TestResultWebApp

v. 0.1.0

Thomas Pollerspöck

04.07.2022

Contents

Chapter 1

Introduction

Under documenting

Chapter 2

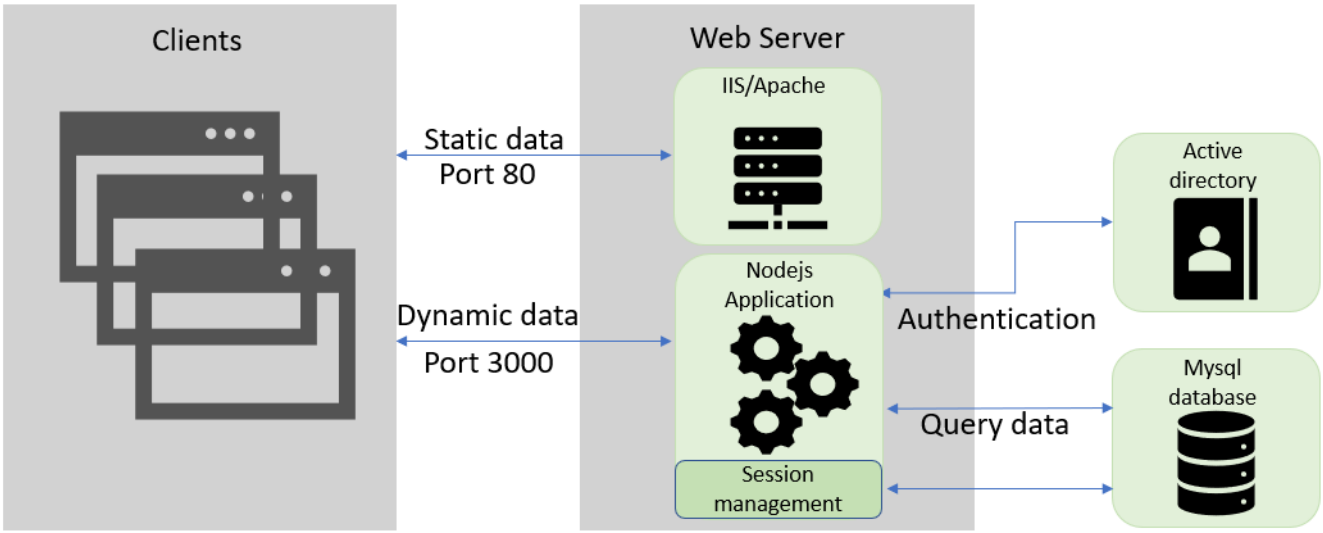
Description

2.1 WebApp Architecture:

The WebApp application includes:

- The database: mysql is used - which contains on the test execution results. For detail of all tables in databse, please refer the database model.
- Active Directory: for the authentication.
- Web server: which is hosted for static data and run the nodejs application for the dynamic data (API call).

Please refer below architecture for more detail:



2.2 Import Data:

The data which will be visualized on webapp are get directly from the database. So that the test execution result information should be transformed first follows the defined database model then import to database.

We have already provided the RobotResults2DB which helps to import the Robot Framework result file(s) *output.xml* to the database of testresult webapp.

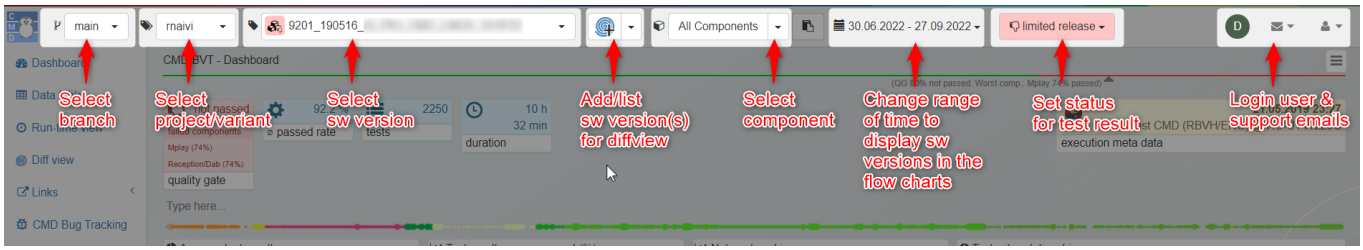
You will need to provide only the Robot Framework result file(s) and database's information, that import tool will parse the test execution result information and interact with provided database to import the according data.

Please refer RobotResults2DB's usage and its document for more detail.

2.3 Data Visualization:

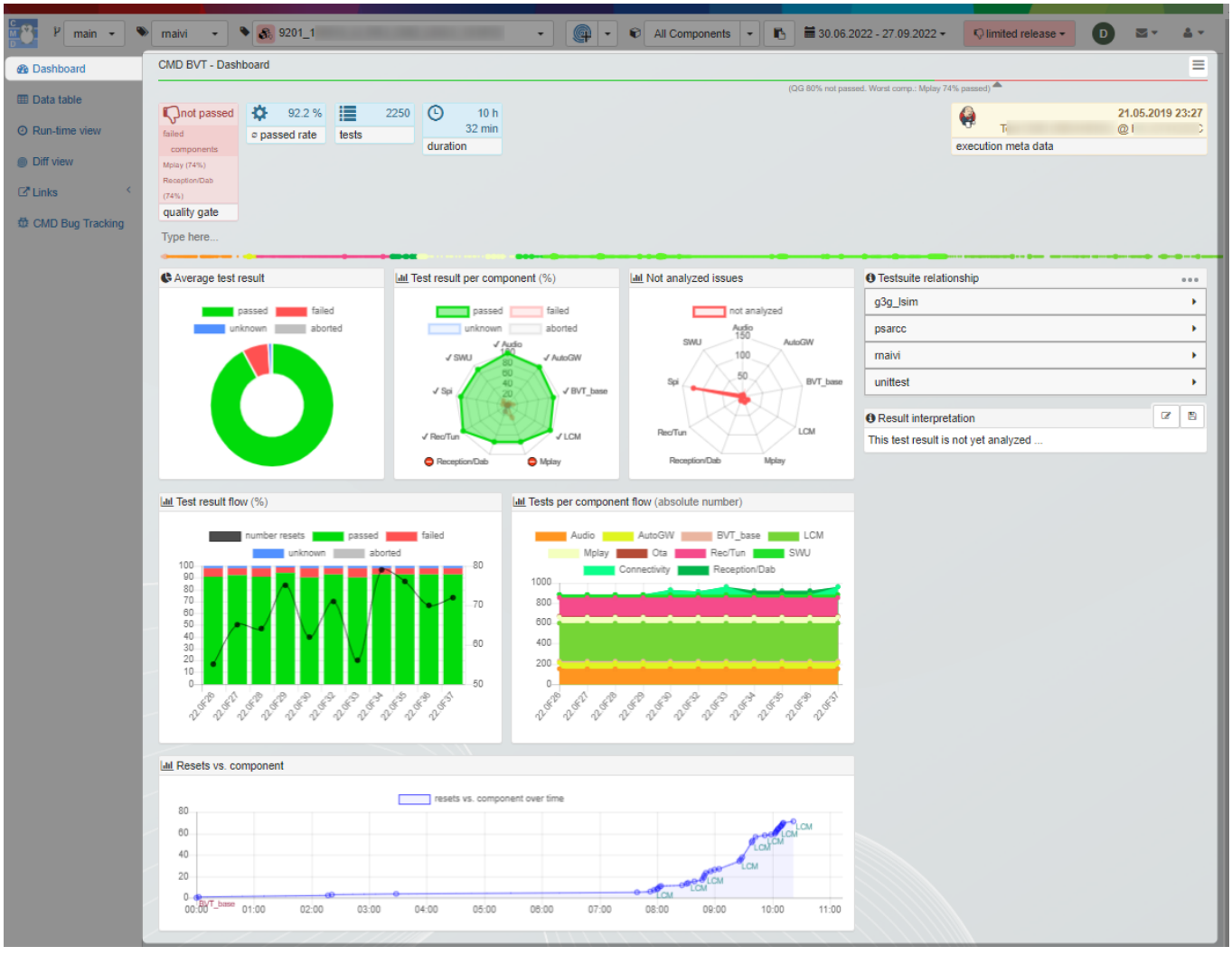
2.3.1 Main menu:

WebApp provides the main menu for user to select the specific branch, project/variant and software version to be displayed.



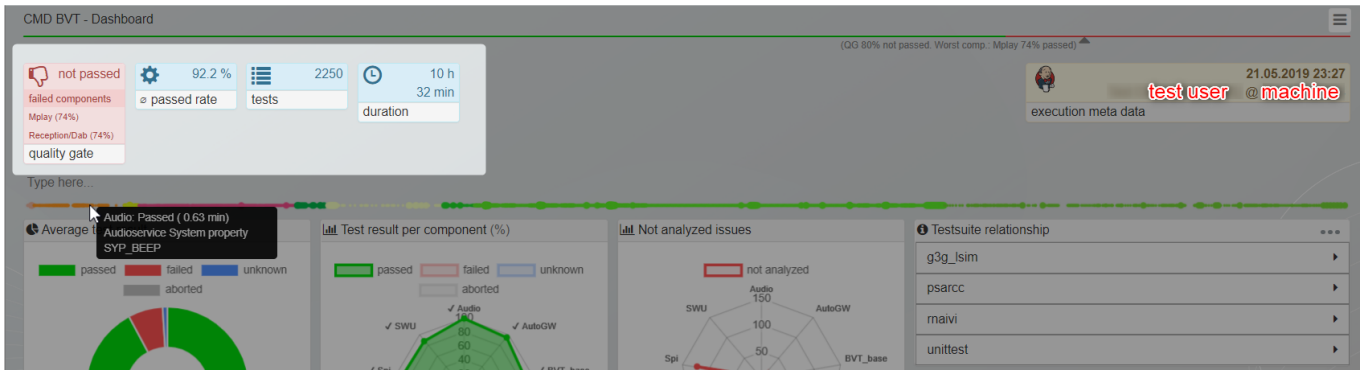
2.3.2 Dashboard View:

Dashboard view does not only provide an overview of test execution results but also shows the correlation betweeen components within the result and relationship with other test execution results.



Result overview:

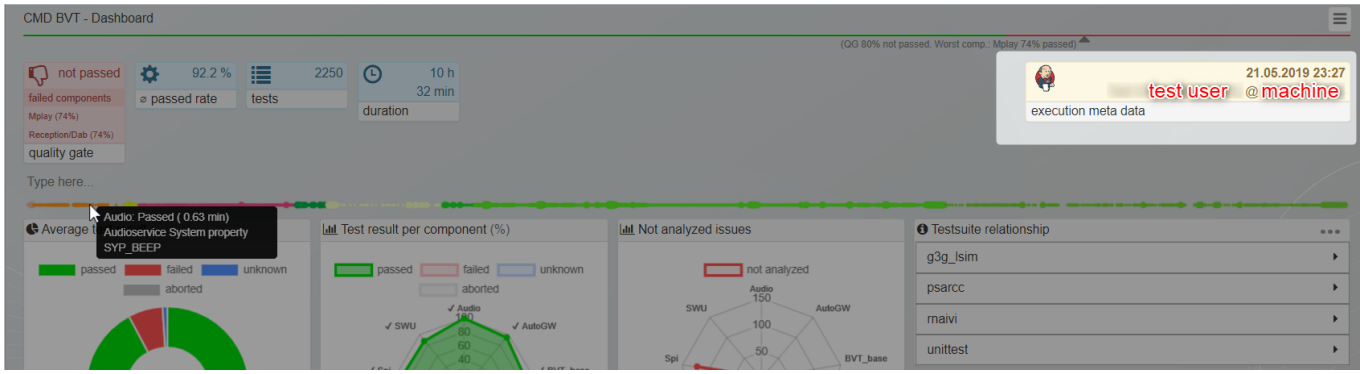
On the top left conner of the Dashboard view, there is test execution result statistics:



Which contains:

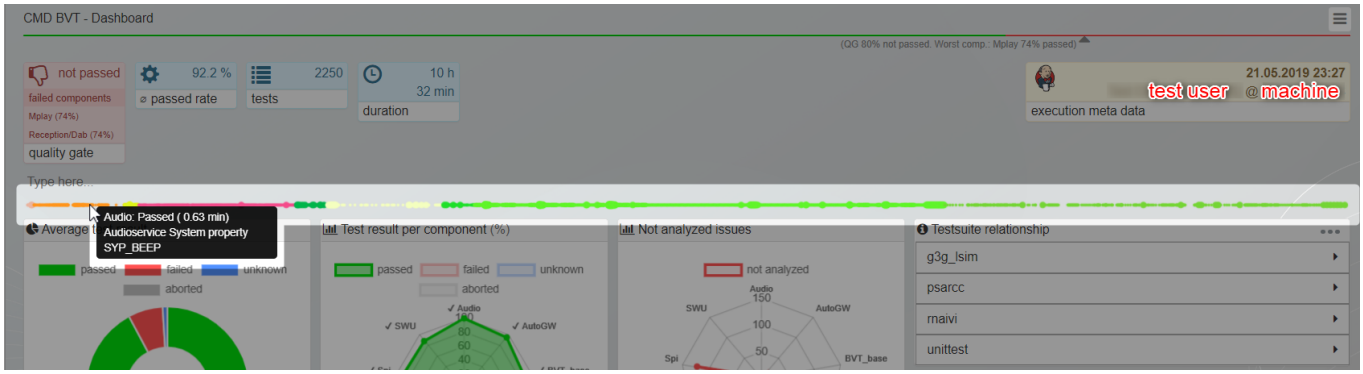
- Overrall status.
- Passed rate.
- Total executed test cases.
- Test execution duration.

On other right-hand side, there is information about execution environemnt:



- Execution time
- Test machine
- Test user
- Jenkins link (embedded URL in the Jenkins's icon)

Below them is the test execution result timeline:



It provides:

- The timeline of the executed test cases by components (different color).
- How much time is consumed by the individual test case by the distance to the next test on time line or the detail pop-up when hovering on the dot of the timeline.
- Test status result: small dot for passed status and big dot for others.

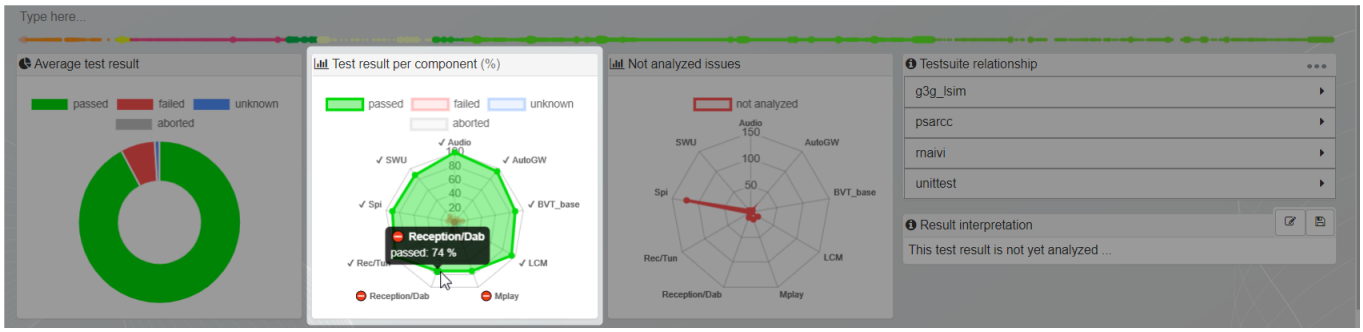
The next **Average test result** chart will give you the detail of test result with the percentage (number of test cases will be showed when mousing over the pie chart) of each result status (Passed, Failed, Unknown or Aborted) in the execution.

So that, you can qualify this test execution result is good or not.

Component's correlation:

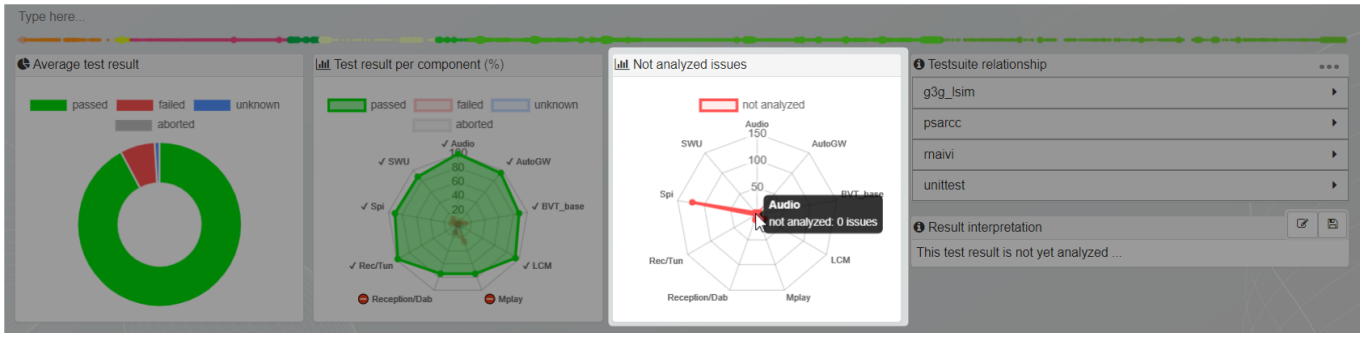
The next charts will help you to get the correlation between components within the test result, so that you can know which component(s) is impacted to the test result.

- **Test result per component** chart: provides the passed percentages of all components within test results and the quality of them compare the expectated quality gate (default: 80%).
So that, you can justify the impact of component to the whole test execution result.



- **Not analyzed issues** chart: you can known how many test cases of components are issued without any analysis.

So that, you can have the appropriate actions to the component team.



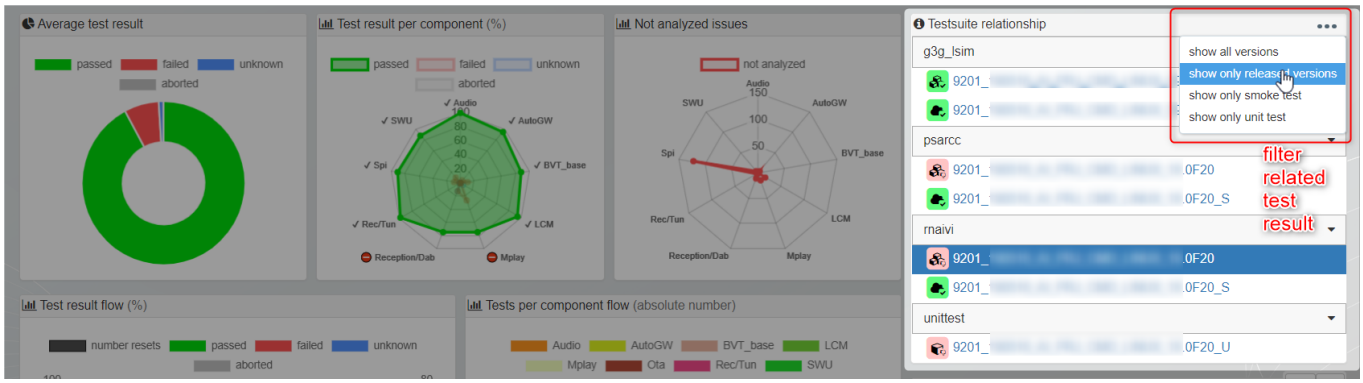
Relationships with other test execution results:

- **Testsuite relationship**: will let you know all the related test results (grouped by project/variant) of the the current selected version.

So that you can quickly go to the related test results to have the comparison about the quality of the selected version across the projects/variants.

For example: the selected version have been executed for 4 projects/variants: *g3g_lsim*, *psarcc*, *rnaivi* and *unittest*. Each variant (except the *unittest*) has 2 test results (one for the smoke test and one for the whole test execution result).

Then, all the related test execution results will be displayed as below:

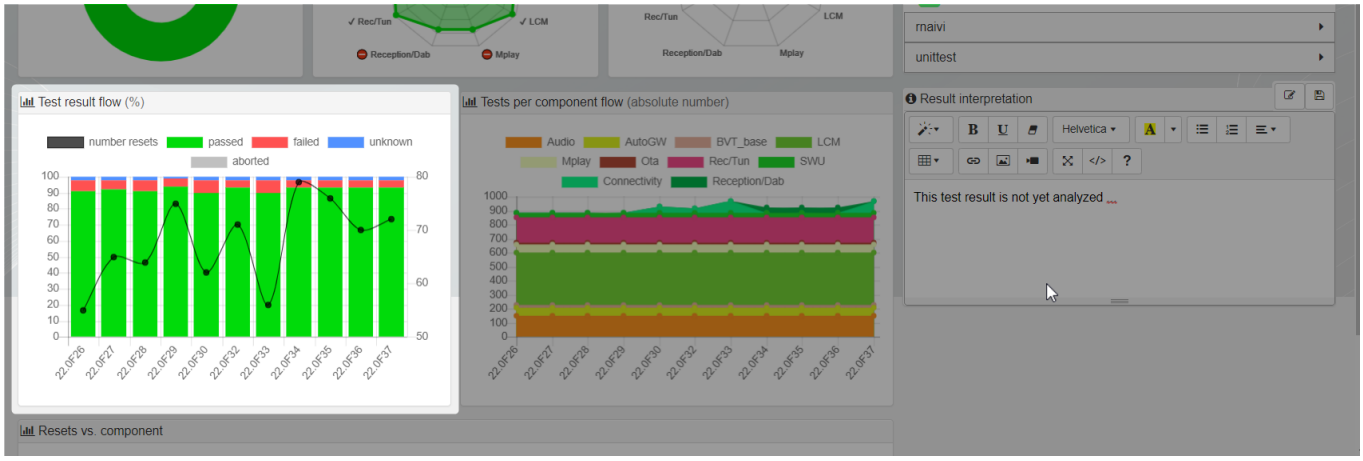


There is the context menu (**...**) that allow you to filter the related test results.

- **Test result flow** chart: provides the picture of quality change (percentages of each status) between versions.

So that, you can understand the quality of testing software is being improved or vice versa.

> **Notice**
>
> The versions which are displayed in **Test result flow** chart are the executed test results within the selected range of time in the main menu (default is **Last 90 Days**).

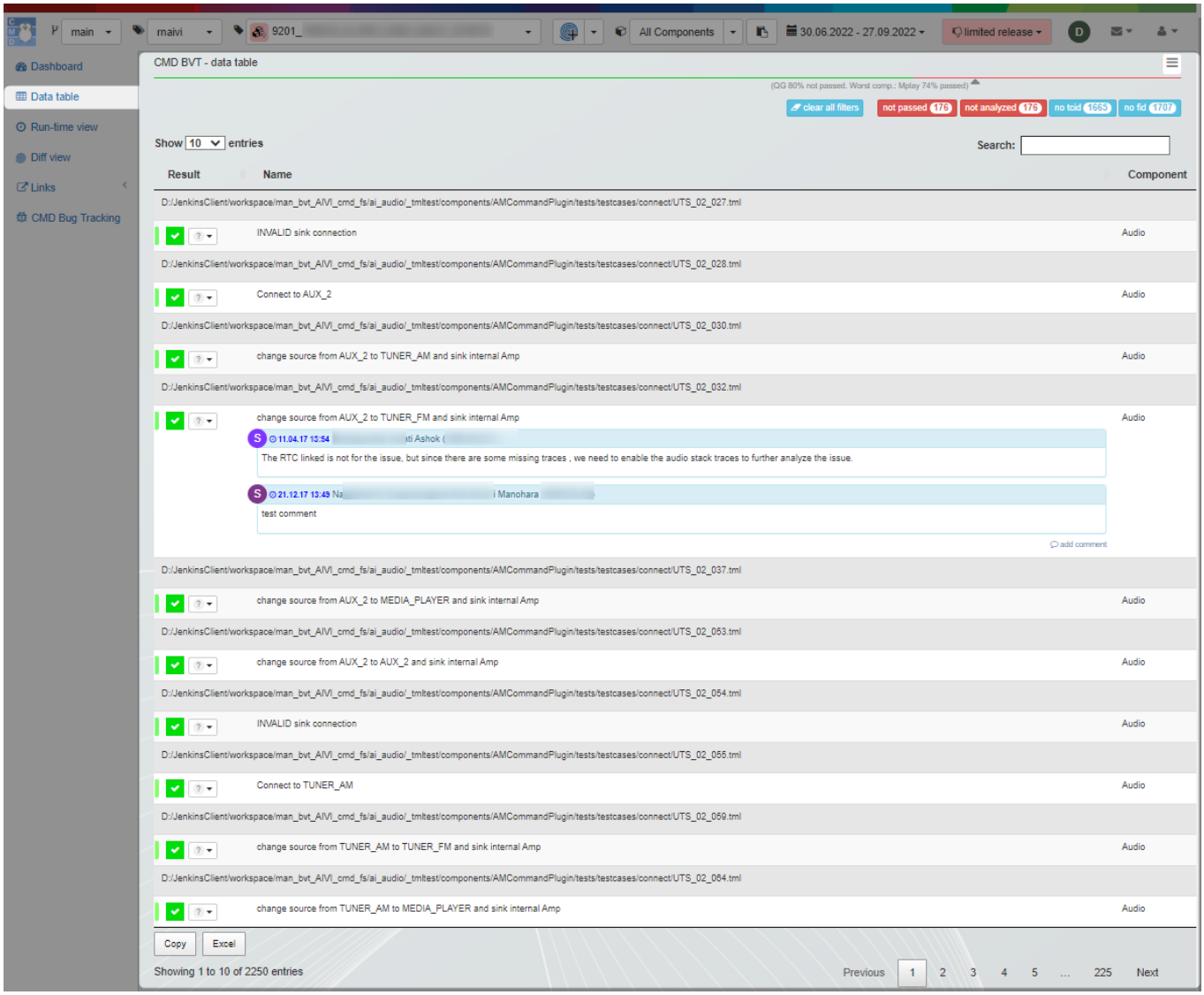


- **Tests per component flow** chart: provides the change of number test cases per component between versions.

You can aware the number of test cases per component and how many test cases are added or removed (per component or the whole test result) from those versions.

> **Notice**
>
> The versions which are displayed in **Tests per component flow** chart are the executed test results within the selected range of time in the main menu (default is **Last 90 Days**).

2.3.3 DataTable View

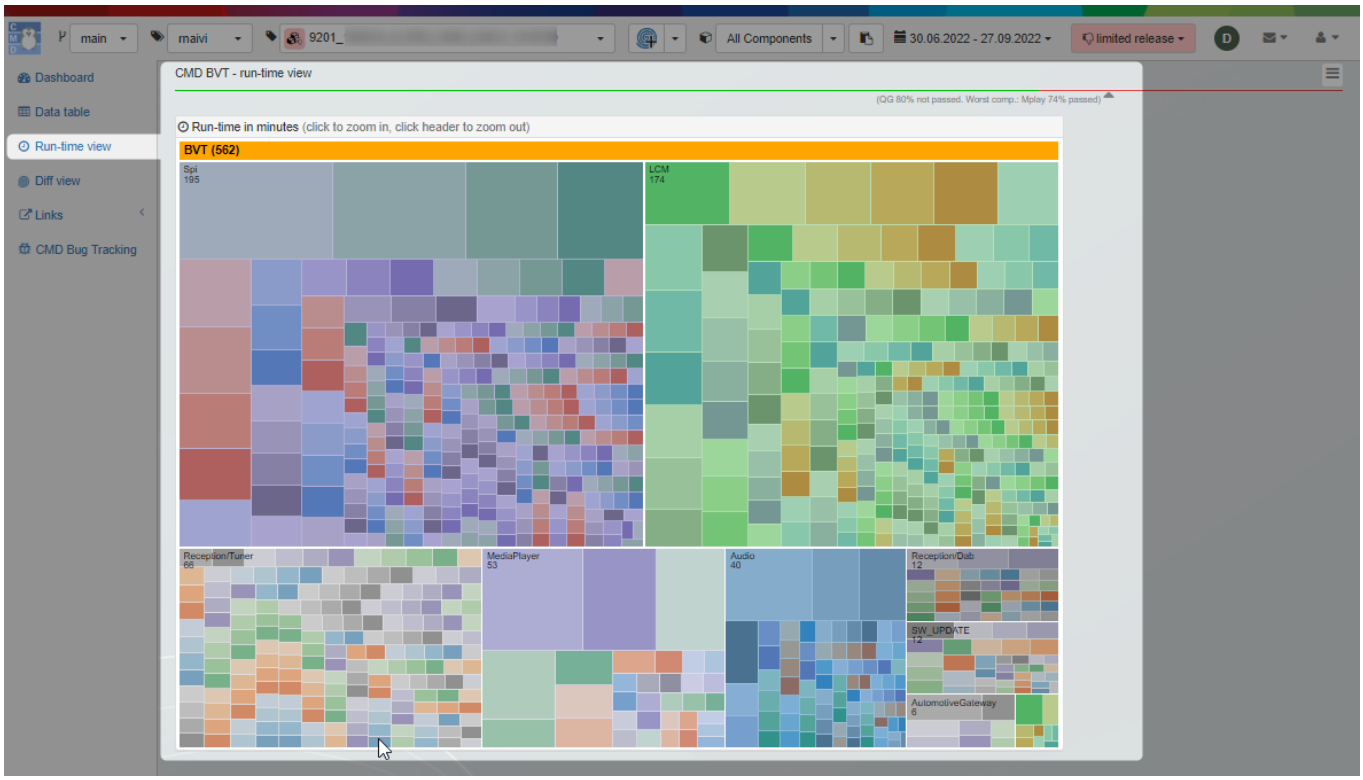


2.3.4 Runtime View

The runtime provides the treemap chart which helps you to know the time consumption of components within test execution result or test cases within each component.

You can click on the component to go to detail run-time of all test cases within it and go back by clicking the component header.

With this view, you can understand run-time of a your test suite then optimize the specific component or testcase if required.

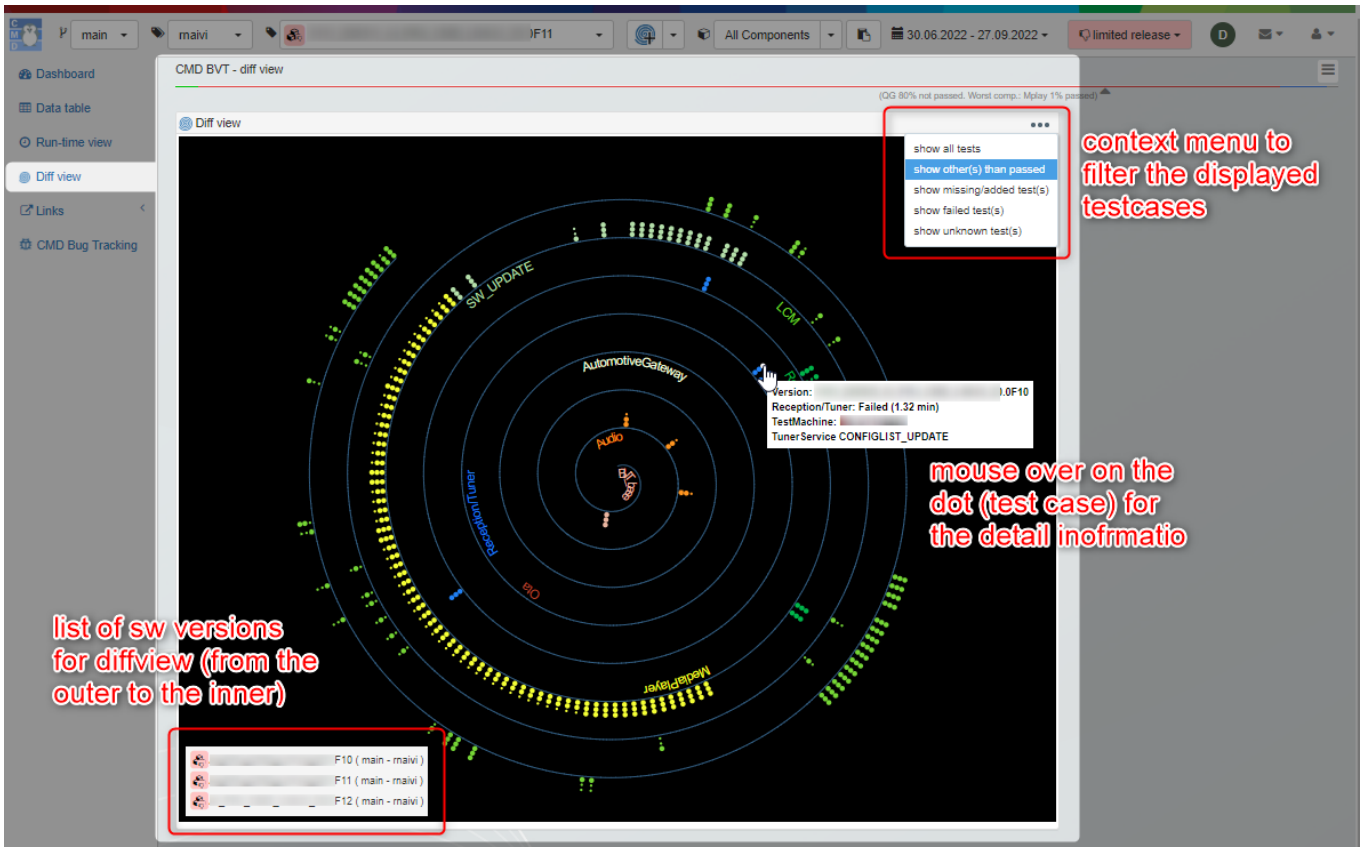


2.3.5 Diff View

The diffview contains only the spiral chart which displays the differences between the software versions you want to compare. So that you can:

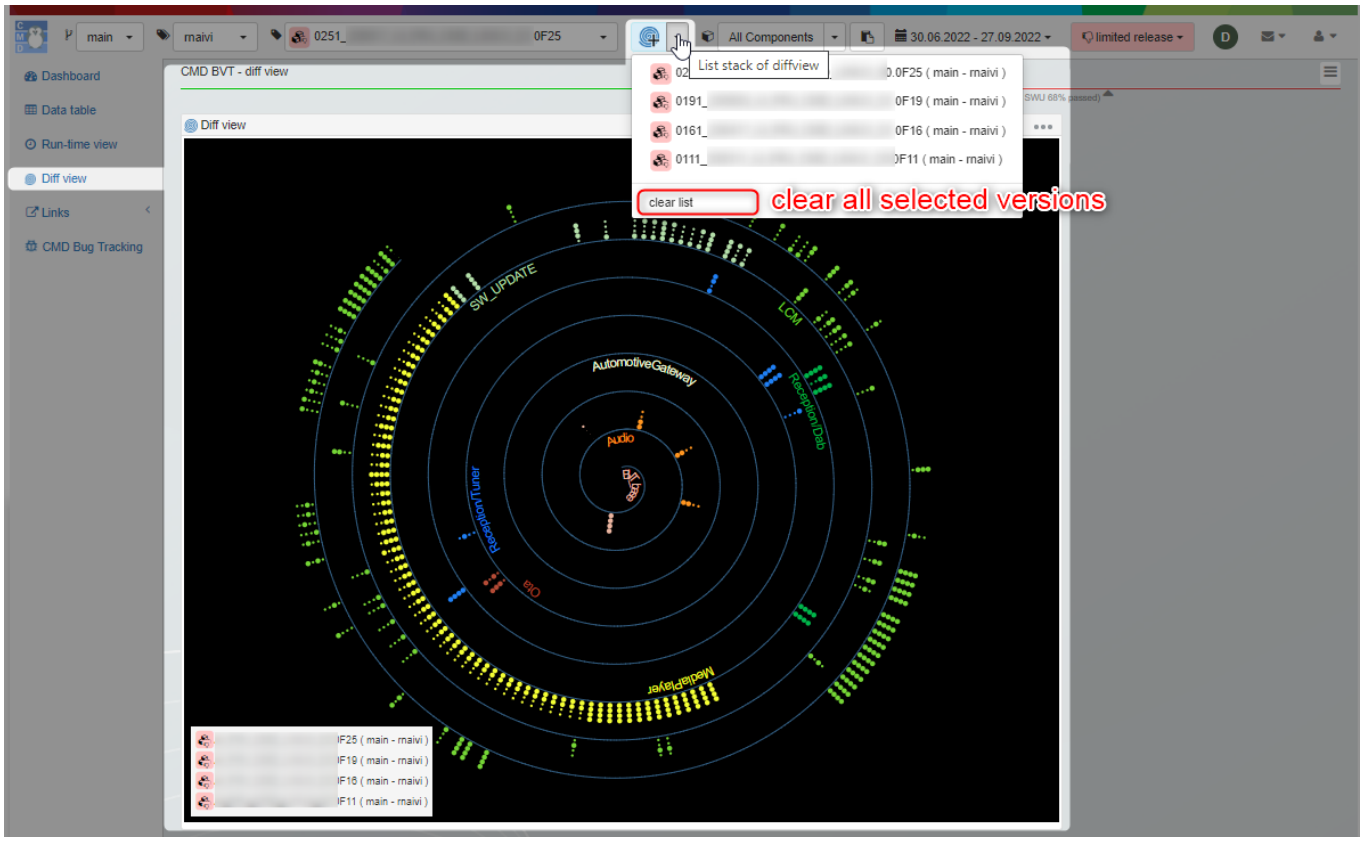
- observe the test case result change (e.g from passed to failed).
- aware the new or removed test case(s).
- recognize the unstable test case(s)/component(s).

Be default, without adding the version for diffview, the current selected version and its around versions (the previous and the next version if existing) will be chosen for diffview as below:



In case, you want to add the specific versions for diffview, select the version from the version select box then click the add button  to add it to the list of diffview.

The next dropdown button is used for viewing your selected version. You can also clear your selection with **clear list** option.



As soon as the new version is added for diffview, the spiral chart is updated immediately.

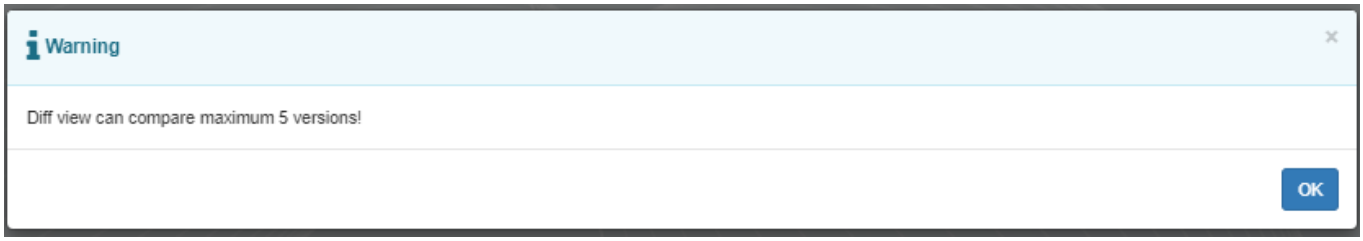
The dots which present for the test cases along the spiral line (small dot is passed and bigger one for other status)

are interactable. It means that you can:

- mouse over the dots to see the test case information.
- click on the failed test case (bigger dot) for the traceback informtion.

Notice:

You can only select maximum of **5** versions for diffview.
If the maximum of selected versions is reached and you click on the button to add more, the warning message will be displayed to prevent that action.



Chapter 3

Appendix

About this package:

Table 3.1: Package setup

Setup parameter	Value
Name	TestResultWebApp
Version	0.1.0
Date	04.07.2022
Description	Web based display of test results
Package URL	testresultwebapp
Author	Thomas Pollerspöck
Email	Thomas.Pollerspoeck@de.bosch.com
Language	Programming Language :: Python :: 3
License	License :: OSI Approved :: Apache Software License
OS	Operating System :: OS Independent
Python required	>=3.0
Development status	Development Status :: 4 - Beta
Intended audience	Intended Audience :: Developers
Topic	Topic :: Software Development

Chapter 4

History

0.1.0	07/2022
Initial version	