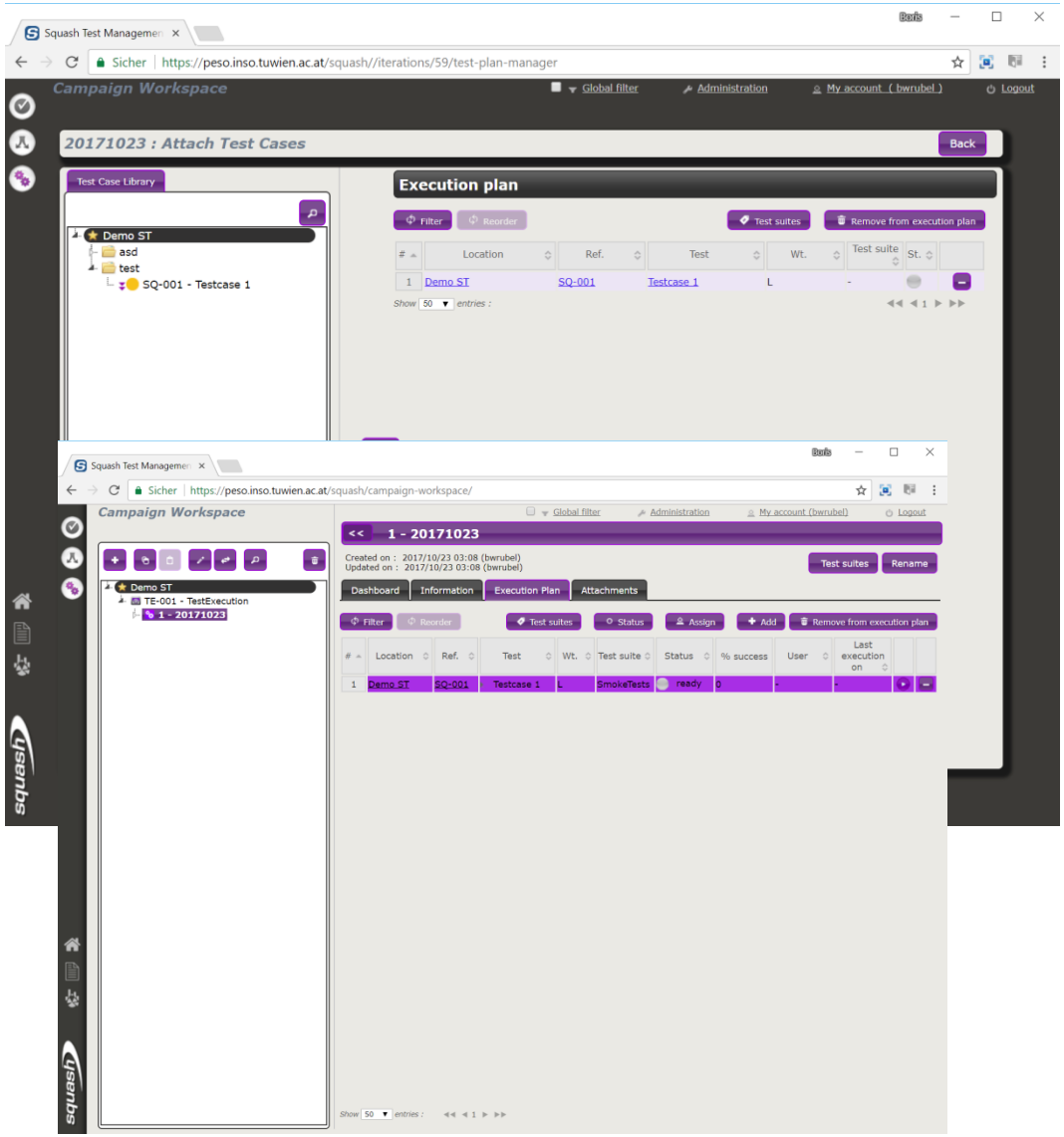


- Before execution of a test case, a campain has to be created

An iteration of a campain can be executed and will have a final status like passed, failed, etc.

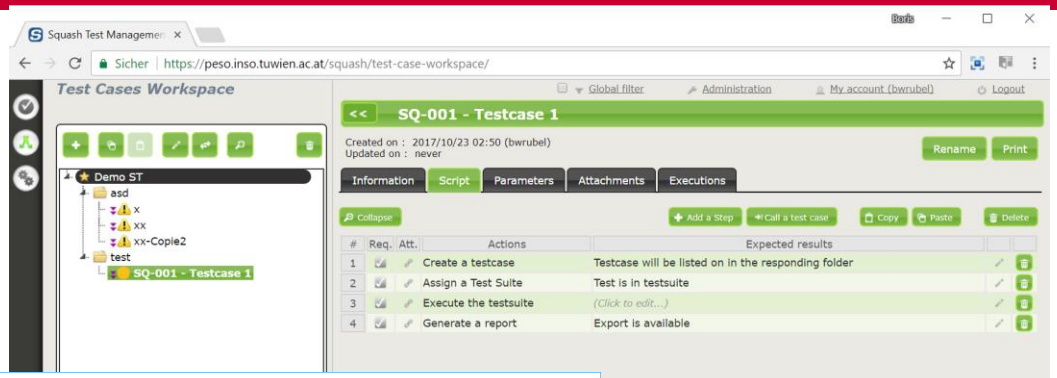


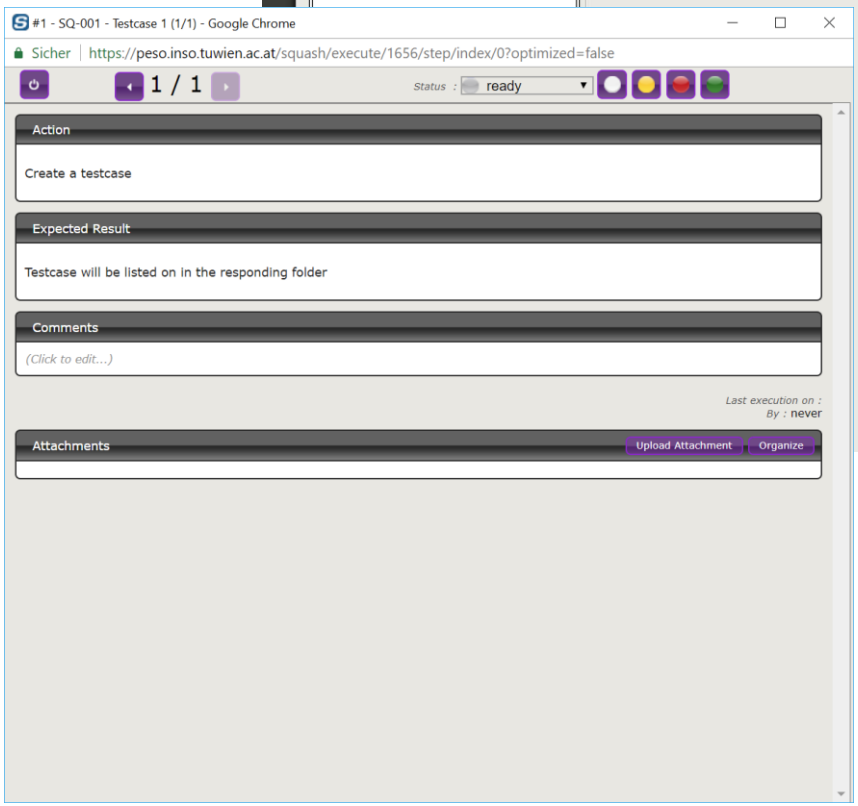
Squash Test



- Add test cases to the execution plan of a campain
- Then „run“ the testcases by clicking the play button

Squash Test





- The execution popup will show every step from the test case script and you can set the status of every step with the colored buttons

Squash Test

- **User guide from Squash TM:**

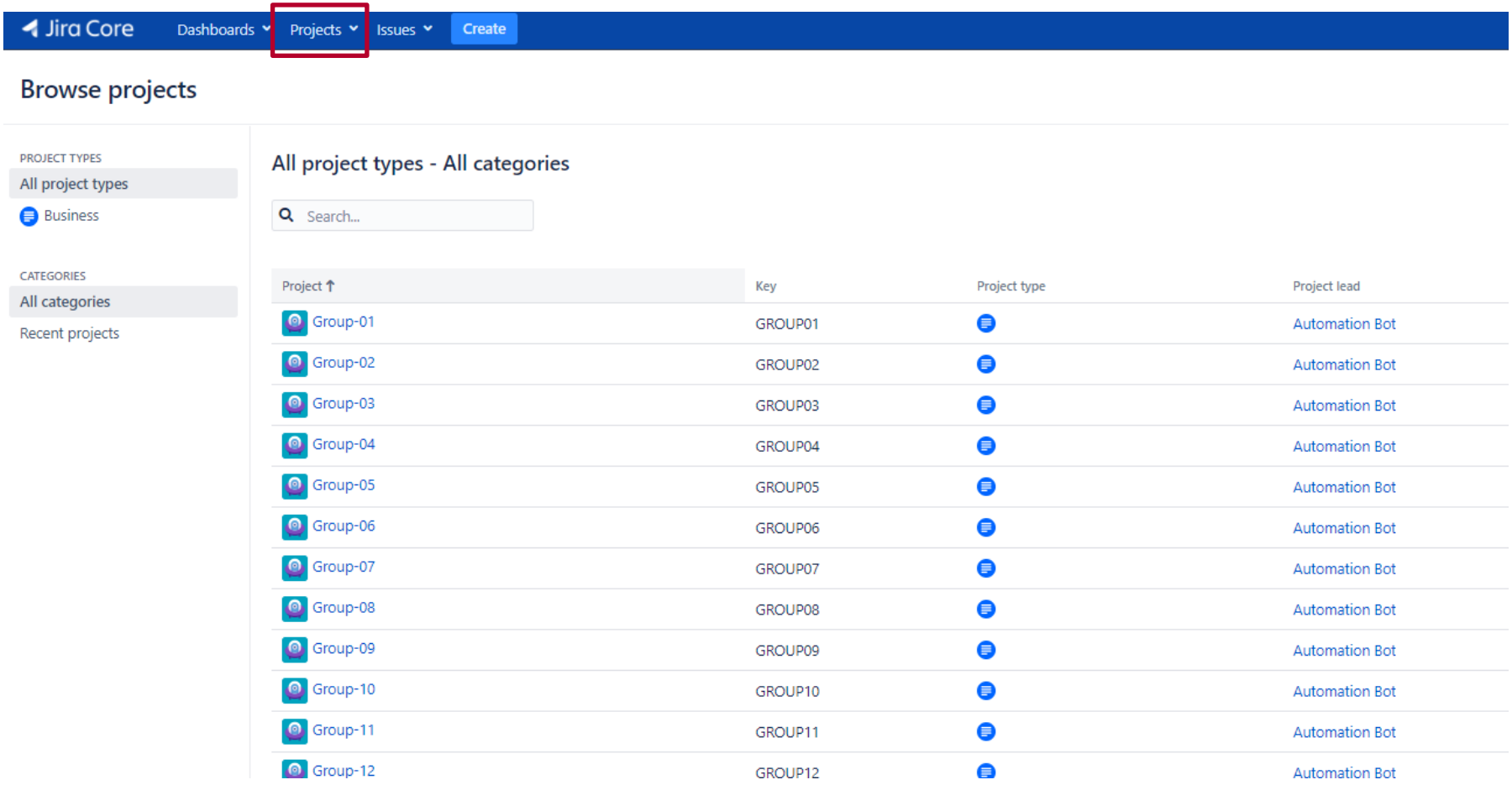
  https://sites.google.com/a/henix.fr/wiki-squash-tm/User

- It is important that you know how to use this tool, try the online – demo to get familiar with the tool **BEFORE** the group labs.

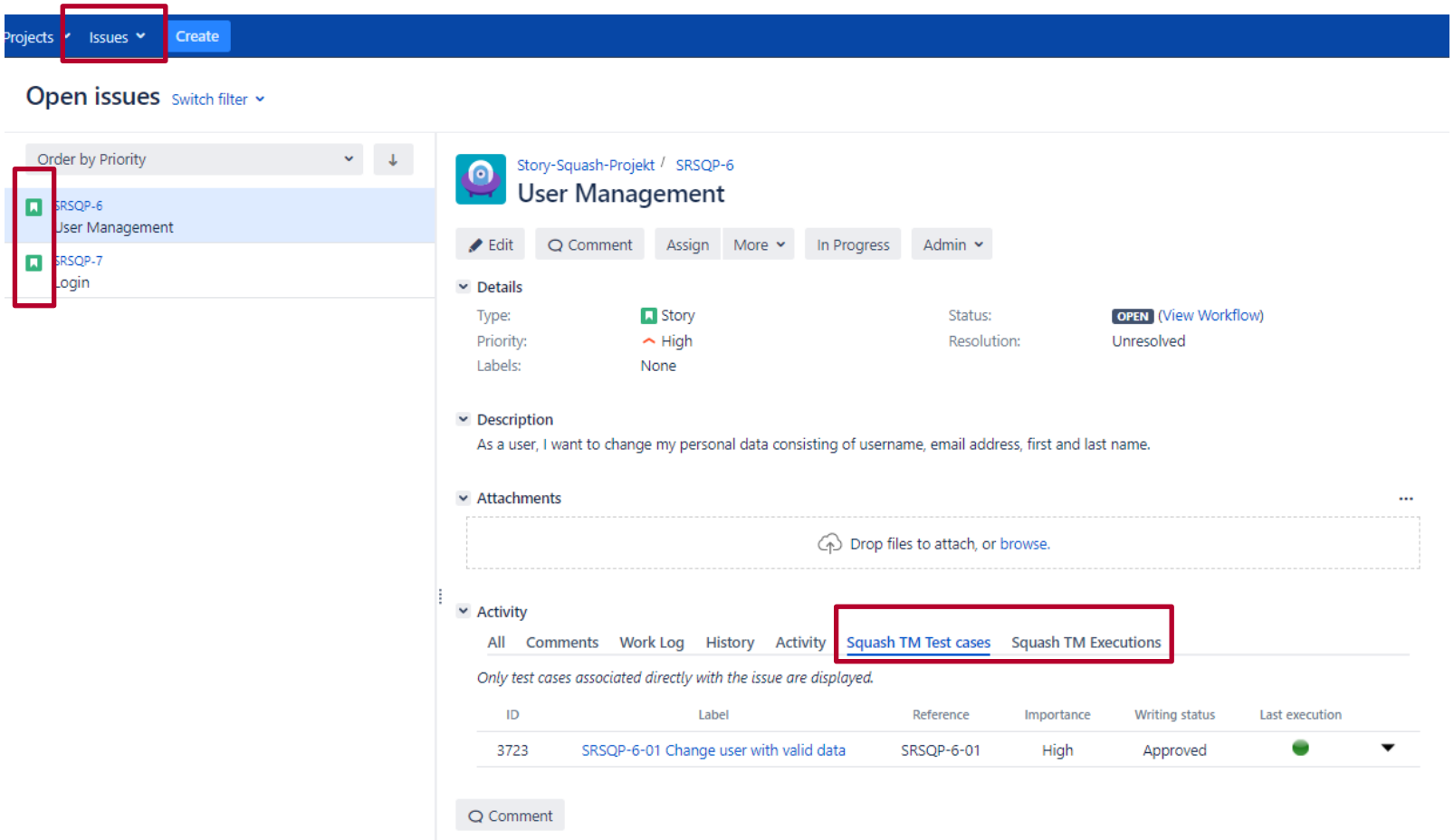
  https://demo.squashtest.org/squash/login

- We recommend to use Squash TM in English version (set browser language to English), since German translation is quite strange

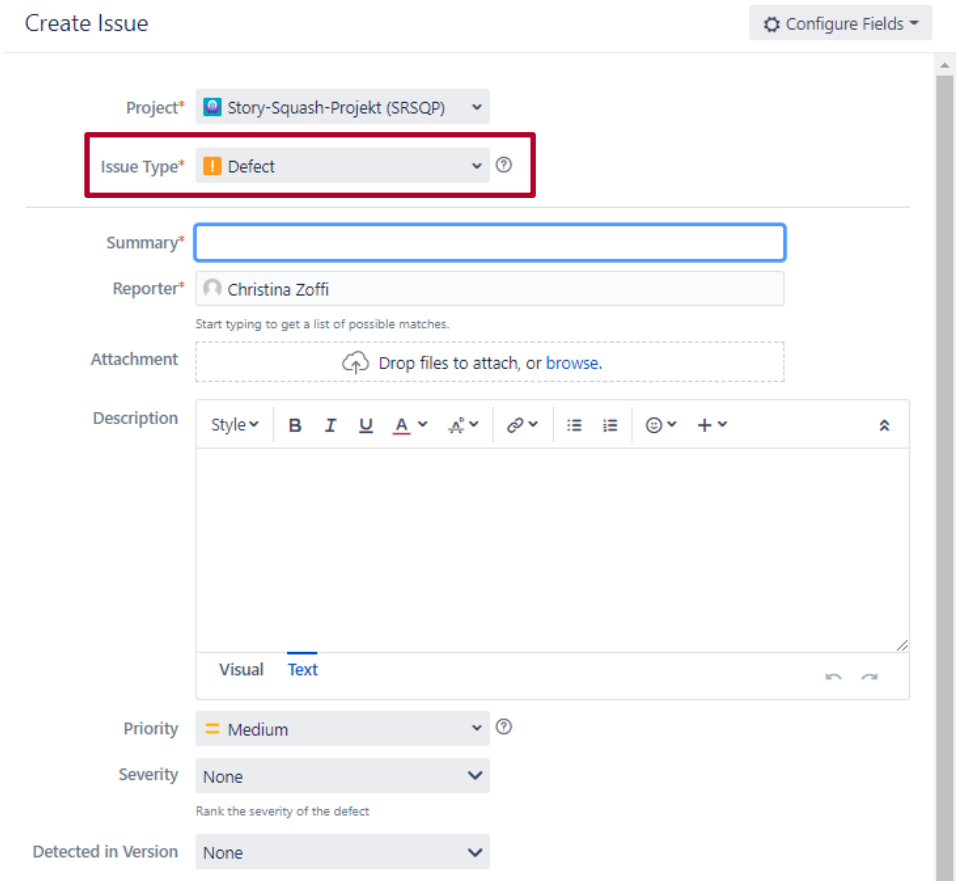
Test Tools – JIRA



Test Tools – JIRA



Test Tools – JIRA



Content

References

- T. Grechenig et al; Softwaretechnik Mit Fallbeispielen aus realen Entwicklungsprojekten, 2010
- ISTQB Foundation Level Syllabus
- A. Spillner and T. Linz; Basiswissen Softwaretest. 5. Aufl. dpunkt Verlag, 2012.
- D. Graham et al; Foundations of Software Testing: ISTQB Certification. Cengage Learning EMEA, 2008.
- IEEE Std 610.12-1990 (R2002), IEEE Standard Glossary of Software Engineering Terminology IEEE, 1990.
- A. Spillner et al; Praxiswissen Softwaretest – Testmanagement: Aus- und Weiterbildung zum Certified Tester, 2014