MES QUALITY COMMANDER User Guide

Release 7.5

Model Engineering Solutions GmbH

Jul 03, 2024

CONTENTS

1 MOTIVATION

1.1 WHAT IS MQC AND HOW CAN IT HELP YOU?
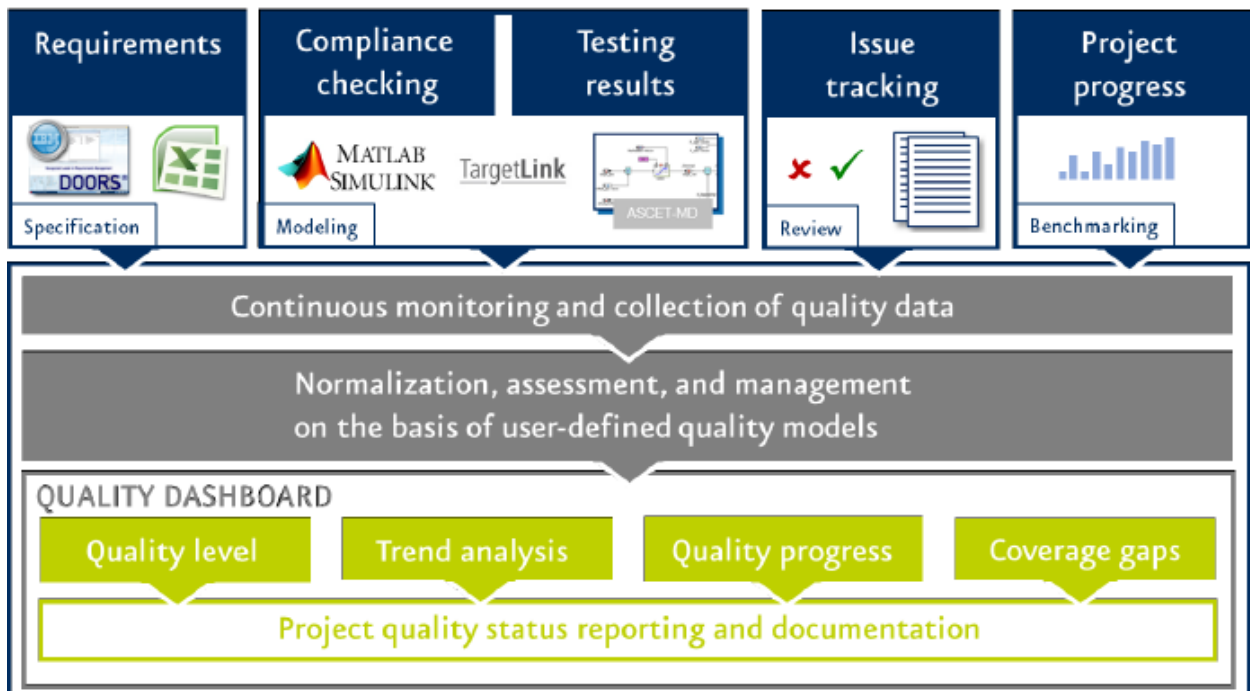
1.1.1 What is MQC?

Figure 1.1: Quality assurance and monitoring with MQC

MES Quality Commander® (MQC) is a dynamic quality monitoring and management tool for software development that captures all the decision-making data that you need throughout the software life cycle. MQC computes and evaluates the quality and product viability of your software, based on relevant development artifacts and the corresponding key performance indicators (e.g. guidelines, complexity, tests, coverage and reviews). User-friendly visualizations of product maturity, weaknesses and need for action during the different stages of a project increase the software's development and product value.

MQC also optimizes return on investment by perpetual availability of trend analysis that indicates the product's achievable level of quality. An efficient visualization of quality and progress for different development projects ensures error proofing very early. Project-specific evaluation with individually configurable quality

models adaptable to ISO 26262 or ASPICE enables quality assurance of safety relevant software development.

MQC provides different possibilities of reporting such as the desktop client itself or the web viewer for sharing information. Thus, effort and changes can be controlled and minimized. The web viewer guarantees multiple users to be able to access your project and supplies interactive reporting along with other features. Therefore, data discovery and operational reporting yield an entire understanding of the data's quality impact. MQC data import supports several operational tools and export formats, which allows a fast and easy setup of quality monitoring projects. Data collection can also be automated and integrated into continuous integration to make full use of your existing infrastructure and workflows.
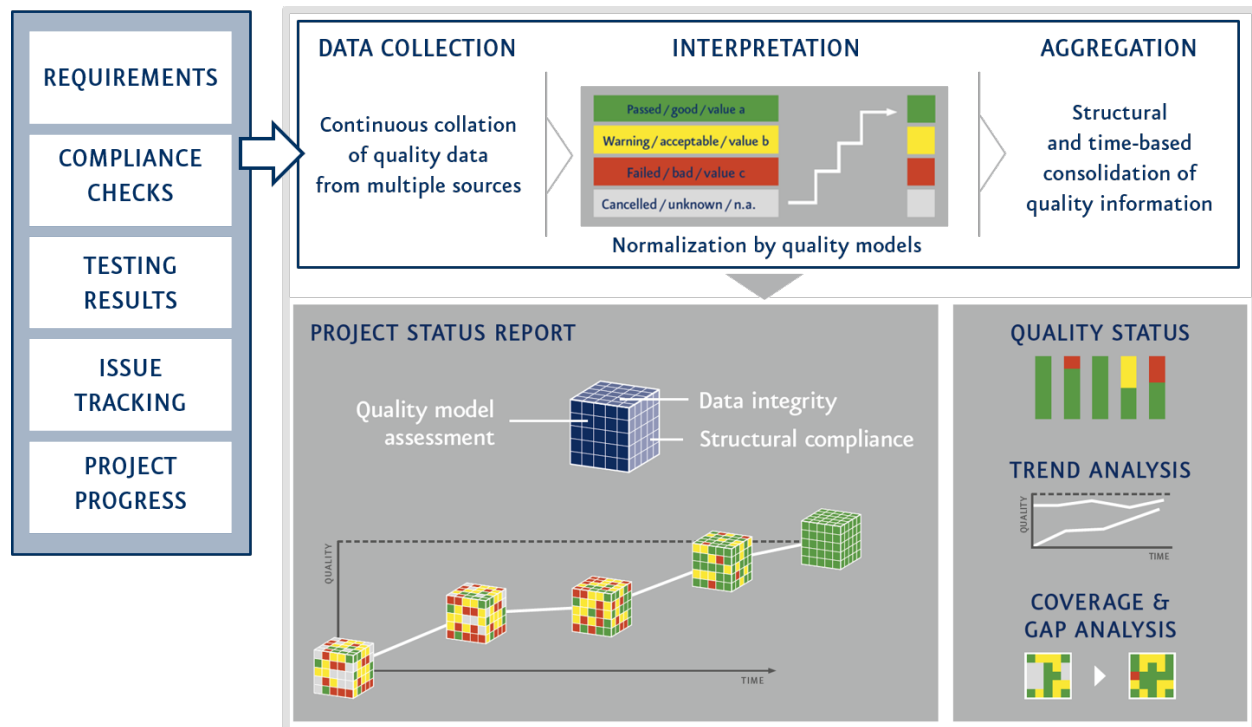

Figure 1.2: Data to quality: MQC's representation of the Software life cycle

Referring to the V-Model, MQC aims to be the Master tool of monitoring and managing your (model-based) Software Development Lifecycle (SDLC). As shown in Figure 1.3, MQC can collect data from various quality assurance activities at each stage of the SDLC. MQC not only provides efficient extraction of data from MES tools, but also from other tools like TPT, Tessy, Polyspace and Embedded Tester.

1.1.2 From data to quality

The project creation workflow can be separated into two parts.

The first and most important part to create an MQC project is "Data and project structure". It contains two processes: "Analysis" and "Creation". During the "Analysis" process, one has to analyze the available data with respect to the data's relevance of quality. In particular, the "7Ws" of data analysis (who, where, when, what, why, how and how many) are fruitful to answer to understand the data's structure. For more details,
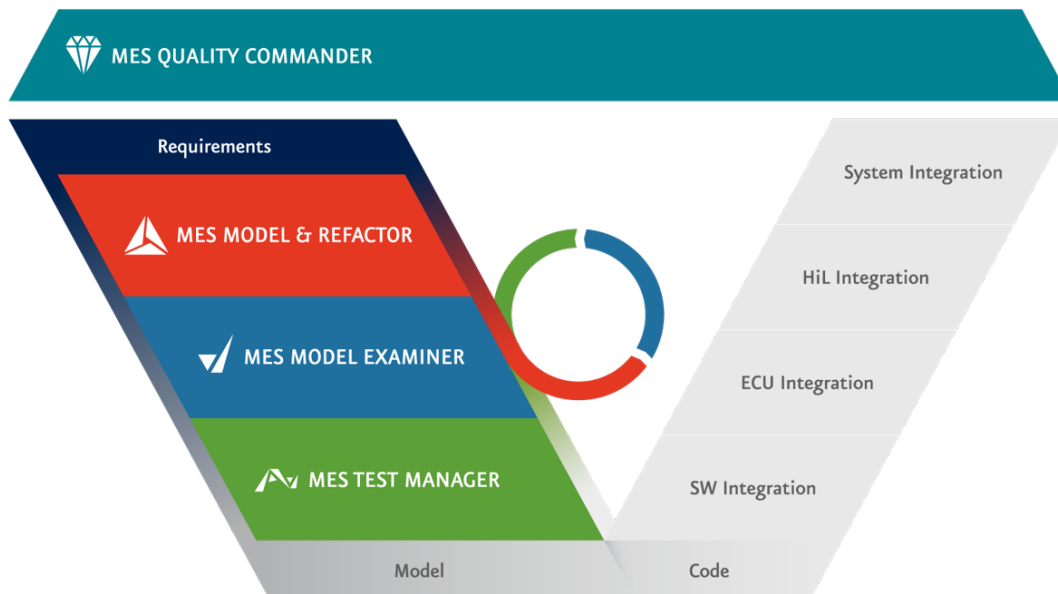
Figure 1.3: MQC evaluates data from a wide range of report-generating tools.

we refer the user to read the book "Agile Data Warehouse Design" (Lawrence Corr with Jim Stagnitto, 2014, pp. 31f).

After having answered these questions, you should have identified relevant data. The next crucial step is the identification of MQC structures in your data. Such structures can be artifacts and hierarchies, data sources and data values, projects and milestones and much more.

After identifying the structures in your data, they need to be mapped onto the MQC structures. Particularly, this is a very abstract step, because it is a priori vague what "mapping" means in detail. Typically, MES quality and data experts, together with customer process experts, analyze the customer process and provide professional support to deduce MQC structures. All these steps form the foundation of the second process – "Creation".

First, note that the order of the sub-processes of "Creation" as depicted in Figure 1.4 may be adapted, but we recommend the given sequence. Since "Artifacts" are those objects for which data arises and for which MQC does quality computation, it is a good idea to start with structuring these objects first. This structuring is an outcome of the "Analysis" process. Secondly, you should define the data source structure, which follows directly from the first process, too. As data sources consider the objects that yield data for the Artifacts, it is natural to create them after creating the Artifact structure.

The next step is creating context categories. By a context category, one is able to connect Artifacts and Data Sources. Vividly spoken, a context category provides information about which data is expected for the Artifacts.

The definition of the project's time structure (so-called Revisions) is crucial to provide a chronological sequence to the changes of data.
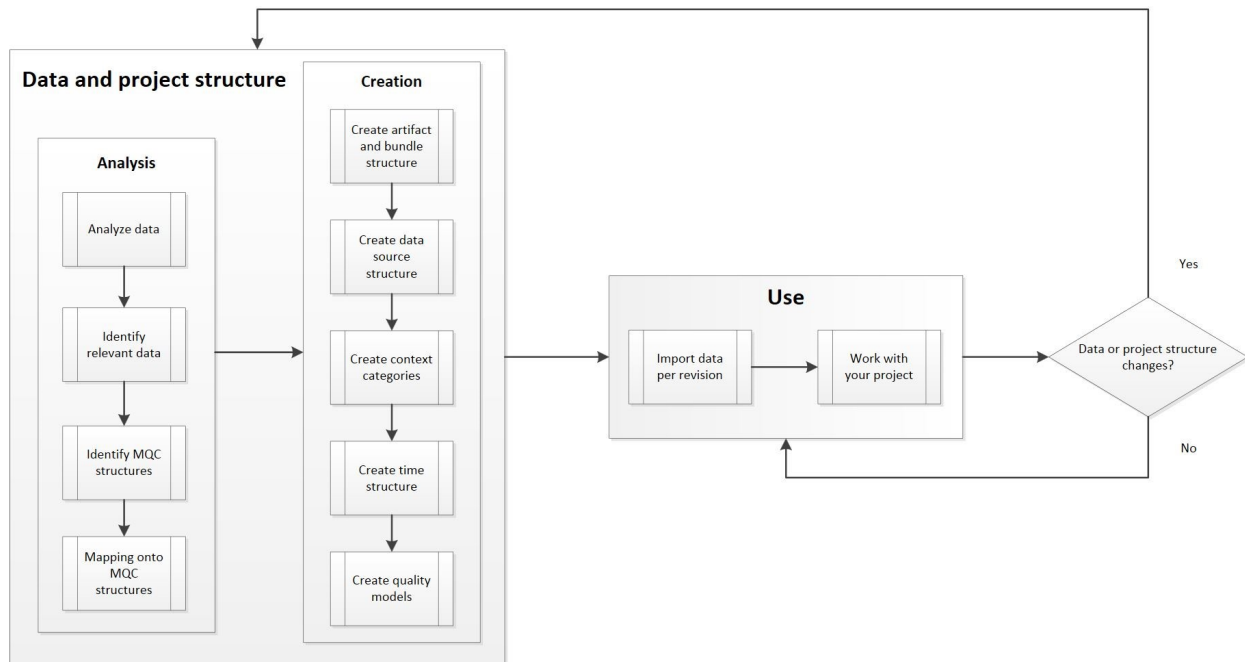
Figure 1.4: MQC Project creation workflow

The second part of the project creation workflow, "Use", is the application of the Data and project structure creation. It consists of (possibly automated) recurring data import, which is the basis for visualizations in MQC. After having imported data, you can work with the visualizations and perform data discovery. If the data or project structure changes, you have to adapt them before importing new data. This entire part deals with quality computation and is a direct result of the previous part. The visualizations appear automatically.

1.1.3 How MQC supports quality assurance and quality improvement

The following illustration provides a coherence of high-level requirements, quality computation and aggregation, as well as (product) quality.

The grey box in the middle of Figure 1.5 symbolizes that the main purpose of MQC is to increase the (software) product quality.

1.2 HIGHLIGHTS IN MES QUALITY COMMANDER® (MQC)

1.2.1 MES Quality Commander® (MQC) v.7.5

Powerful Access to Data Sources with RESTful API Adapters (beta)

- With RESTful APIs like the Gitlab API, you can access data from your Gitlab projects in a flexible and efficient way. MQC has now added new API adapters in addition to the standard file and git adapters. This allows you to retrieve e.g. artifacts directly from your pipeline using the Gitlab Pipeline Artifact Adapter.
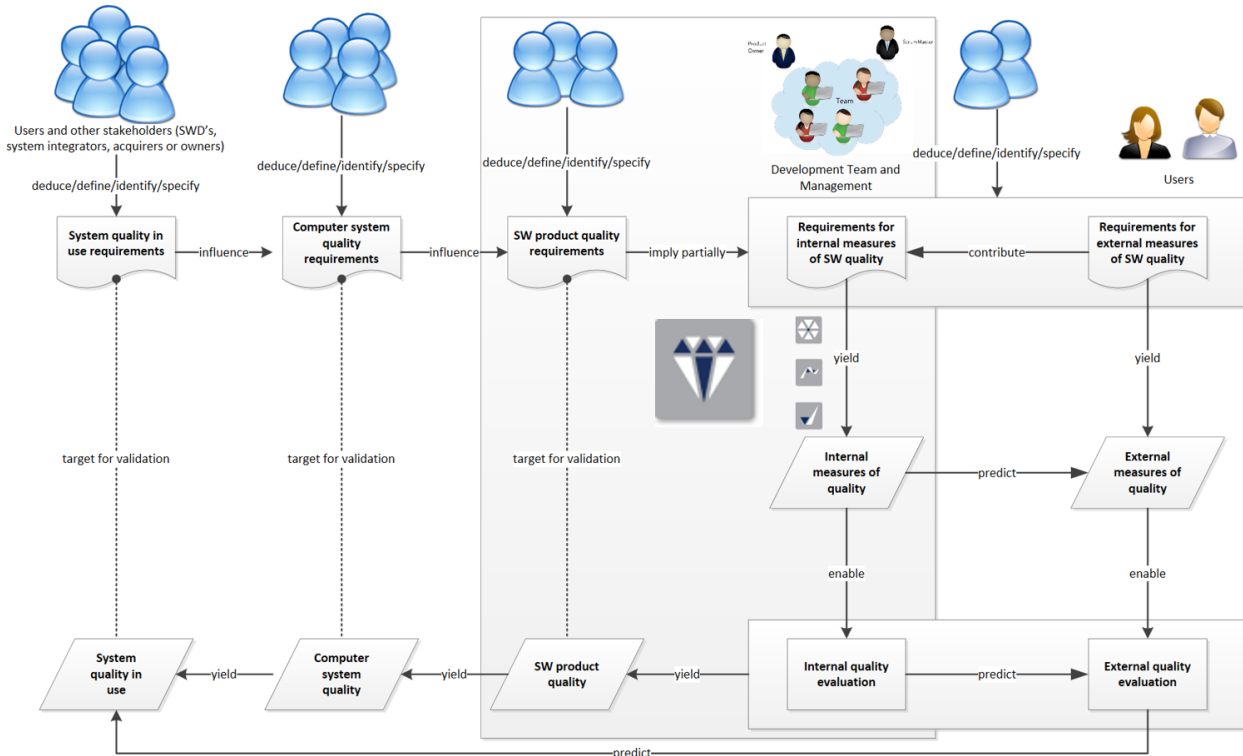
Figure 1.5: MQC Quality Life Cycle Model

- In the API adapters you can choose between either retrieving data directly or downloading files, which can be read by the enabled file adapters afterwards.

- The provided Gitlab API adapters allow you to access data from your Gitlab projects in a flexible and performant way:

  - Gitlab Pipeline Artifacts Adapter: Fetch file artifacts from your pipeline, without having to commit the files to the Git(lab) repository.

  - Gitlab Pipeline JUnit Adapter: Fetch the coverage and test summary of your pipeline directly as data.

  - Gitlab Repository Adapter: Fetch the files from commits in your Gitlab repository.

- The usual caching of files, monitoring for changes, and the differential import of new data are available.

- You can create and use your own custom API adapters in MQC, similar to custom file adapters. The provided API adapters are there as examples and are well documented for easy understanding. RestSharp is provided as a convenient library for accessing RESTful APIs.

For details see *API Adapters*.

Improved Validation of Quality Model Configuration

- We have added more checks to make sure the quality model configuration is consistent when it is imported. If there are any discrepancies, warnings are raised so you can fix the configuration.

• Validation of the interaction between different options has been improved to support quick corrections.

For details see *Validation*.

1.2.2 Highlights History

MES Quality Commander® (MQC) v.7.4

Visualize BTC Embedded Tester (ET) Projects in MQC

• Monitor the quality of your models tested by BTC Embedded Tester (ET) with an adapter to read the data from ET HTML reports. The ET quality model calculates quality for the data regarding test cases, requirements, code coverage, domain checks, and robustness checks.

• MQC extracts the complete data set:

   – TestCase result for MiL/SiL/B2B,

   – Requirement result MiL/SiL,

   – Testcase/Requirement Coverage and Requirement/TestCase Coverage,

   – The many Code Coverage measures: Statement, Condition, Decision, MC/DC, Function and Function call, Relational Operator, Switch-case,

   – Domain Checks: Valid Ranges and Invalid Ranges,

   – Robustness Check: Division by Zero, Downcast.

• The corresponding findings/issues for test cases and requirements are extracted as well.

• The quality model for Embedded Tester is as similar as possible to the other dynamic test tools (especially TPT and MTest).

For details see *BTC EmbeddedTester*.

Relations Between Findings in Data Details

• Showing the connection between e.g. requirements, test cases, and assessments simplifies the process of identifying and addressing the underlying causes of issues. For example, if there is a finding for a test case with linked requirement(s), the relation directly provides the findings of the requirement(s) including the state of the relation.

• The relations between findings are extracted from data sources and visualized in the Data Details List. This enables you to view all related findings in one place.

• The attributes of findings have been partially renamed (Subject, Issue Message).

For details see *Data Details List*.

MES Quality Commander® (MQC) v.7.3

Improve Testing Insights with Data Details for Dynamic Testing

- Elevate your testing insights by combining static and dynamic test data in a unified view. MQC's Data Details provide a comprehensive overview of issues, and seamless navigation through finding structures in an intuitive interface.

- Data Details are extracted for dynamic test (e.g. TPT). Detailed findings/issues regarding test cases, assesslets/assessments, requirements, and structural coverage information are read from the respective reports including information on the used plattforms (MiL, SiL, B2B, …).

- The issues from static and dynamic test are visualized in an integrated view in the Data Details Heatmap. The differing finding structures (i.e. checks/guidelines/documents for static test and measurements for dynamic test) are shown side-by-side in a unified way.

- The Data Details Treemap visualizes the full finding structure over all artifacts or individually per artifact. For dynamic test findings the structure inside requirements, test cases, and assessments is visible in the treemap visualization. This structure can be used to drill-down into specific areas of the findings.

For details see *Interactive Pages* and *Visualizations*.

Track Project Changes with Data Details Diff Mode

- Efficiently monitor and address changes in your projects with the Data Details Diff Mode. Gain a clear understanding of new issues introduced and previous issues resolved, facilitating informed decision-making.

- The diff mode of Data Details concentrates on the changed issues in your project. On the one hand, you see the places where new issues occurred and on the other hand where previous issues were solved. Using this detailed information the developer can work on the issues.

- The number of new and solved findings are shown in the KPIs directly. Tooltips provide detailed numbers in the heat map, treemap, and KPIs.

- The Data Details List shows the new and solved issues in diff mode. These issues can be sorted and grouped as usual.

- In the heat map, treemap, and KPIs the coloring directly shows, if the aggregated result over the respective findings/issues is improved (shades of green) or deteriorated (shades of red). If the aggregated result changes, the improvement or deterioration is labeled as major. Smaller changes are labeled as minor.

- The Diff Trend visualizes the number of solved and new issues per result type for the revisions. This provides a quick overview on the changes in the project over time.

For details see *Diff View*.

MES Quality Commander® (MQC) v.7.2

Collection and Aggregation of Data Details

- MQC collects Data (number of passed/failed guidelines, passed/failed test cases, etc.) and calculates Quality out of these data points.

- Now MQC collects Data Details too. These Data Details are the findings in a static model analysis, the results of each test case, or the non-covered requirements in dynamic testing and so on. Data Details provide detailed information on each Data point and facilitates the understanding of the source of quality deficits.

- Data Details are collected for MES Model Examiner (MXAM) so far. The information about documents, chapters, guidelines, and checks down to findings is available in MQC.

- A new page shows all artifacts' Data Details in interactive visualizations providing different perspectives of the data.

- The amount of the most critical findings per check is visualized in the Data Details heatmap. By simply changing the aggregation level you can switch to see the findings per guideline or even per document.

- To view specific findings you use the Data Details Findings list. Each of the visible findings and their most important attributes are shown in this list. If desired, more attributes (i.e. description) can be shown. A link into the respective tool report is provided (for MXAM directly to the respective check).

- The distribution of findings within the structure of an artifact is visualized in the Data Details treemap. This allows you to view the inside of an artifact recognizing the parts that have issues.

- If you want to focus on specific parts of your project, just use Marking as usual.

- Data Details are propagated in the same way as the related data points (variables of base measures).

- The Data Details visualizations (heatmap, treemap, list) can be displayed on the Quality and the Data page as well. When marking a quality property, quality bin, or measure, you will see the related findings in e.g. the Data Details Findings list or Data Details heatmap.

- The use of Data Details must be switched on (Import data details in Settings, default: off). The extent of Data Details can be configured: - All: all Data Details of all artifacts and all revisions -> loads of data - Last YY revisions: load Data Details of all artifacts for the specified number of revisions - On Demand: load Data Details by selecting which artifacts of which revisions you want to further analyze

For details see *Interactive Pages* and *Visualizations*.

Read Configuration Files from Git

- MQC configuration files (Project structure, Quality Model, Annotations, etc.) can be read directly from a Git repository. This can be the same repository as the report files or separate one(s).

- The configured Git repositories are preserved and can be re-used for the comfortable configuration of report and configuration files.

For details see section *Load / Reload* in Managing Configurations.

MES Quality Commander® (MQC) v.7.1

Multiple Sets of Milestones

Different areas of large projects often require many milestones which now can be defined as multiple sets in the project structure. Milestone sets can be grouped on different levels, allowing for a clearer overview and more focused visualizations.

Each set of milestones is a consecutive list of dates. Milestones from different sets can overlap or even have the same date.

The diffent milestone sets can be organised in a tree structure, so you can easily navigate a lot of milestone sets and quickly enable or disable them.

The toolbar allows the selection of all, one or multiple sets or even individual milestones. The selected milestones are visible in the trend visualization. If more than one milestone between two revisions is visible, only one line is shown. A tooltip gives detailed information about all milestones at that date.

For details see section *Milestone Structures*.

General Adapter Options for All Adapters

Many tool reports support the flexible configuration of content, meaning that the location and presentation of the same information may vary across different projects.

To facilitate the seamless import of reports, tool adapters support the configuration of FilePath based Adapter Options for the ReportDateTime, ArtifactPath, DataSourceName and MeasurementName.

The values for these fields can be extracted from the report file path with regex expressions.

The TPT xml tool adapter additionally supports the configuration of a MeasureNameMapping and MeasurementNameFromXml.

The MeasureNameMapping allows you to map MeasureNames from the xml to your Quality Model.

The MeasurementNameFromXml allows you to define xml paths and matching conditions to read the MeasurementName from a text or attribute of an element in the xml.

For details see section *General Adapter Options*.

MES Quality Commander® (MQC) v.7.0

Configurable Layout for Quality and Data Pages

The revised quality and data pages now allow for multiple visualizations, such as trend charts, to be arranged and saved in different layouts. Users can arrange visualizations as desired, they can be minimized to a sidebar or maximized to full page to adjust the focus.

The custom page layout can be saved.

Date Range Selection

Users can select the date range for trend charts, with options such as week or month, as well as custom date ranges. This limits the number of shown revisons in trend visualizations to a selected period of date.

Additionally, it is possible to move the chosen date range, e.g. go one week back or forth.

Updated MQC Menu

The MQC menu has been tailored to accommodate different user roles.

MES Quality Commander® (MQC) v.6.3

Support for Git sparse checkout (Beta)

MQC supports a sparse checkout of Git repositories. The use of this feature provides considerable improvements regarding disk space usage and checkout speed for huge repositories containing only a limited amount of files relevant for an MQC project.

With sparse checkout no full clone of the git repository is performed. Only the relevant files and directories based on the defined filters are downloaded from remote.

For details see section *Git*.

Load Setup Configuration into Create Project Dialog

The Create Project dialog is capable of loading a setup configuration. As such, an existing or saved setup configuration can be checked and extended before creating the project.

The complete configuration of the Create Project dialog is validated and helpful hints are provided to correct misconfigurations.

For more information refer to section *Importing a setup configuration*.

Dark Mode for the whole User Interface

A dark mode is available for MQC. All visualizations and UI elements are switched to a dark theme. The colors are adapted to be easily distinguishable and recognizable. At the same time the dark mode is pleasant on the eyes.

The theme setting is a user setting. Thus, each user sees its theme when viewing any project.

The report generation uses the light mode theme at all times.

MES Quality Commander® (MQC) v.6.2

New Distribution Visualization including Milestones and Overall Quality/Availability

The new distribution visualization now shows the milestones as vertical lines. This eases the temporal recognition of the quality or availability distribution over time.

All revisions before a milestone can be collapsed. The new distribution visualization then shows the last revision before that milestone. Collapsing revisions provides a streamlined trend (quality or availability) especially for long running projects with many revisions.

With the improved distribution visualization it is possible to switch between a bin view, which shows the distribution of e.g. the computed quality according to categories like "good," "acceptable," and "bad" per revision, and an overall quality or availability view per revision.

As part of the bin view, the new distribution visualization shows the overall quality or availability as an additional trend line.

The new quality distribution visualization reflects the current scope of quality assessment, which can be absolute, available or relative quality. By default the absolute quality is shown.

The reimplementation of the bin distribution visualization facilitates improved formatting of labels, axes, and more consistent marking and tooltips.

Automatic Data Update of Projects on MQC Server

MQC projects stored within the library of an MQC Server can be periodically updated to automatically fetch the latest data changes. This ensures that all projects are always up to date upon opening, without the need to run an import in the client.

The serverside automatic update can be enabled or disabled in the project without changing any configuration in the server.

Serverside updates are only executed if new or changed data was detected.

For more information refer to section *Settings*.

MES Quality Commander® (MQC) v.6.1

Artifacts and Quality in flexible structures

Typical development projects are often structured in various ways, i.e. by product architectures, by product platforms, even by roles and responsibilities. As a result, the artifact and quality model structures in MQC now support multiple flexible levels and freely configurable naming.

Appropriate filtering and marking now allow you to focus on many specific aspects of artifacts or quality properties, in addition to a general overview.

For more information refer to section *Quality Model* and *Project Structure*.

Selectable scope for visual quality assessment

The scope of quality assessment that is shown in different visualizations includes all quality properties by default (=absolute quality). However, it is now possible to adjust that scope by e.g. ignoring all missing quality properties (=available quality) or by adapting the assessment in relation to defined target values (=relative quality). This allows a more differentiated view on project quality at any time during project runtime.

For more information see *Quality Assessment Scope*.

Faster Data Import from Git

Now faster data import works for Git repositories as well. MQC detects new commits and imports only data contained in these commits. New data can therefore be imported into an existing project much faster.

In case some hidden changes were made in your Git repository, you can force refresh reading the data from the Git repository (which may take quite some time).

Use of commit time as report time is now a configuration of the Git data source, which can be found inside the data source configuration dialog.

Predefined User Roles on MQC Server

The MQC server solution automatically comes with two user groups: MQC Editor and MQC User.

MQC Editors are able to create and configure (new) MQC projects and MQC Users are able to load and view existing projects, but not create or edit them.

MES Quality Commander® (MQC) v.6.0

Multiple Quality Models

Your overall quality model can be configured out of multiple (smaller) quality models. This improves the handling, maintenance, and especially the extension of the quality model definition.

We recommend defining each data source in its own quality model. The quality models we provide with MQC follow this recommendation.

Even if there is no quality model defined, MQC recognizes the data sources used and loads the provided initial quality models.

For more information refer to section *Quality Model*.

Faster File Import and Transformations

Now MQC detects new data files and imports only the new data. In this way, importing new data into an already existing project happens much faster.

The time needed for calculations and transformation of the data is now reduced by half. This was achieved through improvements in the data flow and reducing the internal memory consumption.

As a result of the improved data handling MQC is more responsive and gives you more intermediate status updates of running calculations.

The adapter framework is faster now and was modified to reduce complexity. The GUI now focuses on showing only the standard adapters by default. However, the full set of adapters is still available with MQC.

Project Creation with Complete Configuration

We added an advanced mode to project creation, in which all necessary configurations for your project i.e. Project Structure, Quality Models, Settings, Adapters, etc. can be configured before creating the project. This facilitates a fast and comprehensive setup process.

For more information refer to section *Creating an MQC project*.

MES Quality Commander® (MQC) v.5.3

New Project Creation with Interactive Configuration

MQC streamlined the new project creation. When you select "Create new project" a dialog asks for the location of your data (source) and the revision granularity. After confirming the dialog the project will be set up.

With the new dialog less steps are needed to set up your project and you can provide data earlier. The overall setup process is quicker, because you can import your data directly during the setup.

Multiple Git Repositories as one Data Source

If you have a high number of similar Git repositories containing your data, you can now configure these as one data source.

One Git configuration (branches, commits, time range, etc.) can be used for a list of Git repositories with a similar structure (e.g. when each artifact is contained in its own repository).

This considerably reduces the configuration effort for Git data sources.

For detailed information refer to section *Git*.

Direct Access to MQC Showcase

The MQC Showcase project is now shipped with MQC and can be accessed with one click from the landing page in the web player or the client.

The MQC Showcase is a ready to use MQC project where you get the full MQC experience and try out the features for yourself.

For more information refer to section *Quick Start*.

MES Quality Commander® (MQC) v.5.2

Visualization and Aggregation by Artifact Structure Levels

To get a better overview of larger projects (hundreds of artifacts) the visualized elements can be switched from artifacts to higher levels of the artifact structure (i.e. StructureElement, StructureGroup, and Structure-Root). Instead of showing hundreds of artifacts, switching to higher levels of the artifact structure improves the visualization, so that you can recognize the StructureElements or the StructureGroups. At the same time this higher level visualization directly shows the aggregation of the respective qualitative values of the StructureElements or the StructureGroups. Together with a well-defined artifact structure you get a comprehensive view of your project on the different levels answering questions like what is the quality of this component or this ECU. At the same time you can see the elements in comparison to each of the other elements on the same level.

This structural level visualization works on all quality pages and the data availability page for artifact structure, quality model structure, and data structure. You can select a level in the hierarchy of the structure, defined by the Project Structure or the Quality Model, in the dropdown above the respective KPI visualization.

Different main visualizations (status matrix, heat map, and trend chart) respect the structural level selection in the KPIs and show the same aggregated view.

If you are on one of the higher structural levels you can expand/collapse the underlying elements to see more details. The selection 'Groups - Artifacts' shows groups in the KPI, which you can expand down to the artifact level. The main visualization on the right-hand side shows the artifact level. This allows you to quickly and easily mark the higher structural levels while viewing the detailed level on the main visualization.

Sorting and search in visualization

Sorting and searching were added to the KPI visualizations for Artifacts, Quality Properties, and Measures.

Now you can sort these KPIs using a dropdown. Different sorting options are quality ascending/descending, name ascending/descending, availability ascending/descending, and propagation ascending/descending.

Use the search function to reduce the amount of tiles shown and to display the relevant tiles.

Improvements for Annotations

Annotations are more prominently displayed in the main visualization on the Quality Status page. An "A"-Indicator highlights annotated quality points. The Tooltip for an annotated quality point shows the title, description, and the change of quality, if defined.

The Annotation UI dialog was further improved and streamlined. In addition to the grouped viewing mode, a flat viewing mode was added to meet different preferences. Now it is possible to see a list of all the annotations.

The Import and Export features were replaced by a more advanced toolbar. With this new toolbar it is possible to load, reload and save from/to the file system or a network drive. The Load dialog supports either the loading of a file (while keeping a reference to the data) or the uploading of a file, which allows the use of local files on the web player. Saving replaces a referenced file, saves as a new file or downloads as a file. All these allow web player users to save the file locally.

Adding a new annotation comes with more preselected fields based on the current marking, either in the artifact KPI, quality properties KPI, bin distribution or the main visualization. Additionally to quality properties and artifacts, the quality, bins, and revision start and end dates will be prefilled now.

It is now possible to define annotations without a change of quality or bin target, so that you can only enter a comment.

The transformations for annotations have been improved. When adding, modifying or deleting annotations only the necessary transformations will run, significantly reducing the execution time.

For more information refer to section *Annotations*.

Tool Adapter for RTRT HTML Report and RTRT Quality Model

The HTML report of Rational Test RealTime (RTRT) by IBM is supported. Together with the QAC adapter, MQC supports code based projects now.

The adapter reads the test case status and the available code coverage metrics.

The RTRT quality model contains all the read measures and quality properties for code coverage and test case compliance.

For more information refer to section *Rational Test RealTime (RTRT)*.

MES Quality Commander® (MQC) v.5.1

Annotations for Quality Properties (Beta)

MQC added the possibility to adapt quality values in a direct and documented way. These annotations adapt quality property values per artifact.

Annotations can change the quality bin (i.e. from bad to acceptable) or the quality value directly (i.e. from 74% to 93%).

A user interface aides the creation and update of single or multiple annotations.

Each annotation contains a title (short) and a description (long) to document the reasons for the change in quality. It also contains the author of the annotation, the creation date, and the (last) modification date.

The validity of an annotation can be defined for a specific time frame (by date).

For more information refer to section *Annotations*.

Multiple Human Readable Reports per Data File

The connection between data source files (i.e. XML files) and accompanying human readable reports (i.e. HTML files) was extended to handle multiple human readable reports per data source file. For example, in this way the adapters support separate MiL and SiL HTML reports for one XML file containing combined test results for MiL and SiL.

For more information refer to section *Data Origins*.

Extension of TPT Adapter

The TPT tool adapter for the TPT HTML overview report supports handling multiple variants of the overall result visualization.

For more information see *HTML*.

Propagated Derived Measures

Derived measures that use propagated base measures are propagated now as well. This makes the information in the Data Availability more consistent.

For more information regarding propagation refer to section *Data Propagation*.

MES Quality Commander® (MQC) v.5.0

Git as Source for Data (Beta)

MQC can use Git as a direct source for data files. By configuring the Git server, user, and repository now all commits are available as a source for data files.

Filters are available for commits, tags, author, directories, and files. These filters can be include or exclude. Simple text pattern up to full regular expressions can be used for defining the filters.

An option in settings allows the user to use the commit date and time of data files instead of the date inside the report file. The report date and time is set as default.

Employing the current functionality all data files can be directly taken from the version control system. In addition, the analysis by MQC can be constrained to a defined set of commits, directories, files depending on the project and what is needed.

MQC detects updates in the Git repository. The user is notified to refresh the data. If configured, the refresh happens automatically.

For more information refer to section *Git*.

Calculation and Visualization of Quality Differences

MQC now directly calculates and visualizes the changes/differences in quality and availability between revisions. The comparison base can be selected and separate Diff pages are available to visualize quality and availability changes/differences.

Diffs can be calculated in comparison to the previous revision or previous milestone. It is also possible to compare all revisions with a specific revision.

Separate Diff pages are available with difference specific visualization adaptions. These pages use percentage points to make the kind of changes clear. There is differentiation between positive (+) and negative (-) changes/differences. The titles of the visualizations describe the nature of the visualized data.

A Diff page is shown for a specific page (e.g. Quality Status page), when choosing "Show Diff View" in the Action Panel. Diff pages can also be managed via the "Manage Pages" menu.

The difference calculations and visualizations depend on the filter panel selection.

Quality Bin Configuration

The quality bins used on quality pages of MQC can be configured by name, color, and the used quality boundaries/thresholds. The number of bins is flexible. In this way a project might use specific names instead of good/acceptable/bad and the respective colors green/yellow/red.

An optional new sheet to define the parameters of bins is available in the Quality Model configuration. These parameters are used throughout all quality pages. The previously used configuration is still the default configuration in MQC.

For more information refer to section *Quality Bins*.

MES Quality Commander® (MQC) v.4.5

Configurable Dashboard

The dashboard can be used to get a quick project overview. Configurable tiles show you exactly the information you need for a first yet thorough impression. Tiles can be added removed, resized, and moved around using drag and drop.

The configuration of the dashboard is saved as part of the project.

For more information refer to section *Customization*.

Fuzzy Rules for Action Calculation

The calculation of action priorities depending on quality properties can now be done with the help of fuzzy rules. This provides flexibility in defining more complex non-linear relations between actions and quality properties. The fuzzy rules are implemented by the Fuzzy markup Language (FML) ISO standard.

The definition of linear dependencies in the quality model and the fuzzy definition can be used at the same time and can even be mixed.

For more information see section *Definition of Actions*.

A Sankey diagram on the Action List page shows the dependencies between quality properties and derived actions.

Additional/extended Tool Adapters

  • Tool adapter for TPT XML report and TPT Quality Model extended

  • Tool adapter for Simulink Check (Model Advisor) HTML report added

  • Tool adapter for Simulink Design Verifier HTML report added

For more information refer to chapter *Adapters*.

MES Quality Commander® (MQC) v.4.4

Action List (Beta)

MQC now recommends actions to improve quality by solving deficits in a project. Each action is related to a particular artifact with a priority from 'Very High' to 'Very Low' according to the corresponding quality value. The list of actions is then sorted according to the priorities, which helps the user to easily identify the most urgent actions to be taken next.

A new Action List page is provided which gives an overview about the actions for the current or a selected revision. Filtering and marking may be used to focus on details like particular artifacts.

For more information refer to chapter *Actions*.

Data Origins (Imported Data and Report Files)

MQC provides access to human-readable reports, for example in HTML or PDF format, if available. The underlying data reports, whether originally imported and/or human-readable, can be opened directly through MQC.

Using the new entry **Show Data Origin** in the MQC Action panel, makes it easy to crosscheck the original file source for a particular piece of data.

For more information refer to chapter *Data Origins*.

MES Quality Commander® (MQC) v.4.3

Tool Adapter API to Allow Custom Adapter Implementation

Tool Adapters can be added when working with MQC using the new API. In this way the import capabilities of MQC can be easily extended.

Adapters can be written in C# and (Iron)Python.

Examples for adapters reading XML, HTML, Excel, and CSV/TXT are available including all currently available tool adapters.

For more information refer to chapter *Custom Adapters*.

Tool Adapters for MXSuite, CTC++, and QAC Added

Code coverage data from CTC++ can be imported from XML and HTML reports. In addition to the code coverage data (statement/decision/… coverage) the number of source lines and measurement points are read as well.

Static analysis data from QAC can be imported from XML and HTML reports.

Test result data from MXSuite can be imported using MXSuite XML reports.

For more information refer to chapter *Data Sources*.

Significant Performance Improvement of Data Transformation Flow

The data transformations are now executed in a specific order, which is controlled and triggered in such a way as to prevent duplicate transformations. This improvement reduces the time needed for the transformations by up to half.

MES Quality Commander® (MQC) v.4.2

Significant Import Time and Memory Reduction for Huge Data

The import of huge sets of data files and complete directories of files is much faster now. In addition, the memory consumption was reduced considerably.

The mechanism of monitoring for file and directory changes was reworked to detect changes safe and with less computing resources. Due to the adapted caching mechanism the reimport of changed files is much faster now as well.

User Experience

The marking in the bin distribution is more intuitive now. Revisions on status pages can be selected with just one click. Similarly, a quality bin on trend pages can be selected for all revisions with one click.

The extension of marking when combining marking on bin distribution and in KPIs is more intuitive now.

The showing of the KPIs and the calculations inside the KPIs are fully dynamic now and reflect the marking on the other visualization of the page.

Artifact and revision marking on quality or data pages is directly reflected on other quality and data pages.

A full description of these mechanisms can be found in *Marking*.

Availability of MQC Server and MQC Web Viewer/Editor

The MQC Server is now available for on-premise installation and setup. Now you can have a secure library of your analysis including access to your local network (and data).

With this availability of the internal library you can easily create a shared analysis (and some tasks can run regularly on the server, i.e. to update an analysis with new data)

Together with the MQC Server, the MQC Web Viewer/Editor is now available. It provides the functionality of the client directly in a browser while the analysis is opened/running on the MQC Server.

MES Quality Commander® (MQC) v.4.1

Improved Configuration Dialogs

MQC settings dialogs have been redesigned for better usability, and are available in MQC editor as well as MQC web viewer.

For more information refer to chapter *Settings*.

Fine grained configuration of Context Categories

Context categories define the relations between artifacts and data sources, measurements, and base measures. They help to define and focus on the artifact-relevant data from the available data pool.

For more information see context categories in *Artifacts*.

MES Quality Commander® (MQC) v.4.0

Report generation project status

The new status report in html format documents all details of an MQC project, including graphics and context-specific tables to structure the information. The structure and extent of the report is configurable.

For more information refer to chapter *Report*.

Configure expected data via context categories

Improved tool adapter for Polyspace (The Mathworks)

For more information refer to section *MathWorks Polyspace*.

Extended tool adapter for TPT (Piketec)

For more information refer to section *PikeTec TPT*.

MES Quality Commander® (MQC) v.3.5

From quality to data details

Via the context menu, selected quality properties or artifacts link to detailed data specific to the selection.

For more information refer to section *Data from Quality*.

Introducing artifact weights for quality aggregation

The artifact weights, as well as the weights for quality values can be visualized in a hierarchical heat map, where the impact of an artifact and a quality value scales with the size of a colored tile.

For more information refer to section *Artifacts*.

Additional tool pages for MES M-XRAY and MES Test Manager

MES Quality Commander® (MQC) v.3.4

Target value configuration for measures and quality properties

For each measure or quality property one or multiple target values (thresholds, expectations) can be defined per project milestone.

For more information refer to chapter *Target Values*.

Introducing quality property weights for quality aggregation

For each quality property a weight can be defined to be used in the aggregation of quality properties to sub-characteristics in the structure of the quality model.

For more information refer to chapter *Quality Properties*.

Tool adapter for Polyspace (The Mathworks)

2 INSTALLATION

In this chapter, we will guide you through the installation procedure for the MQC editor.

2.1 INSTALL MQC DESKTOP CLIENT

To install MQC, you must download the required Windows installer. You received or will receive an email from MES containing a link and the necessary credentials to download the installer file from our server.
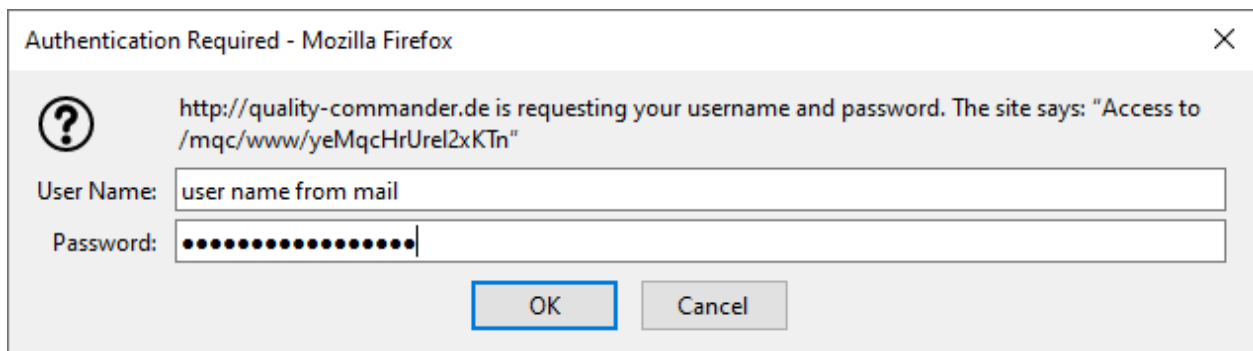

Figure 2.1: Type in the credentials and press OK

Please note that you must be able to unzip the files.

After extracting the zip-file, navigate to the subfolder: `MES_MQC_SpotfireAnalyst11-4-2_single_user_win\Produc` `Spotfire Installer`.

Now open the file `setup-single-user-11.4.2.exe` and proceed with the next step.

When installing MQC, the login to the MQC server is mandatory. This step will validate your MQC license and will provide you with the latest updates of the current version of MQC.

Please note that you must necessarily log in at least once every 30 days. This makes sure that your MQC license is revalidated and you can take advantage of updates, bug fixes and further improvements.

2.2 CHANGE DEFAULT PASSWORD

If you would like to change the default password provided by MQC, please open a browser and navigate to http://mes-qualitycommander.com. Login with your personal credentials.
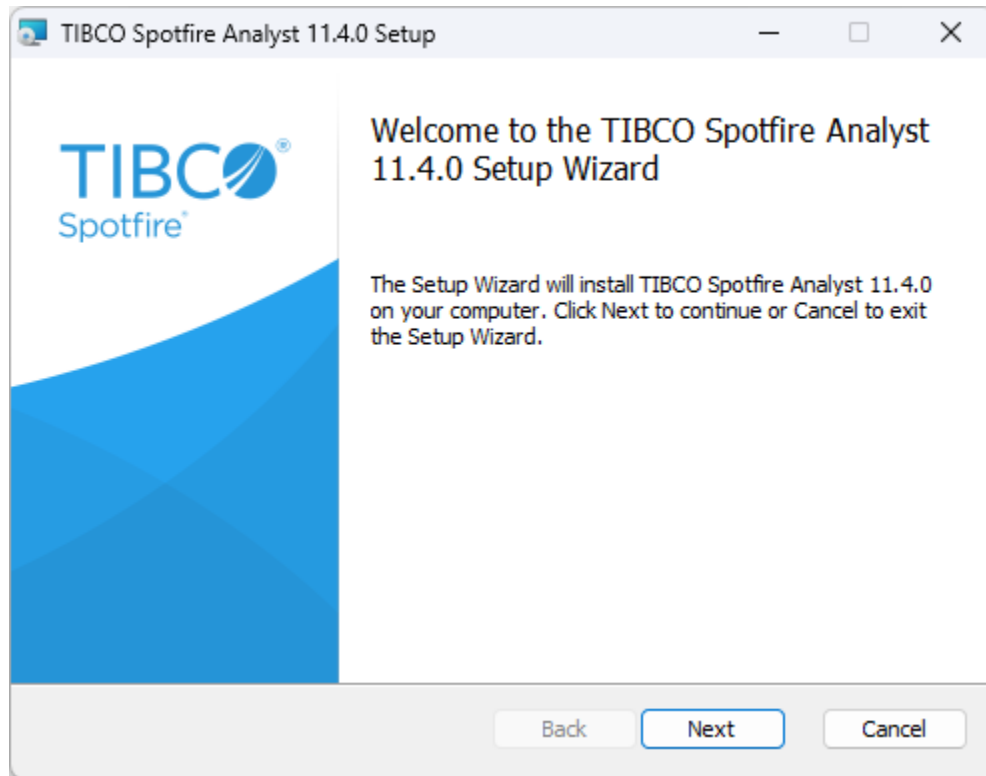
Figure 2.2: In the installation wizard, press Next.

Then inside your browser click on your user name at the top-right of the window and choose 'My account'. This opens the Administration Console for your account. Select 'Change password' and follow the instructions of the 'Change TIBCO Spotfire password' dialog.

2.3 SYSTEM REQUIREMENTS MQC EDITOR

Note: MQC Editor installation includes TIBCO Spotfire Analyst installation
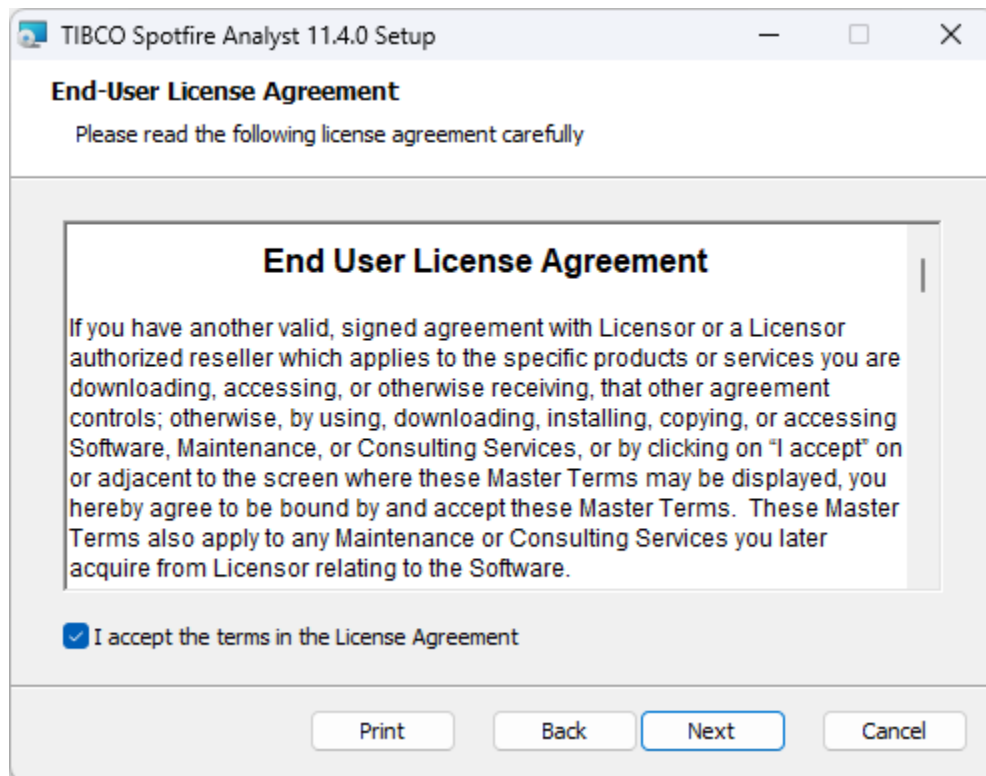
Table 2.1: System requirements for the MQC editor Hardware

Figure 2.3: Please read the license agreement carefully, accept it using the check box and click on Next.
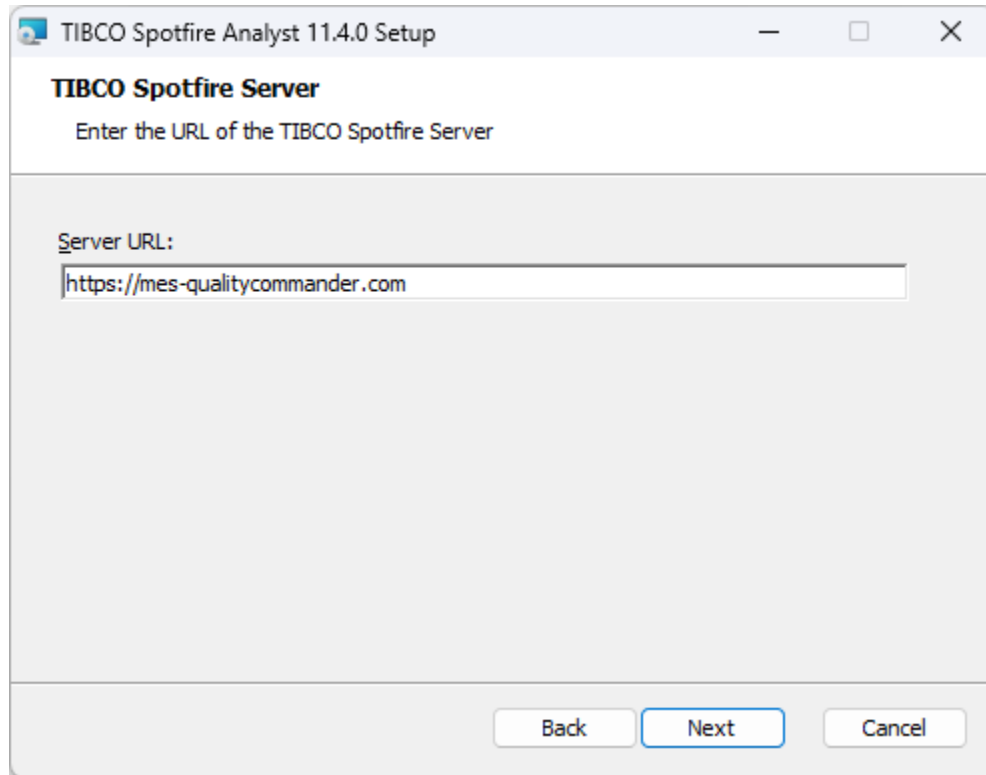
Figure 2.4: Enter the URL http://mes-qualitycommander.com/ as server address and press Next.
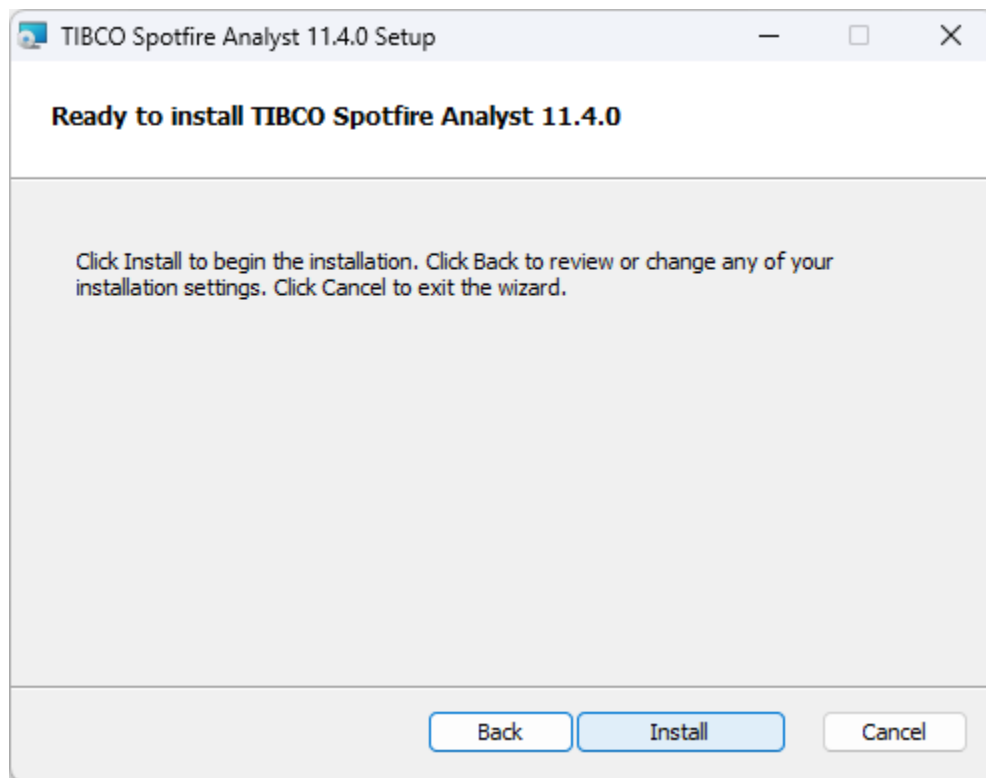
Figure 2.5: Finally to install the program, press Install.
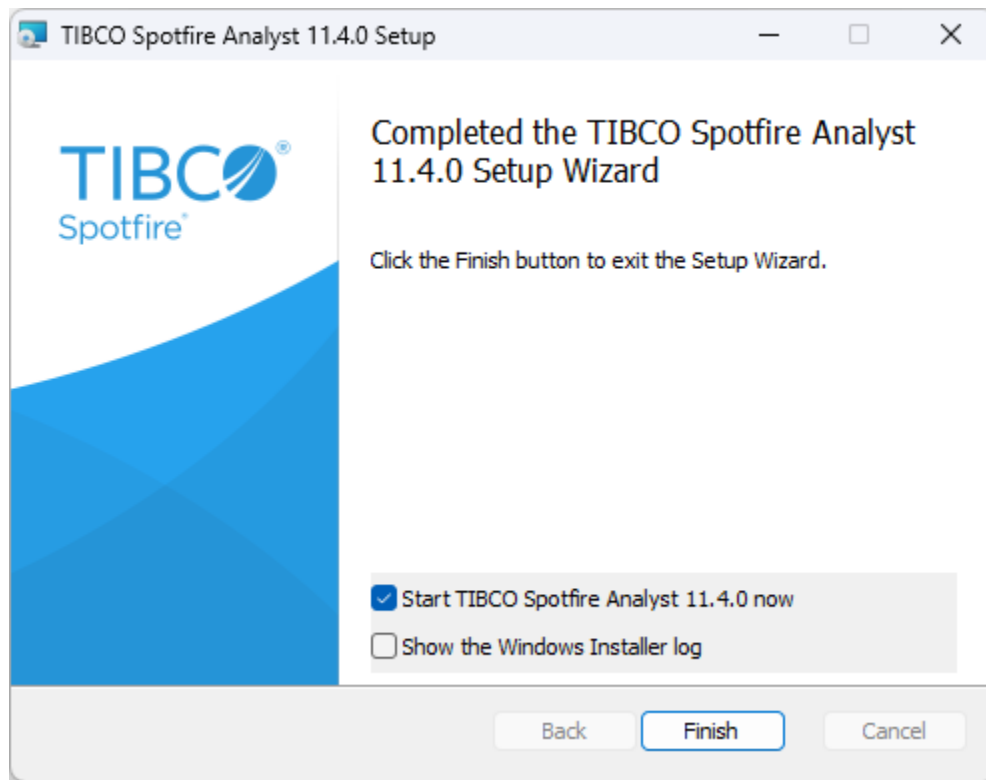
Figure 2.6: The installation may take a few minutes. Afterwards, please confirm with Finish.
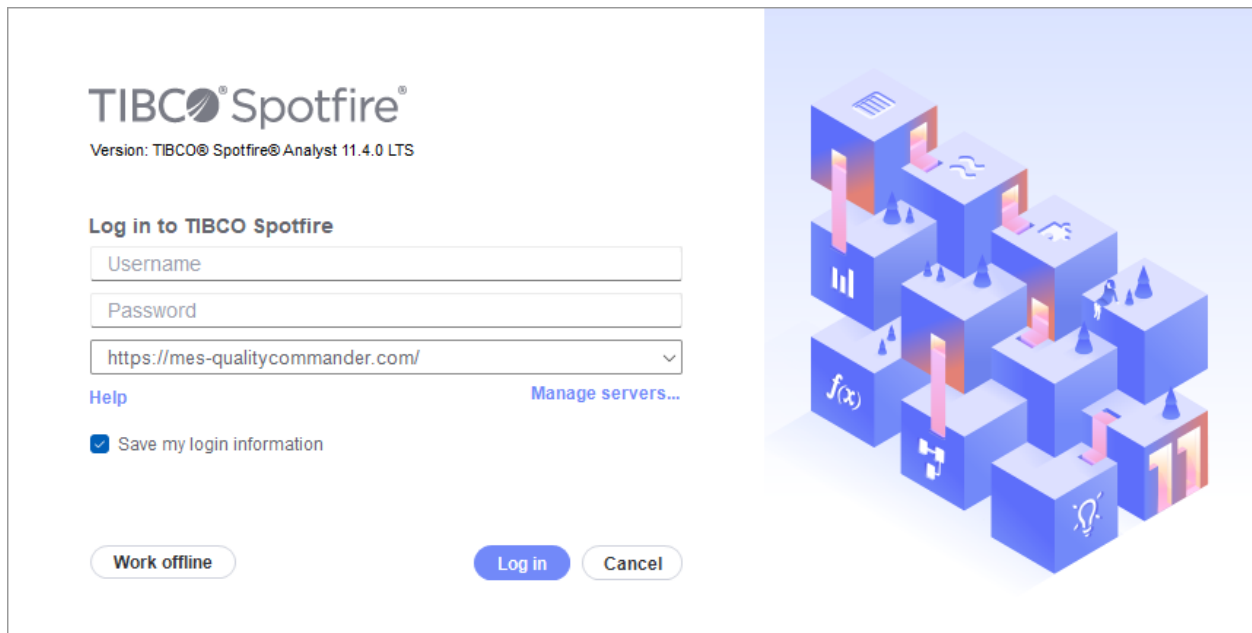

Figure 2.7: Please, open Spotfire and login with your personal credentials using the username and password provided to you by MES. Please note the lowercase and press Log In.
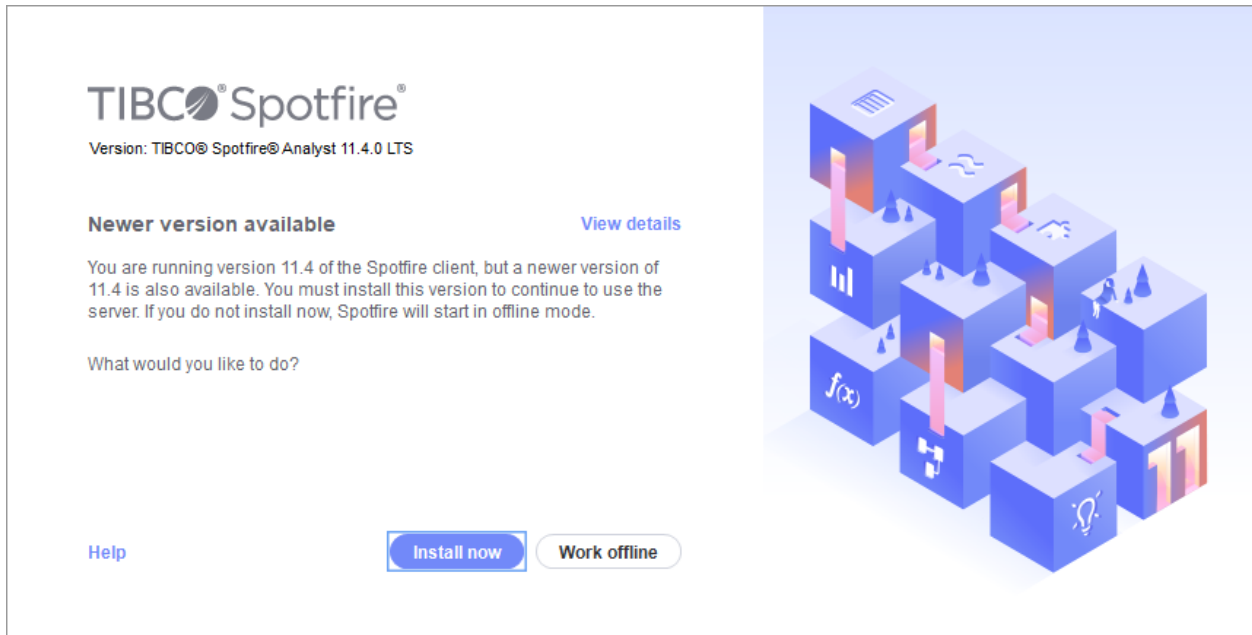
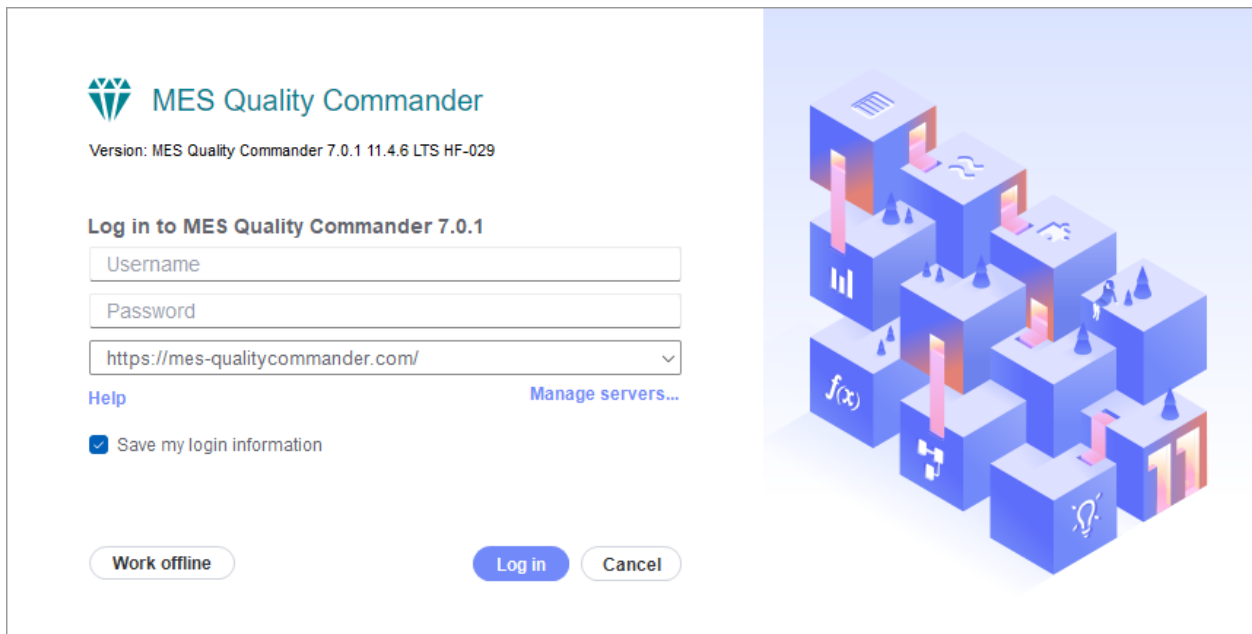**Figure 2.8: Please confirm the requested update by pressing Install Now**

Figure 2.9: Once the update is complete, you can now start MQC by logging in with your credentials as mentioned above. Click on Log In

Category	Requirement
Hardware	
Processor	Minimum: 2 Cores, 2 GHz Recommended: 4 Cores or more (Intel Core i5 or equivalent), 2+ GHz, 64-bit
RAM	Minimum: 4 GB Recommended: 8 GB or more Note: Large data sets can require more RAM
Hard disk space	10 GB is recommended for installation and normal use.
Display	Minimum: 1024x768 pixel resolution, 16 or 32-bit color depth. Recommended: 1920*1080 pixel resolution or higher, 32-bit color depth.

Table 2.2: System requirements for the MQC editor Software

Category	Requirement
Software	
Operating System	Microsoft Windows 10 Microsoft Windows 8, 8.1 Microsoft Windows 7 64-bit
Installation Permissions	Administrator rights are required.
Microsoft .NET Framework	Microsoft .NET Framework version 4.5.x, 4.6.x, 4.7.x. NOTE: Microsoft .NET Framework 4.6 (or higher) is strongly recommended. NOTE: If Microsoft .NET Framework 4.5 isn't installed when running the installer, the installer can (if the users accepts this when prompted) download and install it.
Microsoft Office (Optional)	Microsoft Office 365 Microsoft Office 2016, 32- and 64-bit versions Microsoft Office 2013, 32- and 64-bit versions Microsoft Office 2010, 32- and 64-bit versions NOTE: Microsoft Office needs to be installed in order to use functionality that integrates with Office products (such as importing data from Excel or Access, or exporting to Power Point)

3 APPLICATION

3.1 INTRODUCTION

After opening and logging into MQC, you will see the landing page with the Open-Dialog visible. The Opening Dialog per default shows "Recommended", where the recently opened projects are listed to provide a quick jump into.
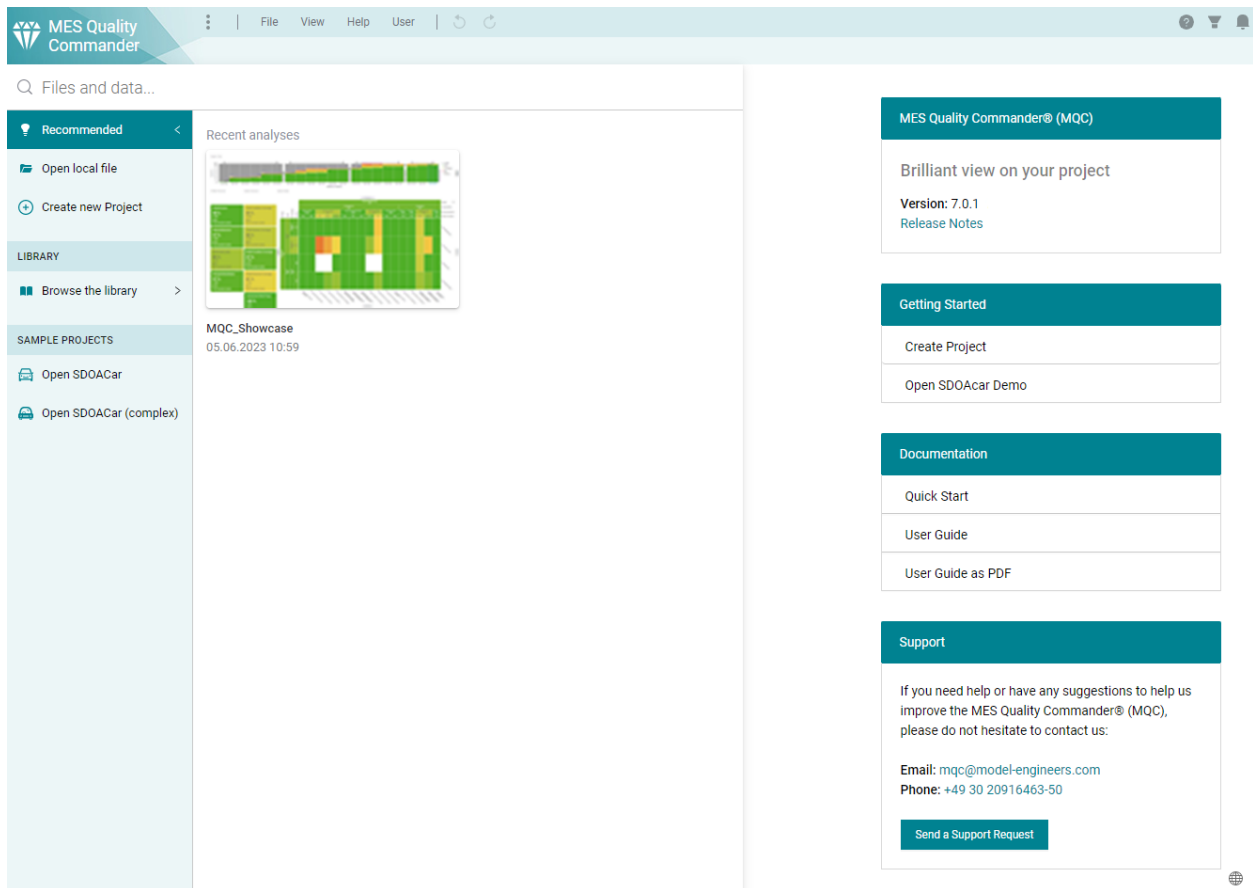

Figure 3.1: Landingpage with Open-Dialog

The Open-Dialog gives you the ability to Open a local stored project (.dxp), to Create a new Project (only for Editors), to Browse the *Library* and to open a project from there, or open one of the *Sample Projects*.

In the landing page you will find the current version of MQC with a Link to the Release Notes, Getting Started

links, Links to the Documentation and the Contact details of our Support Team.

3.1.1 Web Player and Desktop Client

The MQC Server provides a Web Player to open a project directly in the browser. When opening a project, a new instance of MQC is created on the server, which allows working with multiple open projects in parallel.

The Web Player is feature identical to the MQC Desktop Client and is recommended.

A link to a specific project opening directly in the web player uses the following format: https://your-mqc-server.intern/spotfire/wp/OpenAnalysis?file={LibraryPath}

3.1.2 Library

The MQC Server library is a virtual storage of created projects (.dxp).
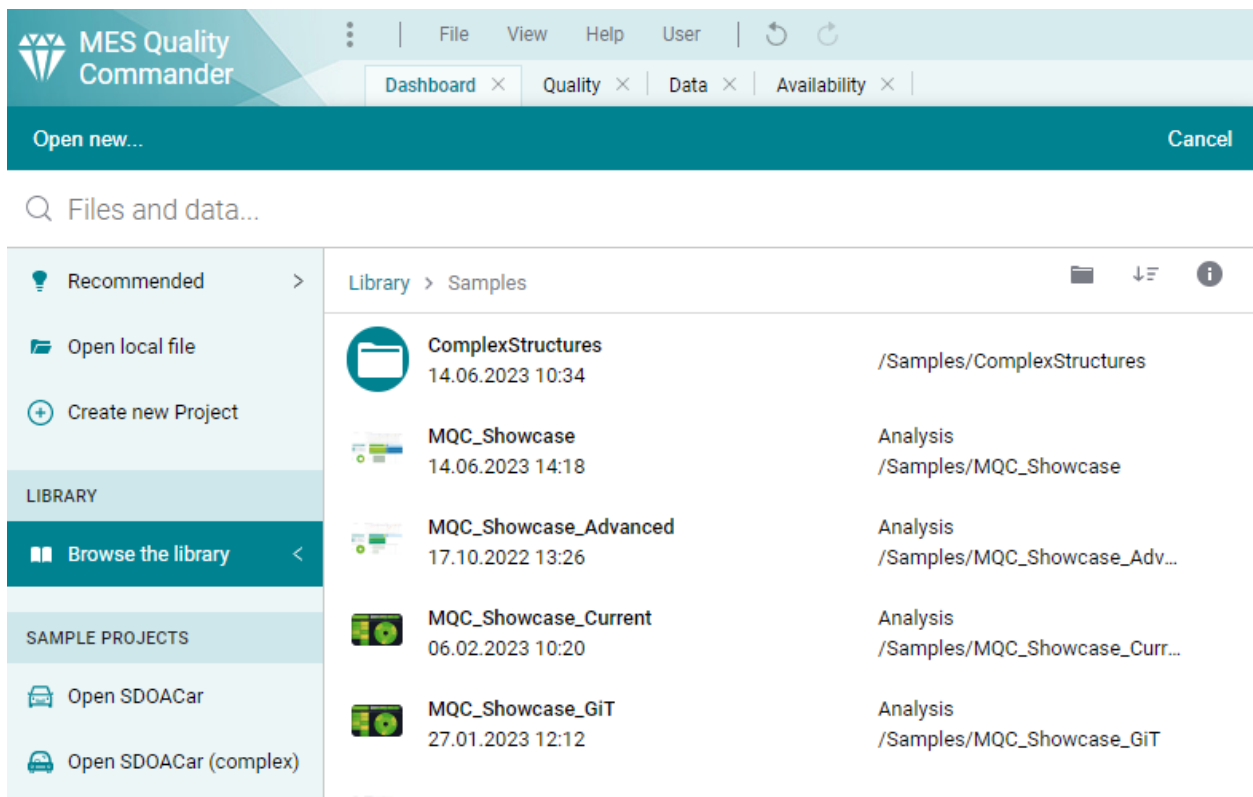

Figure 3.2: Server library in the web player or desktop client

Every user has access to the same library, but the permissions can restrict access to functionality and whole folders.
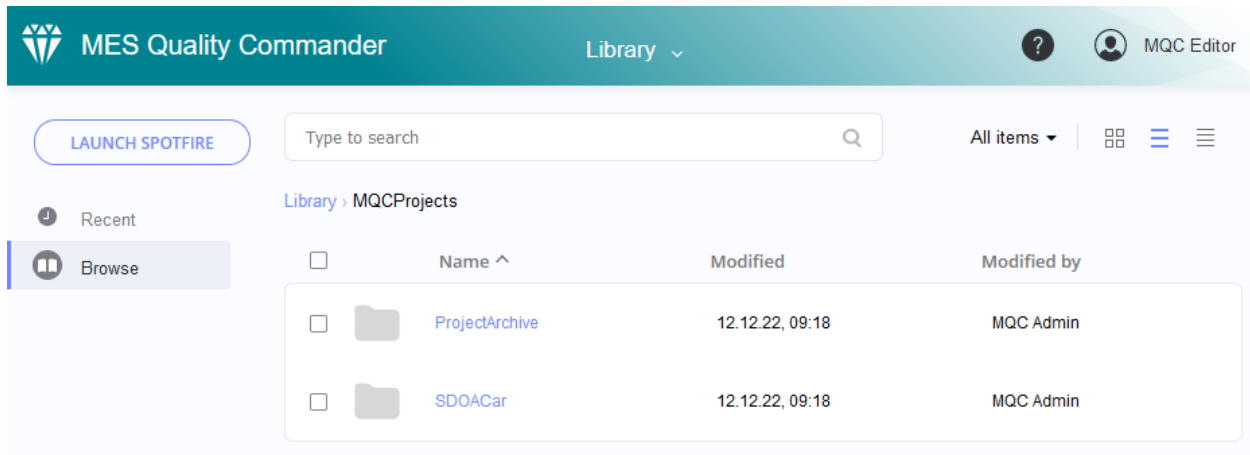
Figure 3.3: Server library with accessible folders and saved analyses

3.1.3 Sample Projects

The example project "SDOAcar" (Self Driving Obstacle Avoidance car) is meant as an introduction to MQC. Opening the default or the complex configuration of this project presents the user with a fully functional MQC project to play around with (see *Quick Start*).

3.2 QUICK START

One of the biggest advantages of MQC is its interactivity. The tool does not show static graphs. Instead, the information provided by each visualization will adapt according to the actions of the user. It is possible to focus on specific details as well as to arrange visualizations conveying different aspects next to each other to get a better picture of a project.

In the following, some of the main use cases when working with MQC are described.

3.2.1 Check the availability of the expected data

The data availability as shown on the Availability page acts as a first indication of how the project is progressing.

Grey areas inside the Availability Heatmap (see Figure 3.4) are depicting, which of the expected data for the expected artifacts is still missing. That may point to not yet available data reports respectively it may be an indicator for not yet executed tasks, especially in a very early phase of the project.

Missing data leads to missing quality, which has an impact on the overall quality of the project, because per default missing quality is treated as 0%.

Data that was available in previous revisions can be used in later revisions as well until it is replaced with data from a newly created report, e.g. after a test re-execution.

In MQC this is called data propagation (see *Data Propagation*). Propagated data is indicated by a light-blue color as shown in the Availability Bin Trend chart on top of the Availability page.
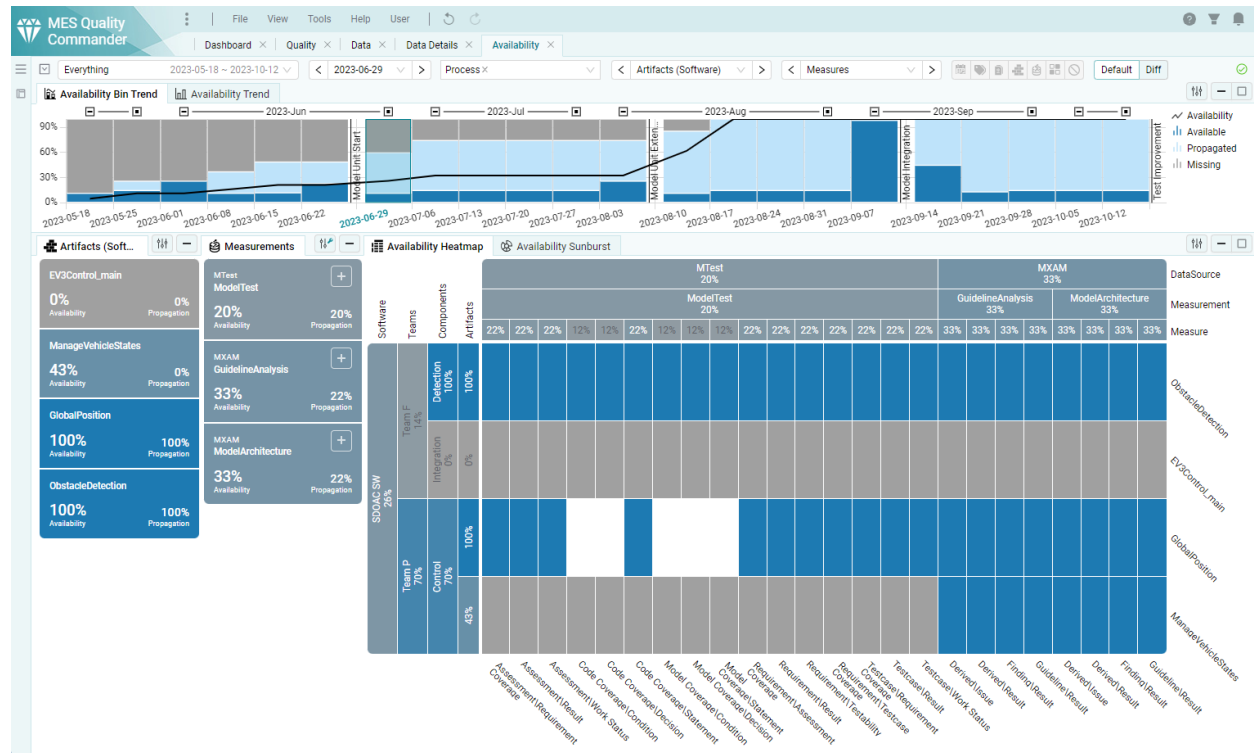
Figure 3.4: Check the availability of the expected data on the Availability page.

3.2.2 Check the current quality status

One of the most important information about a project is the current quality status. Quality in MQC is calculated based on the data read from the available reports as described in detail in *Quality Computation*.

MQC shows a quality value per artifact and quality property (see Figure 3.5) as well as aggregated on several levels, for instance the quality for groups of artifacts and/or specific quality characteristics as defined in the underlying quality model according to ISO-25010 (refer to *Structure*).

MQC initially uses a traffic light coloring scheme to visualize the computed quality. Having green for good quality and red to depict that quality is bad, makes it easy to detect problems, i.e. parts of the project with insufficient quality.

3.2.3 Check the quality progress of the project

Besides the status, the quality trend (as well as data trend) provide further useful information.

Based on the development of the quality measures over time as shown in Figure 3.6, and especially in combination with milestones and expectations (i.e. targets), it is possible to derive a prediction about the project progress. Areas with an increased need for action can be easily identified.

Trends in MQC are shown for various items, like artifacts and quality properties, as well as aggregated on different levels.

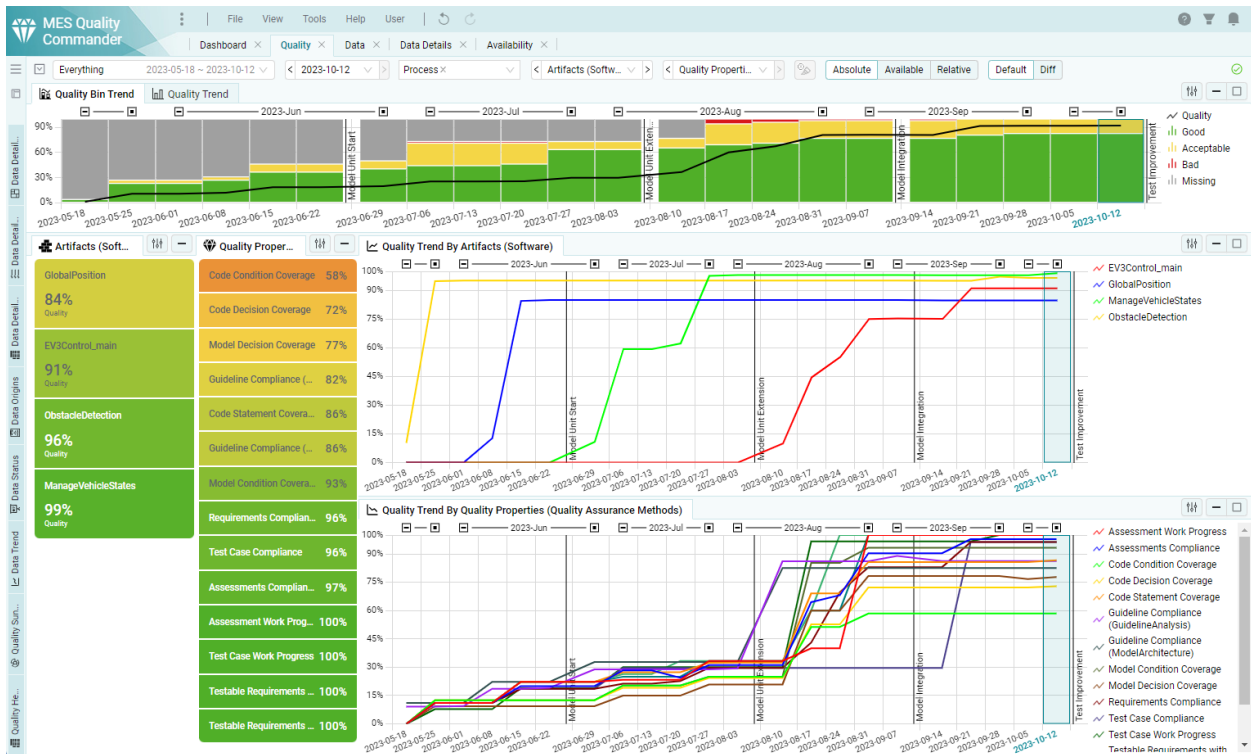**Figure 3.5: Check the actual quality status of the project on the Quality page.**

Figure 3.6: Check the quality progress of your project on the Quality page.

Find more information about how to add visualizations to a page or how to switch between different visualizations in *Interactive Pages*.

3.2.4 Check for areas with poor quality

Due to the traffic light color schema used in MQC, areas with poor quality can be easily detected. Focusing on such areas is possible by the Marking functionality (see *Marking*).

In case of quality issues have been identified, the tracing functions provided by MQC may be used to gather more specific information about these aspects, for instance by checking the underlying data that was used to calculate the corresponding quality measures, as it is shown in Figure 3.7.
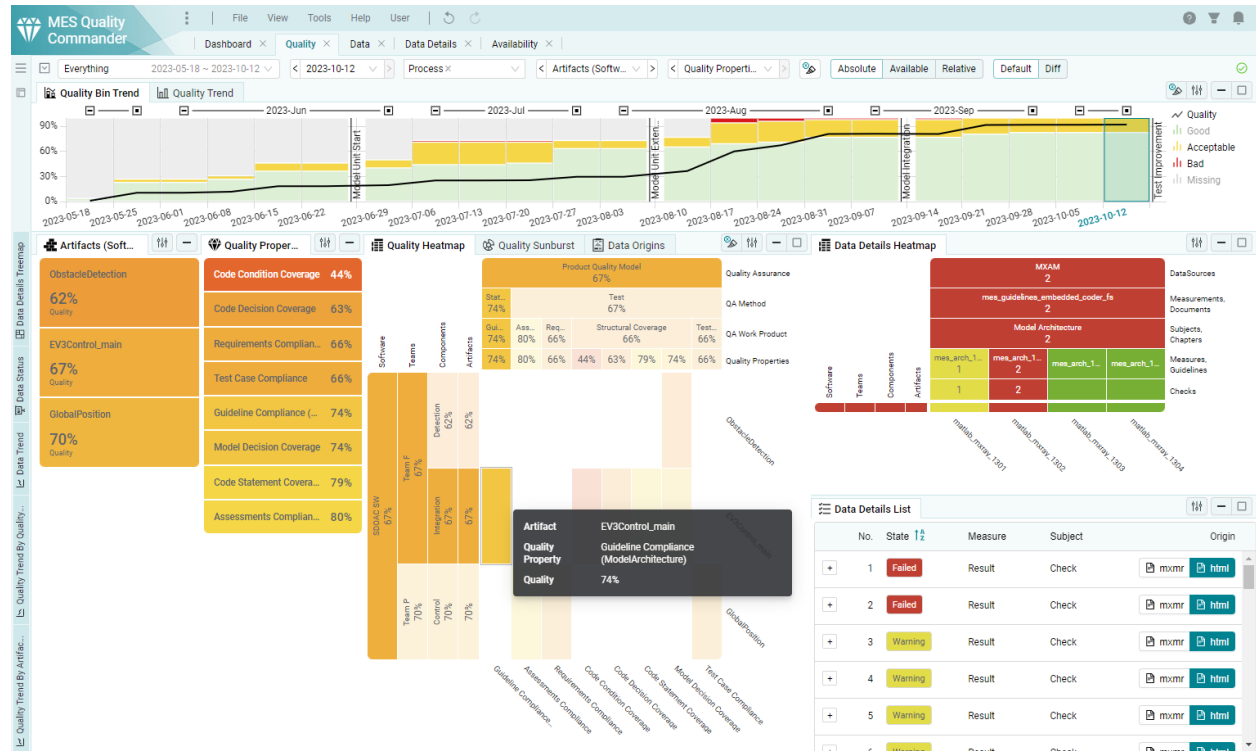

Figure 3.7: Focus visualization on quality properties and data producing poor quality.

Find more information about how to add visualizations to a page or how to switch between different visualizations in *Interactive Pages*.

3.2.5 Check what has been changed

With MQC, it is possible to easily compare the current status of quality or data availability with, for instance, the status of the previous revision.

As shown in Figure 3.8, this provides a detailed view of what exactly has been changed. Especially in the case that the overall quality of the project stays the same, the diff view per artifact and quality property may show improvements of the same magnitude as deteriorations in other aspects of the project. Thus, identifying and focusing on problematic areas is facilitated.
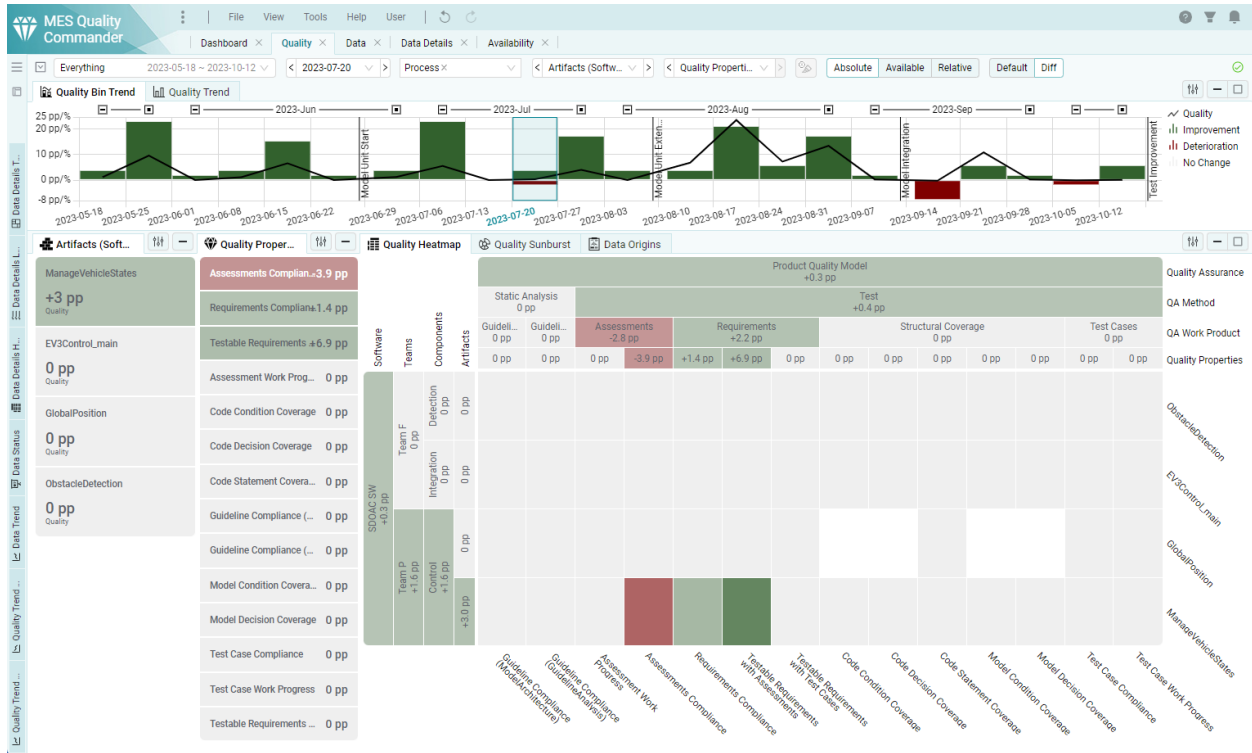
Figure 3.8: Use the diff view to compare the current status of e.g. quality against the status of a previous revision.

The view mode can be changed for all visualizations or for specific visualizations only. See *View (Default / Diff)* on how to switch the current view mode.

3.2.6 Check the data origins of problematic quality measures

As depicted in Figure 3.9, MQC is not just able to show the underlying data, which produced problematic quality measures, but also to open the reports the data initially was read from.

With a single click, it is possible to directly analyze specific details down to the root cause of a problem instead of manually searching inside a high amount of possibly relevant report files.

Whenever available, MQC offers a human readable report file (for instance an HTML report) next to the report the data was read from (e.g. in XML format), which is opened in a browser.

3.3 INTERACTIVE PAGES

The interactive pages of MQC are separated into 4 different types, that share the same functionality but differ in the information they present and how it is shown.

- **Quality Page**

  The quality page shows the quality of your project that is calculated by definition of the quality model out of your data.

Figure 3.9: Data reports can be directly opened via MQC.

It is also possible to view the data related to that quality.

(see *Data from Quality*)

- **Data Page**

    The data page shows the Data of your project in the way it was imported by the data source adapters.

    It is also possible to view the data origins and open the relevant report files directly in MQC.

    (see *Data Origins*)

- **Availability Page**

    The availability page shows the availability of the data in your project and also highlights propagated data.

    It is easy to see the still missing data, that is not excluded as "not expected" by context categories.

    (see *Context Categories*)

- **Data Details Page**

    The data details page shows detailed information about the data in your project.
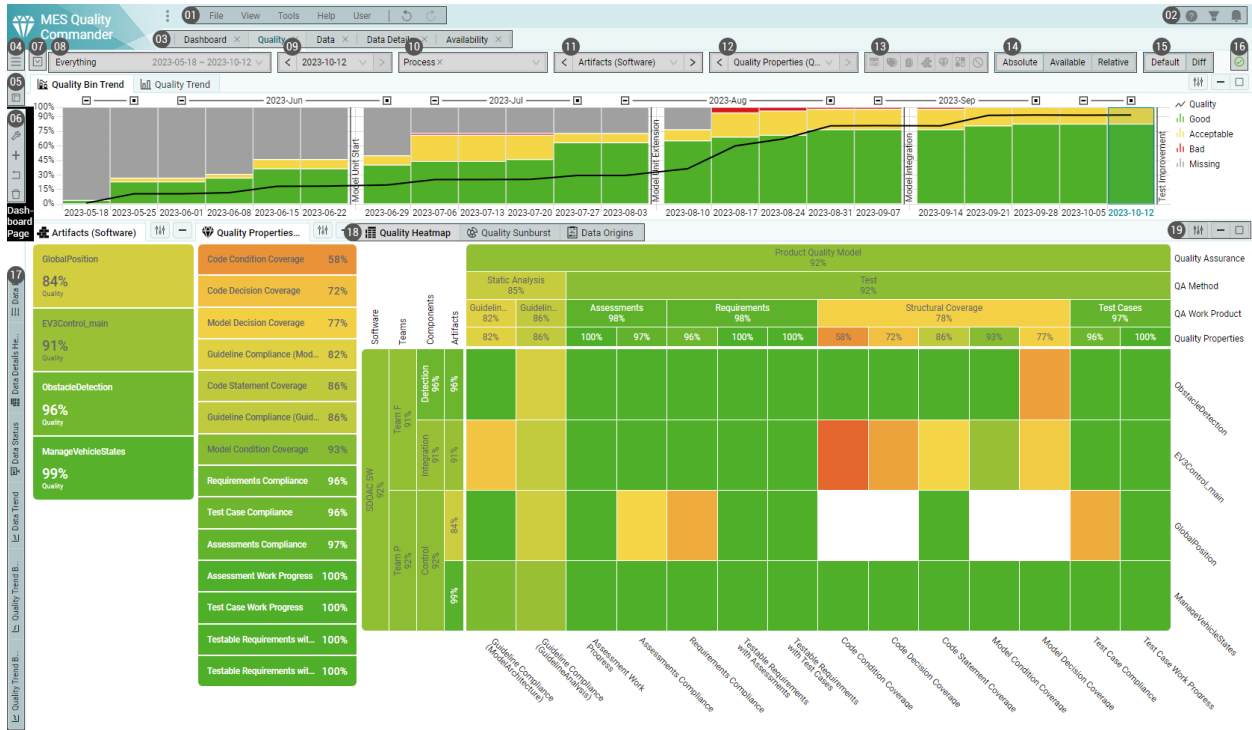
3.3.1 Overview

1. **Menu**

Figure 3.10: Quality page with important features annotated

(see *Menu*)

2. **UserGuide, Filter panel, Notifications**

   - UserGuide - Link to the user guide.

   - Filter panel - Show or hide the filter panel.

   - Notifications - Open the notification menu where warnings and errors are listed. New notifications pop up as messages temporarily before they disappear. Until the notifications are discarded, they remain in the notification list in the menu. Notifications should be considered with care and only ignored or discarded when the problems behind them are understood.

3. **Page Navigation**

   All enabled pages for your project are shown in the page navigation and can be opened by clicking on them. It contains dashboard, interactive, and custom pages. Only an Editor user can remove and/or add pages.

4. **Configuration Menu** (only for Editors)

   (see *Managing Configurations*)

5. **Page Layout Switcher**

   The Page Layout Switcher allows you to select a different predefined page layout from template.

   (see *Templates*)

6. **Dashboard Controls** (only on Dashboard Page)

(see *Dashboards*)

7. **Toolbar Menu**

   The toolbar menu contains all toolbar items for the current page, so in case some toolbar items are not visible because the screen width is not enough to render all the items, they are still accessible in this menu.

   (see *Toolbar*)

8. **Date Range Selection**

   (see *Date Range Selection*)

9. **Revision Selection**

   (see *Revision Selection*)

10. **Milestone Visibility Selection**

    (see *Milestone Visibility Selection*)

11. **Artifact Structure Selection**

    (see *Structure Selections*)

12. **Quality Property Structure Selection** (only on Quality Page)

    Measure Structure Selection (only on Data Page)

    Finding Structure Selection (only on Data Details Page)

    (see *Structure Selections*)

13. **Active Markings**

    (see *Active Markings*)

14. **Quality Assessment Scope** (only on Quality Page)

    (see *Quality Assessment Scope*)

15. **View (Default / Diff)** (not on Data Page)

    (see *View (Default / Diff)*)

16. **Data Import State**

    (see *Data Import State*)

17. **Minimized Visualizations**

    Minimized visualizations are collected on the left side of the page. They can be restored to the page by clicking on their tab or by dragging them into the page areas.

    (see *Visualizations*)

18. **Visualization Tabs** (Visible and Invisible Visualizations)

Multiple visualizations can be placed together in a group. Only one visualization in a group is active, while the rest of the visualizations are inactive. By clicking on a tab, the clicked on visualization becomes active.

(see *Visualizations*)

19. **Visualization Controls** (Options, Minimize, Maximize)

Visualization options are different for each visualization and allow specific settings for the visualization and/or the overriding of global toolbar options.

MQC allows you to maximize and minimize visualizations. When maximized, a visualization fills all page layout areas. However, the page layout is not modified and can be restored by clicking the restore (unmaximize) button.

(see *Visualizations*)

3.3.2 Menu

The menu consists of File, View, Help and User

- **File**

  Open, Save (only as editor) and Close your project.

  Create Report allows the creation of a pdf or html file. (see *Report*)

  View Library opens the library browser of saved projects on the server.

  Download a DXP file allows you to save a local copy of the project to your pc, that can be opened with the MQC client.

- **View**

  Change your theme: Dark or Light mode.

  Reset all filters. (see *Filter Panel*)

  Reset all markings. (see *Marking*)

  Filter panel: Show or hide the panel.

  New window: Create a second instance of MQC.

- **Help**

  Link to the user guide.

- **User**

  Logout and manage your account.

3.3.3 Page Layout

The page is divided into 3 different areas: Top, Kpi and Main.

While the Top and Kpi area of the page collapses when all visualizations inside these areas are minimized, the Main area expands into the space a collapsed area leaves behind.

The Kpi Area is reserved only for Kpi visualizations and Kpi visualizations can only be placed into the Kpi area.

The Top and Main Area support all non-Kpi Visualizations.

Visualizations can be placed side by side or stacked to each other in any complex nesting.
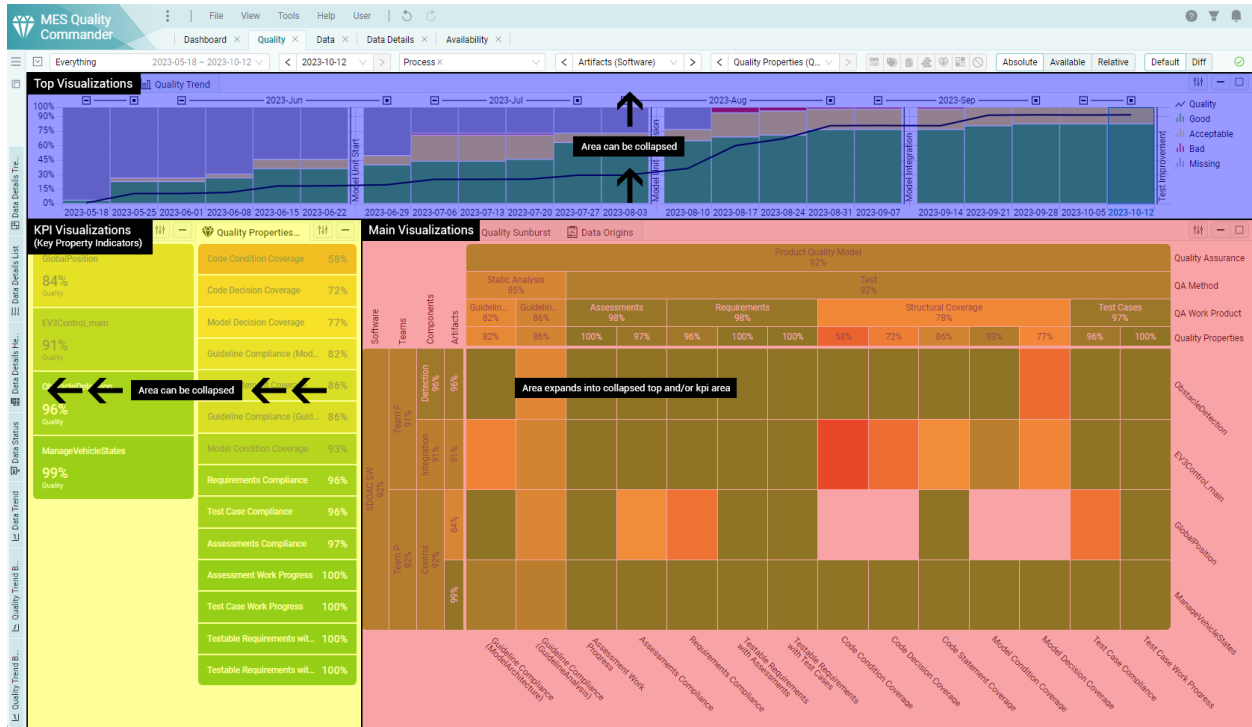

Figure 3.11: Quality page with the 3 page layout areas highlighted

Visualizations

Visualizations can be:

- **Active**

  The visualization is visible and the tab of the visualization is in the active state.

- **Inactive**

  The visualization is part of a visualization group and is currently not the active tab.

- **Minimized**

  The visualization is removed from the page and minimized to the left of the page as a vertical tab.

- **Maximized**

  The visualization (and it's group with inactive visualizations) take up the whole space of the page. The page layout is not modified and the visualization can be restored to its initial state.

Drag & Drop

The page layout can be modified through drag & drop.

Active, inactive and minimized visualizations can be dragged from their current position and dropped at valid drop locations.
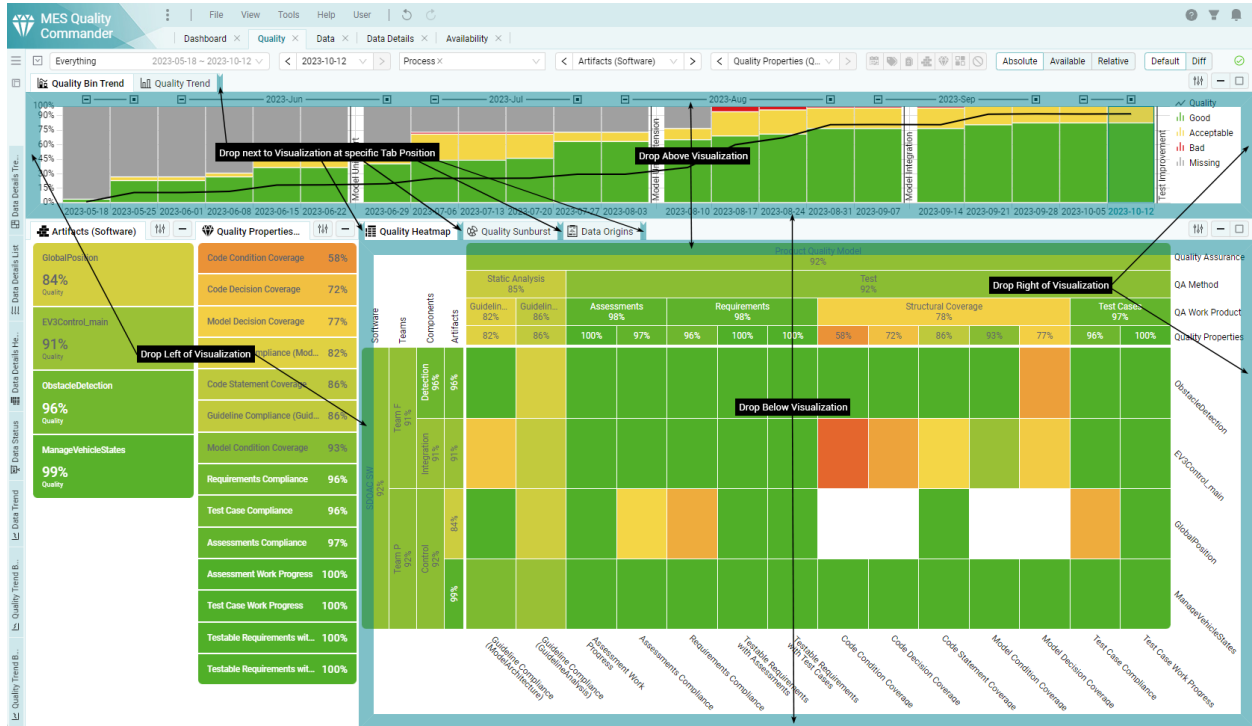

Figure 3.12: Quality page while dragging a visualization tab

Valid drop locations are:

- **Left or Right of a Visualization**

  By placing a visualization left or right of another visualization, it is placed side by side to it. Each visualization in a side by side takes up the same width. (e.g. 2 visualizations side by side means 50% of the width for each. 3 visualizations would mean a width of 33.34% for each)

- **Above or Below a Visualization**

  By placing a visualization above or below of another visualization, they are placed in a vertical stack. Each visualization in a vertical stack takes up the same height. (e.g. 2 visualizations side by side means 50% of the height for each. 3 visualizations would mean a height of 33.34% for each)

- **Next to a Visualization (Tab Position)**

  By placing a visualization next to another visualization it is put into a group with the visualizations already there. Only one visualization of a group can be active, the other visualizations in that group are inactive until their tab is clicked on.

Templates

The Editor can supply page layout templates that provide predefined page layouts (see *Pages Layouts*). A predefined layout can be applied by selecting it in the Page Layout Switcher.

MQC ships with 4 page layout templates for the quality page for the most common use cases:

- **Status**: Quality status of one revision

  (with Kpis for marking)

- **Trend**: Quality trend over a date range

  (with Kpis for marking)

- **Status with Data**: Quality status of one revision with related data status

  (Marking in Heatmap or Sunburst allows viewing of related data)

- **Trend with Data**: Quality trend over a date range with related data trend

  (Marking in trends allows viewing of related data)

- **Data Details**: Quality status of one revision with related data details

3.3.4 Toolbar

Every page includes a toolbar on top. Depending on the page type, the toolbar contains different elements and displays relevant options for the current page. It shows, for instance, the selections of date range, revision, structures, and view mode that are applied and active. In addition, it allows quick access and fast modifications to these options.

Date Range Selection

Selecting a date range filters all visualizations and the *Revision Selection*. If the marked or selected revision is not part of the new date range, the latest revision in that time frame is selected automatically.

The date range selections provides different forms of selections:

- **Everything** (Default)

  All revisions of the project are shown.

  This is the best selection for short projects, to get a full overview of your project.

- **Weeks or Months or Years**

  Select one or multiple weeks, months or years.

  After selecting this form of selection, a week, month or year can be chosen in a date picker.

  If more than one date is needed, additional dates can be added with the [+] button (e.g. 2 weeks for a fortnight).

  The arrow buttons move the date range in one interval: week, month or year.

- **Manual**

  The manual selection allows for a date range based on a specific start and end date. These dates can be chosen with a range date picker.

  The arrow buttons move the date range in the duration chosen (e.g. if a start to end range of 3 months is chosen, the arrows switch to a date range of the next/previous 3 months.)

Revision Selection

One revision is always selected as the active revision for the status visualizations. Trend visualizations show a highlighting of this revision. By default the latest available revision is selected. By marking one or more revisions in a trend visualization, the options in this selection are reduced. Selecting an revision changes the information, that is show in the status visualizations. The revision selection can be overridden in visualization options of status visualizations.

Milestone Visibility Selection

Multiple milestones or milestone sets can be selected as the visible milestones. Trend visualizations show the milestones with vertical lines and a tooltip. By default all milestones are selected.

If there is a tree structure of milestone sets defined (see *Milestone Structures*), the milestone sets can be expanded and collapsed in the selection dropdown by clicking on the arrow left of the checkbox.

- By clicking on the checkboxes multiple milestones and/or milestone sets can be selected or deselected.

- By clicking on the name of the milestone and/or milestone set only the milestone / milestone set is selected and other selections are deselected.

- The milestones selection allows to search and filter the milestones / milestone sets in the dropdown by typing in the selection box.

Structure Selections

Structure selections show the configured structures of the quality model, the project, the measures, and the findings. The structure levels are shown in a tree dropdown, while the real structures are only visible in the visualizations (Heatmaps, Sunburst, Treemaps, Lists and KPIs). When selecting a level in a structure, all elements that are not assigned to this structure are filtered out. By selecting different levels of a structure the visualizations aggregate the quality or availability in a way that respects the weights of the structures and its elements.

The structure selections can be overridden in visualizations that use the related structures.

Finding Bin Structure

This option is only available in the data details page or for data details visualizations. With configuration of the finding bin structure different finding results (from the same or different data sources) can be renamed, sorted and grouped together, to treat similiar results the same way. Switching the active finding bin structure

is possible with a click, or by using the drop down menu. The option *None (original)* shows all findings with their original result type.

Finding Bin Visibility

This option is only available in the data details page. You can use the finding bin visibility to filter the finding results in the KPIs, Heatmap and Treemap. In the List the findings itself are filtered by this selection. Switching the active finding bin visibility is possible with a click, or using the drop down menu.

Active Markings

There are different types of markings that act as filterings for other visualizations. (see *Marking*)

The toolbar item shows 5 categories of markings and informs if there is currently a marking in those marking categories:

- **Revisions**

  Marking in the trend visualizations, used as filtering for Quality Properties or measure Kpi, artifact Kpi, trend and status visualizations.

- **Bins**

  Marking in the bin trend visualization (legend), used as filtering for quality properties or measure Kpi, artifact Kpi, trend and status visualizations.

- **Artifacts**

  Marking in the artifacts Kpi visualization, used as filtering for quality properties or measure Kpi, trend and status visualizations.

- **Quality Properties**

  Marking in the quality properties Kpi visualization, used as filtering for artifacts Kpi, trend and status visualizations.

- **Findings**

  Marking in the finding Kpi visualization, used as filtering for artifacts Kpi, trend and status visualizations.

- **Main**

  Markings in the status or trend visualizations, used as a filtering for the data / data details visualizations on the quality page.

  The Marking toolbar item also allows the resetting of marking categories or all active markings by clicking the reset buttons.

Quality Assessment Scope

There a up to 3 different Quality Assessment Scopes:

- **Absolute**

    Shows the absolute quality.

- **Available**

    Shows the same as *Absolute*, but the missing quality is ignored.

- **Relative** (only available if target values have been configured)

    Shows the relative quality related to target values. (e.g. if the quality for a specific artifact for a specific revision is 80% and the related target values is 80% the relative quality is 100%.)

View (Default / Diff)

Switching the view is possible with a click. The *Default* view shows the quality or availability of the project, while the *Diff* view shows the differences between Revisions or Revisions to the last milestone as Increase or Decrease in percentage points.

Data Import State

This icon shows the current state of the imported data.

If the data is up to date, a green check mark is shown. If the monitoring detects a change in the data source a refresh button allows the manual update of the data.

3.3.5 Marking

By marking you can select one or more elements in visualizations. Elements can be a tile in Kpi, heatmap and sunburst visualizations, a legend item in trend visualizations or a bar in trend and status visualizations. The marked elements are highlighted and a button on the top right of a visualization is enabled, with which the marking can be reset. Keep the Ctrl or Shift button pressed during clicking on an element adds this element to the existing marking, so that you can mark multiple elements.

The purpose of marking elements in a visualization is to filter or highlight related data in other visualizations.

Data Origins

The data origins visualization is available on the quality and data page and shows a list of the imported data reports.

The list of reports in the visualization can be reduced by filtering (see *Filter Panel*) and marking on the quality and data pages. Only the reports from one revisions are visible in the visualization at once. By marking a revision in the trend visualization or using the revision dropdown, the reports of another revision can be viewed. Reports that have been propagated from older Revisions are shown with grey text color.

This visualization shows the origin of the quality or data of the page and can be used to find and directly open the reports of data source for a specific artifact and revision. If additional human readable report files (e.g. html) were detected, they can also be accessed directly inside MQC.
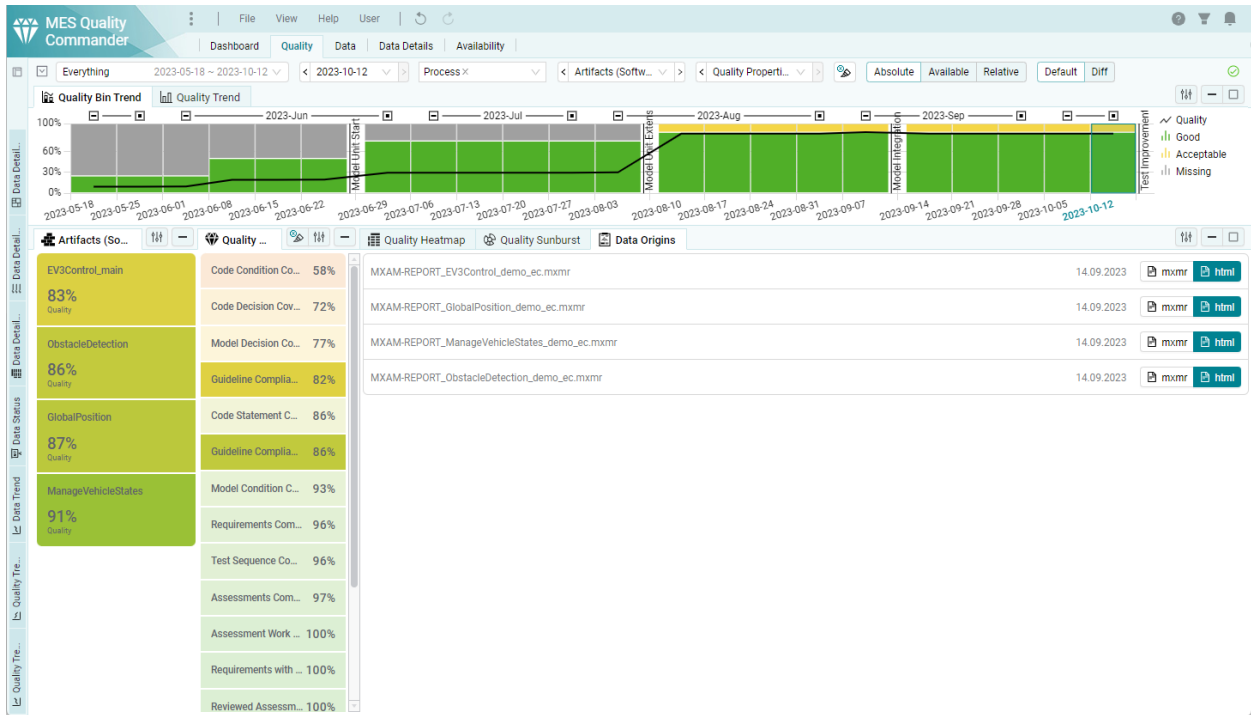
Figure 3.13: Data Origins visualization in the Quality page with reports from MXAM.
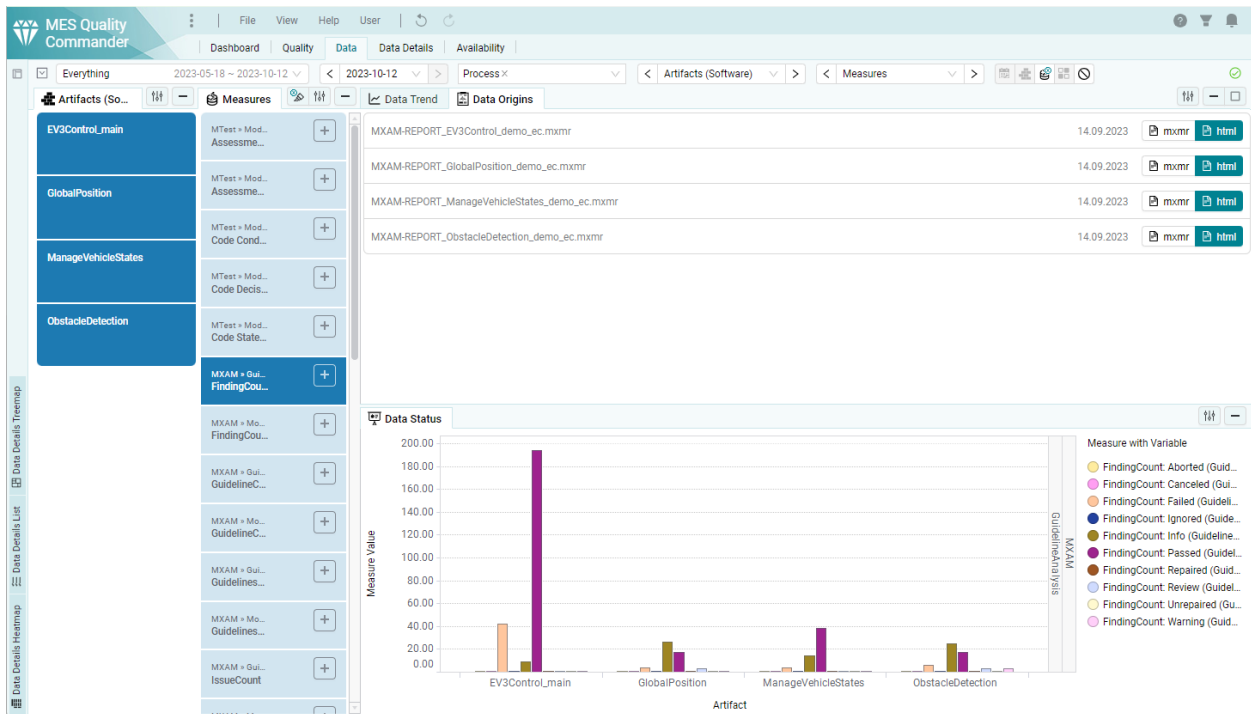

Figure 3.14: Data Origins visualization in the Data page with reports from MXAM.

Data from Quality

The data trend and data status visualization are available in the quality page and make it possible to easily get an overview of the underlying data, that was used to calculate marked quality values.

All markings on quality visualizations like artifacts, quality properties, quality bins and revisions, reduce the shown data in these data visualizations by marking.
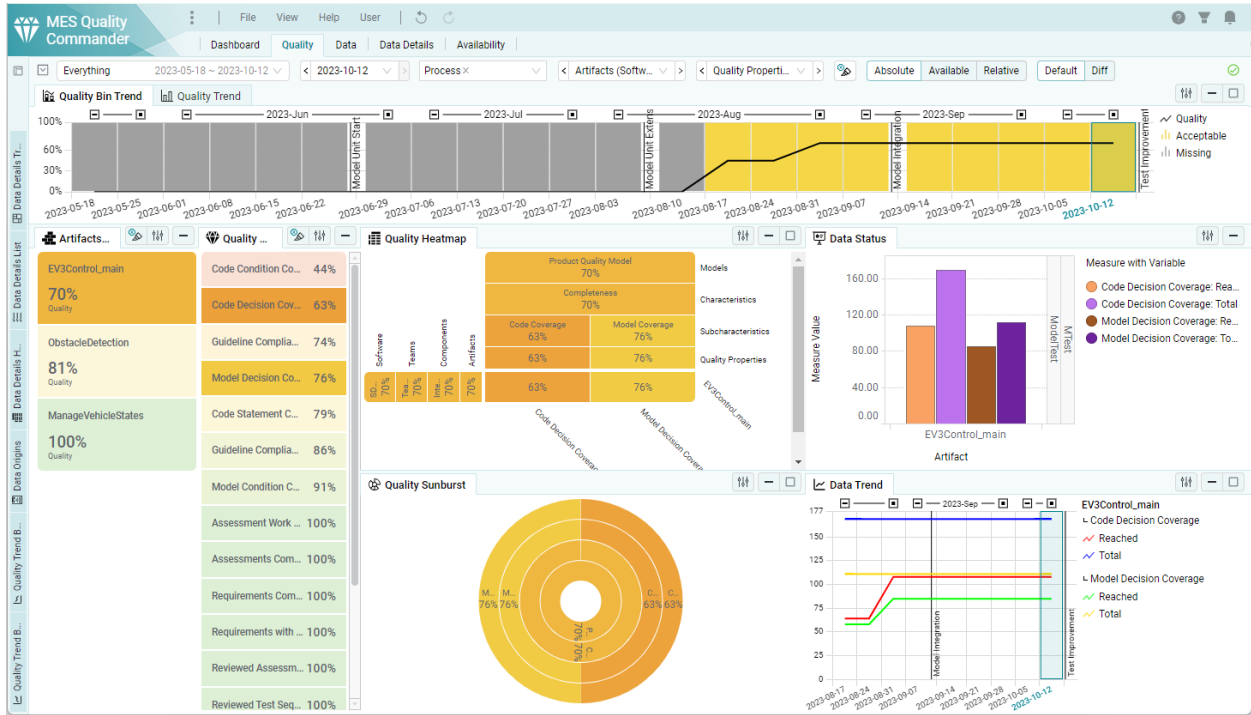

Figure 3.15: Data status and trend visualizations in the quality page to view the underlying data of quality by marking elements in quality visualizations.

As you can see in this picture, with the flexibility to change the layout of the page and marking the intended items, MQC gives you a clear picture of the transformation from data to quality.

Data Details from Quality

The data details trend and status visualizations are available in the quality page. These visualizations give an in depth view on the findings making up the data, that was used to calculate quality.

All markings on quality visualizations like artifacts, quality properties, quality bins and revisions, reduce the shown findings in the data details visualizations by marking.

Data Details from Data

The data details trend and status visualizations are available in the data page as well. Here you can see the details of the data that was marked in the KPI or Data Trend visualizations.
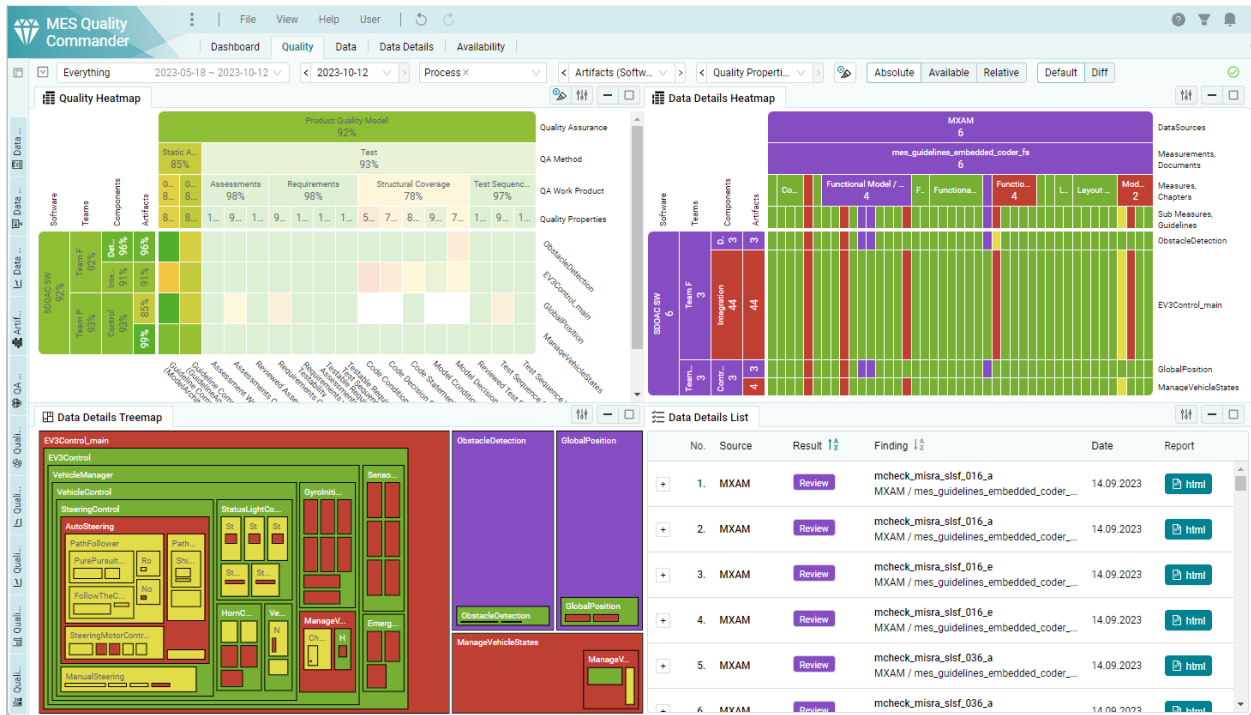
Figure 3.16: Data Details List and Treemap visualizations showing underlying data findings of the marked elements in the quality visualizations.

3.3.6 Display Options

Options that only affect how MQC is displaying information can be changed by any user.

These display options are:

- Toolbar Selections (see *Toolbar*)
- Markings (see *Marking*)
- Active Page
- Active Page Layouts for Quality, Data and Availability pages (see *Page Layout*)
- Visualization Options (see *Visualizations*)

The display options can be restored to the ones the editor saved for the project by using the menu item "View > Restore all display options".

User Preferences

MQC saves the display options per user and per project on the server automatically.

When reopening a project from the library the display options you have chosen previously are applied in the background and the project will have the same state of display options as before.

3.4 VISUALIZATIONS

3.4.1 Trend

The Trend visualizations show all visible revisions in a time based bar / line chart (see *Date Range Selection*).

Visible milestones are shown as vertical lines (see *Milestone Visibility Selection*).

Grouping of revisions is by default set to months. This can be changed with the "Date Groups" display option to Weeks, Months, Quarters or Years. The groups are visible on top of the revisions and can be collapsed [-], expanded [+] and focused on [ ].

Visualization options allow the override of the View selection (normal / diff).

The selected revision for the status visualization is highlighted with the border and name colored blue.

Marking of one or multiple revisions is possible by clicking in the chart area of the visualization.

Bin Trend

The Bin Trend visualization shows stacked bins of Quality, Availability or Data Details. Additionally a line shows the total Quality or Availability.

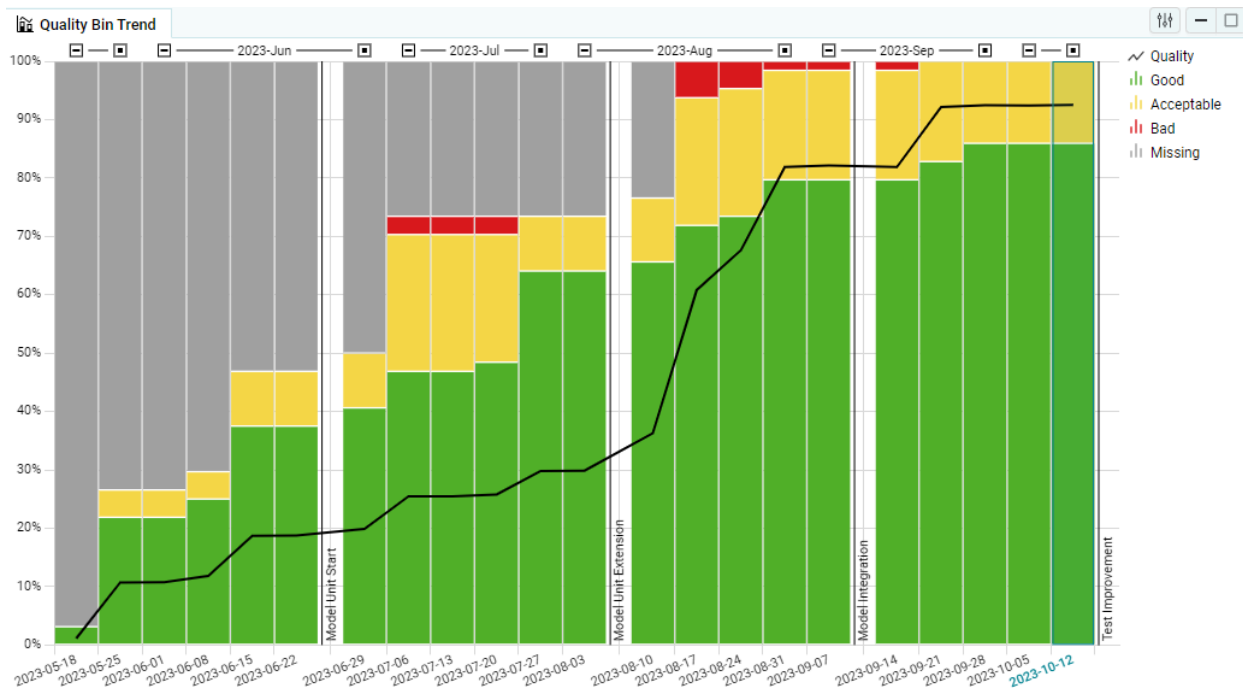The legend allows the marking of one or more bins.

Figure 3.17: Quality Bin Trend on the Quality page as main visualization

The following visualization options allow customizing the bin trend:

- **View (Default / Diff)**

Changes the view for this visualization (see *View (Default / Diff)*).

- **Date Groups**

Changes the date ranges the revisions can be collapsed [-], expanded [+] and focused on [ ].

Quality / Availability Trend

This trend visualization shows the total quality or availability as bars.
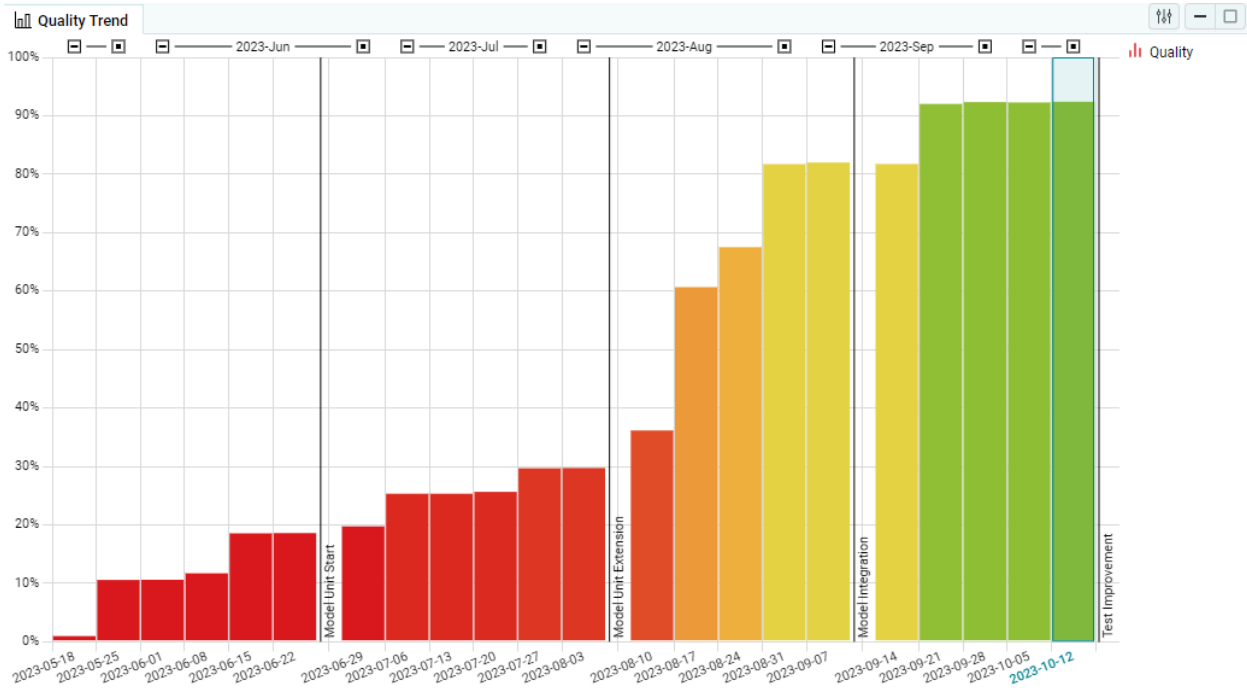

Figure 3.18: Quality Trend on the Quality page as main visualization

The following visualization options allow customizing the quality/availability trend:

- **View (Default / Diff)**

Changes the view for this visualization (see *View (Default / Diff)*).

- **Date Groups**

Changes the date ranges the revisions can be collapsed [-], expanded [+] and focused on [ ].

Quality Trend by Artifacts

This trend visualization shows lines for each artifact that shows their quality.

The legend allows the marking of one or more artifacts.

The following visualization options allow customizing the bin trend:
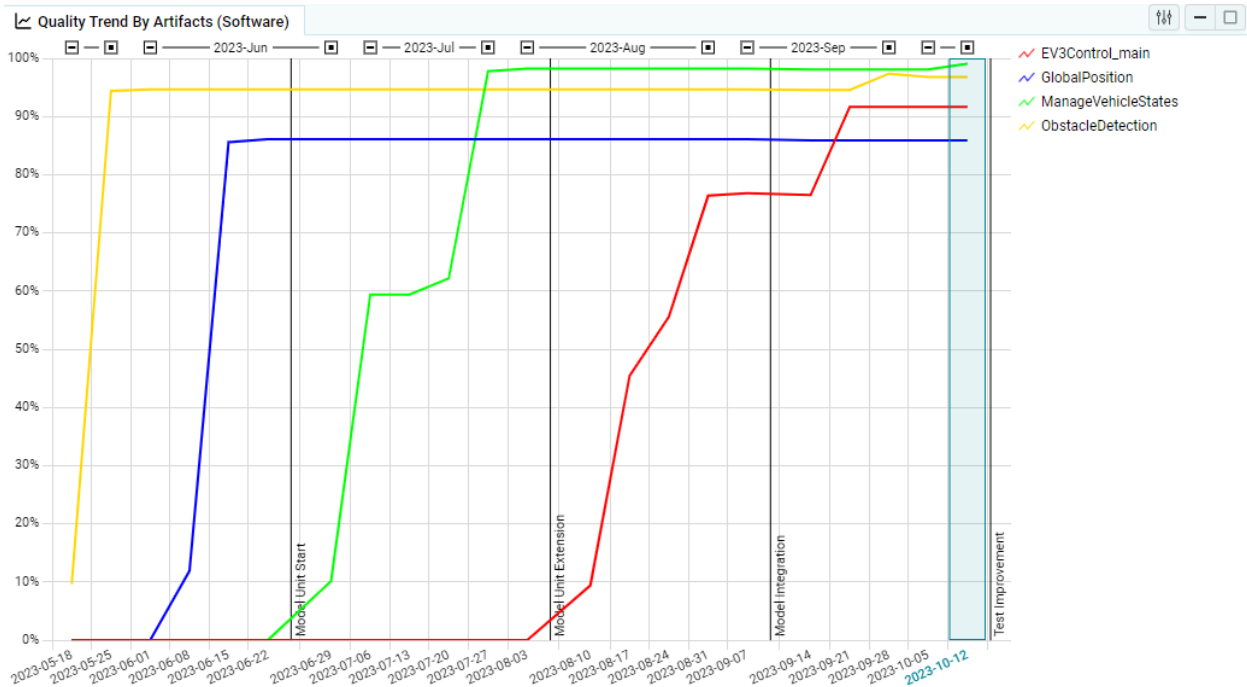
- **Artifact Structure**

Figure 3.19: Quality Trend by Artifacts on the Quality page as main visualization

Changes the artifact structure level that is used here (see *Structure Selections*).

- **View (Default / Diff)**

Changes the view for this visualization (see *View (Default / Diff)*).

- **Date Groups**

Changes the date ranges the revisions can be collapsed [-], expanded [+] and focused on [ ].

Quality Trend By Quality Properties

This trend visualization shows lines for each quality property that shows their quality.

The legend allows the marking of one or more quality properties.

The following visualization options allow customizing the bin trend:

- **Quality Structure**

Changes the quality property structure level that is used here (see *Structure Selections*).

- **View (Diff Mode)**

Changes the view for this visualization (see *View (Default / Diff)*).

- **Date Groups**

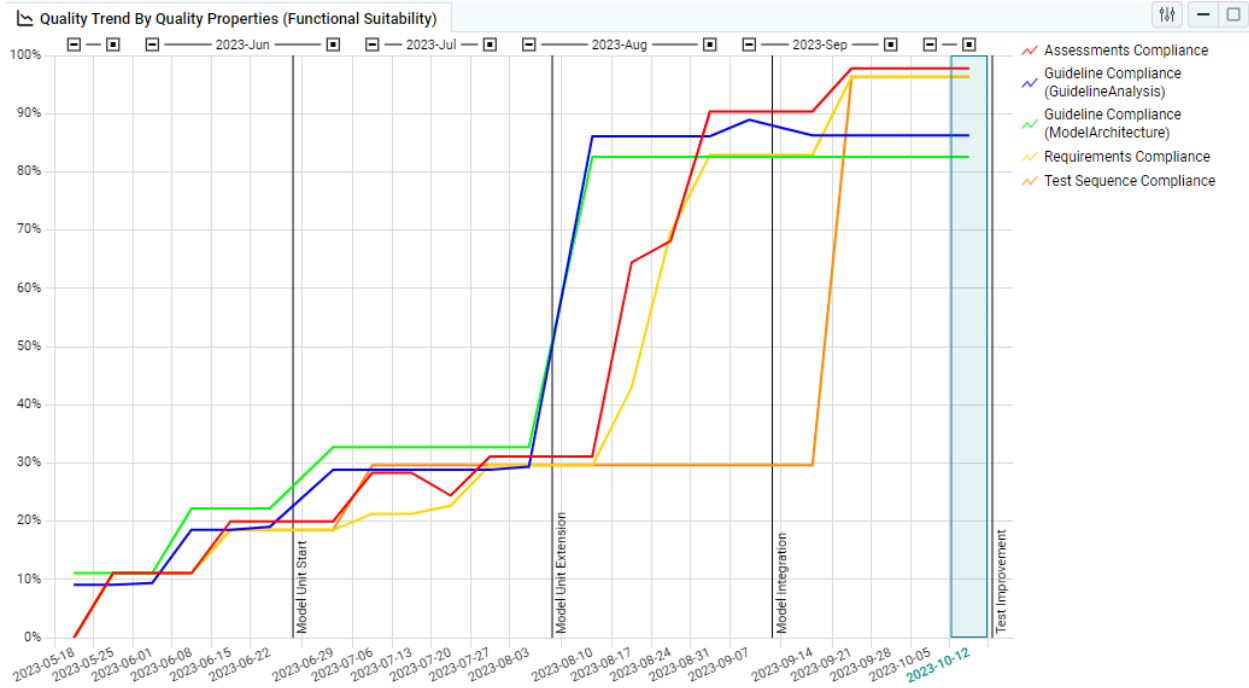Changes the date ranges the revisions can be collapsed [-], expanded [+] and focused on [ ].

Figure 3.20: Quality Trend By Quality Properties on the Quality page as main visualization

Data Trend

The data trend visualization shows lines for every measure variable for each artifact for one measure. The lines are colored in adjacent colors for each artifact.

You can highlight specific measures by selecting them in the legend, or you can reduce their number to the ones of interest by selecting their corresponding elements in the KPIs.

The legend groups the measure variables by the visible artifacts and allows the marking of one or more artifact + measure variables.

The following visualization options allow customizing the bin trend:

- **Artifact Structure**

  Changes the artifact structure level that is used here (see *Structure Selections*).

- **Date Groups**

  Changes the date ranges the revisions can be collapsed [-], expanded [+] and focused on [ ].

3.4.2 Status

The Status visualizations show the Quality, Availability or Data Details of the selected revision (see *Revision Selection*).

Visualization options allow the override of the Artifact, Measure, Quality Property and/or Finding Structures and the View selection (normal / diff).
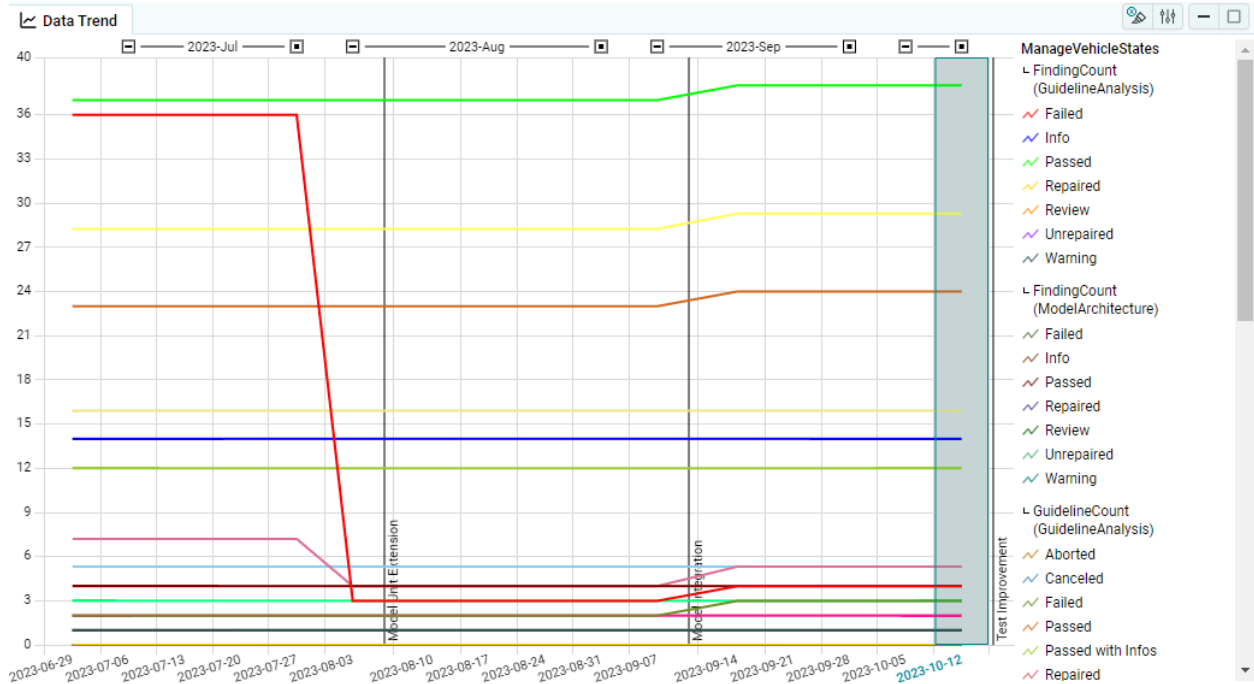
Figure 3.21: Data Trend on the Quality page as main visualization

It is possible to mark elements in quality, data or data details by clicking in the chart area of the visualization.

Heatmap

The Heatmap shows the Quality or Availability with a Matrix of Artifacts x Quality Properties, Artifact x Measures or Artifact x Findings based on the Structure selections.

The selected Artifact Structure is shown with aggregated values on the left and the Quality Property Structure Measure Structure or Finding Structure is shown with aggregated values on the top.

The following visualization options allow customizing the heatmap:

- **Artifact Structure**

   Changes the artifact structure level that is used here (see *Structure Selections*).

- **Quality Structure**

   Changes the quality property structure level that is used here(see *Structure Selections*).

- **View (Diff Mode)**

   Changes the view for this visualization (see *View (Default / Diff)*).

- **Weights**

   Indicates whether the columns and rows of the heatmap be sized according to the defined artifact and quality property weights.
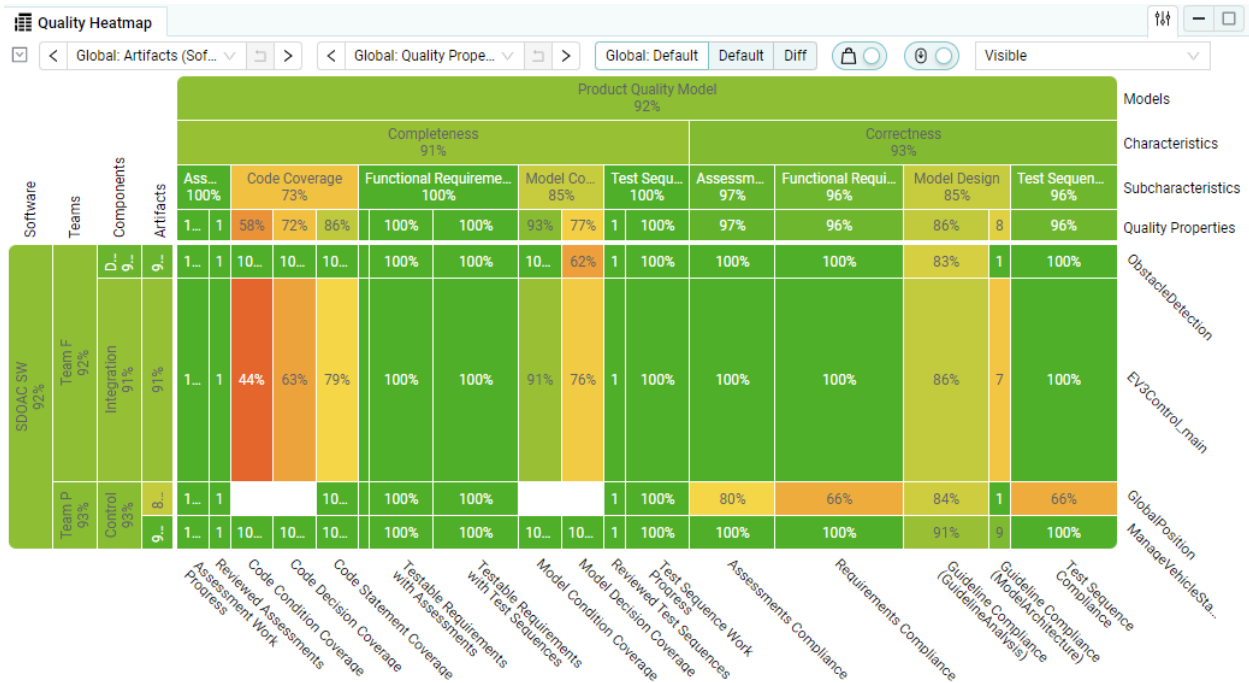
Figure 3.22: Quality Heatmap on the Quality page as main visualization

(this option has no impact on the calculation of the quality aggregation, the weights are always used for the calculation)

- **Scrolling**

  Indicates whether the visualization have a minimum height for the rows and show a vertical scrollbar if there are too many artifacts to show at once.

- **Labels**

  Indicates which and whether the labels/values are displayed.

Sunburst

The sunburst shows a aggregated view on the quality property structure for quality and measure structure for availability.

The following visualization options allow customizing the sunburst:

- **Quality Structure**

  Changes the quality property structure level that is used here (see *Structure Selections*).

- **View (Default / Diff)**

  Changes the view for this visualization (see *View (Default / Diff)*).

- **Interaction**

Figure 3.23: Quality Sunburst on the Quality page as main visualization

Defines what will happen when clicking on an element in the chart area. By using Alt+Click the not selected option (Drill-Down or Marking) can be achieved without changing the interaction.

- **Labels**

  Indicates which and whether the labels/values are displayed.

- **Values**

  Indicates whether the values are displayed. (avaible for Quality or Availability)

Data Details Treemap

The Data Details Treemap shows the findings in a structured way. This can be, for example, their artifact substructure (default), which is usually the structure of the underlying software to which they refer.

The color of the structure elements and levels corresponds to the color of the result with the worst impact on the overall result. A tooltip shows more detailed information about the numbers and result types of the related findings.

The following visualization options allow customizing the data details treemap:

- **Finding Bin Structure**

  Changes the finding bin structure that is used for the result types in this visualization (see *Finding Bin Structure*).

- **View (Diff Mode)**

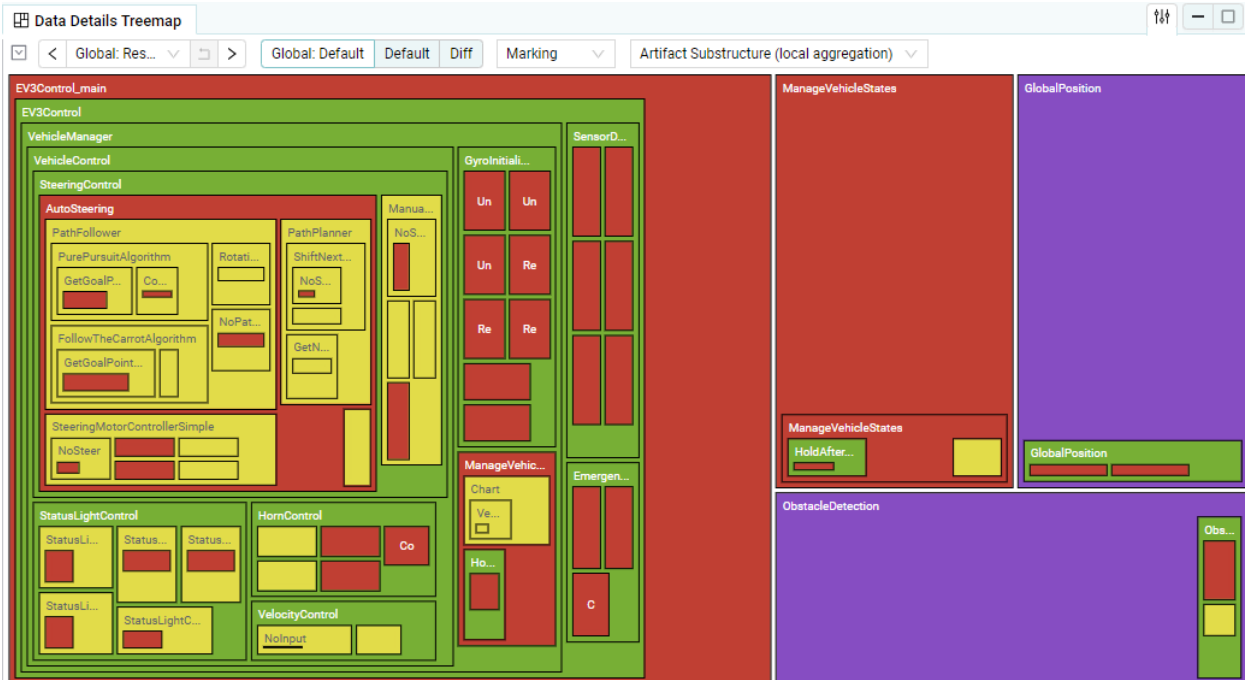 Changes the view for this visualization (see *View (Default / Diff)*).

Figure 3.24: Data Details Treemap

- **Interaction**

  What should happen when clicking on an element in the chart area. By using Alt+Click the not selected option (Drill-Down or Marking) can be achieved without changing the interaction.

- **Structure**

  Indicates whether the Treemap shows the findings structured by the artifact sub structure or by the full finding structure.

  If "Artifact Substructure" is selected, the worst result will not aggregate to higher structure levels. If "Finding Structure" is selected, the worst result will aggregate to higher structure levels.

Data Details List

The Data Details List shows the imported findings in a table.

You can sort the list by clicking on the column headers. The previous selection is always used as the secondary sorting direction, which makes it possible to sort by two columns.

By pressing the '+' button on a row you can view further details of the finding. If a human readable report file is available, a button is shown that opens the file. An html report scrolls to the position of the finding, if possible.

To ensure good performance, the number of findings displayed in the Data Details List is limited in large projects. By using grouping (see option below) or marking in other visualizations (see *Marking*) only the relevant findings will be shown.

Figure 3.25: Data Details List with expanded details

The following visualization options allow customizing the data details list:

- **Finding Bin Structure**

  Changes the finding bin structure that is used for the result types in this visualization (see *Finding Bin Structure*).

- **View (Default / Diff)**

  Changes the view for this visualization (see *View (Default / Diff)*).

- **Group By**

  Indicates whether the findings should be grouped by a specific criteria (e.g. by artifact).

Data Status

The Data Status shows the measure values as a bar chart for the selected measure.

The following visualization options allow customizing the data status:

- **Artifact Structure**

  Changes the artifact structure level that is used here (see *Structure Selections*).

Data Origins

The Data Origins show the imported data source files with human readable files, if present. By clicking on the report file you can open and view the corresponding file directly from MQC.
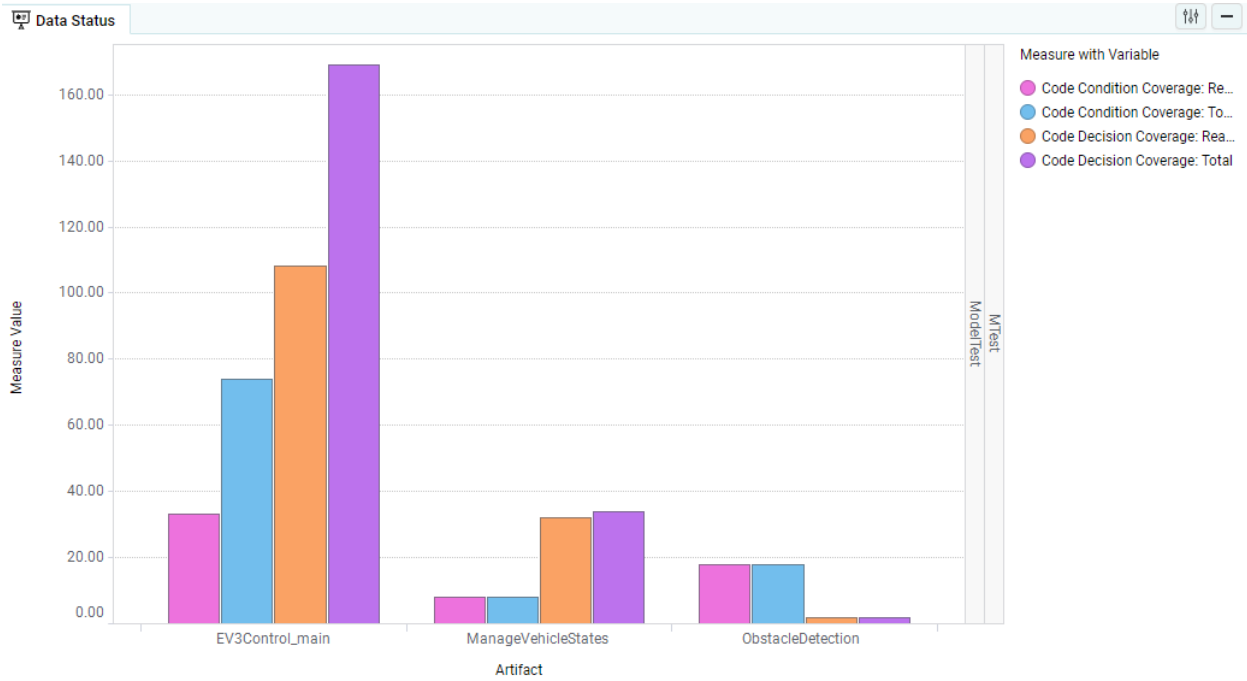
Figure 3.26: Data Status on the Quality page as main visualization

Figure 3.27: Data Origins on the Quality page as main visualization

3.4.3 KPI

The Key Property Indicator visualizations show the Artifact Structure, Quality Property Structure, Measure Structure or Finding Structure for the selected revision. The Quality or Availability values are aggregated. In the Data Details page the KPI-elements show the worst result and the number of findings per result.

When a level of the structure is selected the visualization allows the expanding and collapsing of the sub-levels by clicking on the [+] or [-] buttons.

Marking of one or multiple Artifacts, Quality Properties or Measures is possible by clicking on them in the chart area of the visualization.
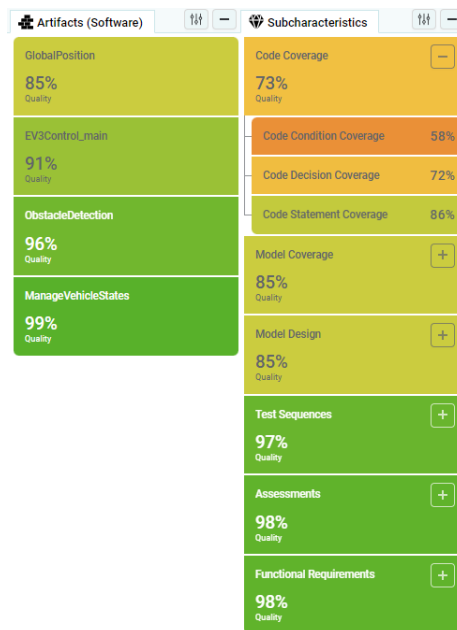
Figure 3.28: KPI visualizations of Artifacts and Quality Properties on the Quality page

Visualization options allow the override of the Artifact, Measure, Quality Property or Finding Structures and the View selection (normal / diff).

The following visualization options allow customizing the kpi:

- **Structure Level**

  Changes the structure levels that are used here (see *Structure Selections*)

- **View (Diff Mode)**

  Changes the view for this visualization (see *View (Default / Diff)*).

- **Search**

  Filter the visible elements by searching.

- **Sorting**

  In which order the elements should be shown.

3.5 DIFF VIEW

Use the Diff View to see how the state of your project has changed since the last revision.

Advantages of the Diff View are:

- to focus only on the elements that have changed instead of your entire project
- to identify causes of negative developments more quickly
- to check whether the project is progressing steadily

The Diff View can be enabled via the option in the toolbar for all visualizations of the project at once in which this mode is supported (see *View (Default / Diff)*) or in a specific visualization in order to activate the view locally in this visualization only (see *Visualizations*).

Depending on the page and visualization, the Diff View can give you different kinds of information. The changes are shown as increases and decreases for availability or improvements and deteriorations for quality. On the Data Details page the changes are shown as new and gone in the trend and list, while the kpis, heatmap and treemap show major improvements, improvements, minor improvements, major deteriorations, deteriorationsand minor deteriorations. *'''*

3.5.1 Quality Diff

Figure 3.29: Quality Page in Diff View

KPIs and Status Visualizations

The KPIs and status visualizations like the Heatmap and Sunburst have the same behavior as in the Default view. But instead of showing the quality in %, they show the changes of quality in percentage points (pp). Improvements are represented by a green color and deterioration by a red color. The color intensity shows the magnitude of the change.

Trend Visualizations
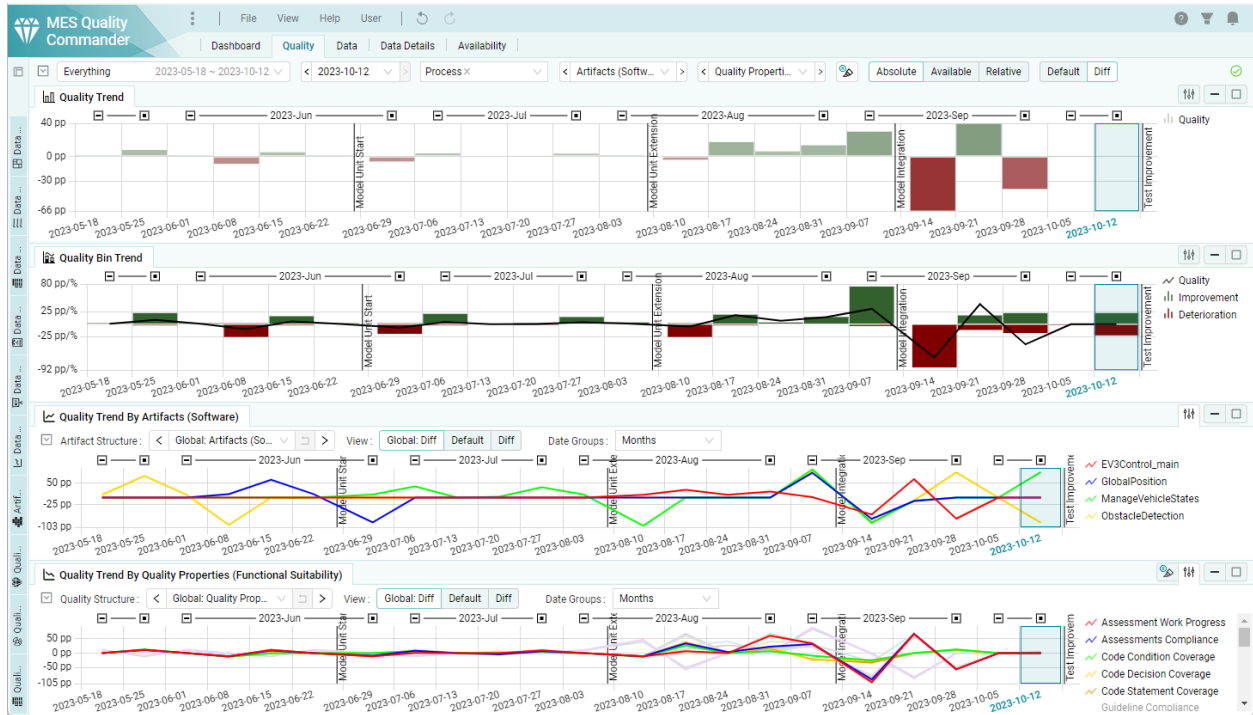
Figure 3.30: Different Quality Trend visualizations in Diff View

- **Quality Trend**

  In Diff View the Quality Trend shows the increase (green) and decrease (red) of the absolute quality in percentage points (pp, see Wikipedia) since the last revision or milestone.

- **Quality Bin Trend**

  The Quality Bin Trend shows how many quality points (per Revision x Artifact x Quality Property) have a improvement or deterioration in percent. The percentages with an improvement are shown in green and as positive and those with a deterioration are shown in red and as negative value.

  Additionally the changes in the total quality is shown as percentage points with a black line.

- **Quality Trend By Artifacts**

  The Quality Trend By Artifacts visualization shows the increase and decrease of the quality of each element of the selected artifact structure level as lines.

- **Quality Trend By Quality Properties**

  The Quality Trend By Artifacts visualization shows the increase and decrease of the quality of each element of the selected quality property structure level as lines.

3.5.2 Data Details Diff
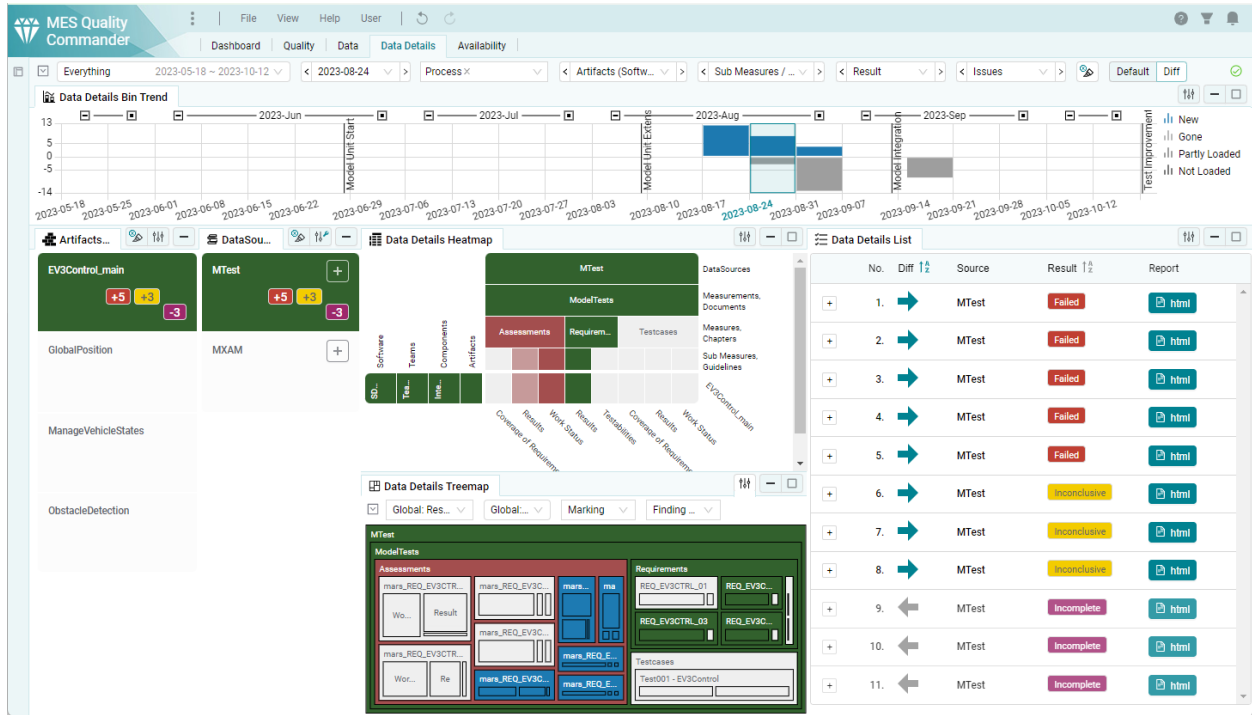
Figure 3.31: Data Details Page in Diff View

- **Data Details Bin Trend**

  In Diff View the Data Details Bin Trend shows the changes in the number of findings. New findings are shown in blue and as positive and gone findings are shown in grey and as negative.

- **KPIs**

  In the KPIs the changes are shown aggregated for the related structure element. The color of the KPI element and its intensity represents the general improvement (green) or deterioration (red). The number of finding changes are listed by the colors of their results, that are new (+) or gone (-).

- **Data Details Heatmap / Treemap**

  The Data Details Heatmap and Treemap show improvements (green) and deteriorations (red), as well as new (blue) and gone (grey) available data. Improvements and deteriorations are further broken into major (the worst result changed), normal (the number of finding results of the worst result changed) and minor (the number of finding results of not the worst result changed).

- **Data Details List** The Data Details List contains a "Diff" column while the Diff view is active. The new column shows if a finding is New (blue right arrow) or Gone (grey left arrow). Unchanged findings are

not shown at all.

3.5.3 Availability Diff
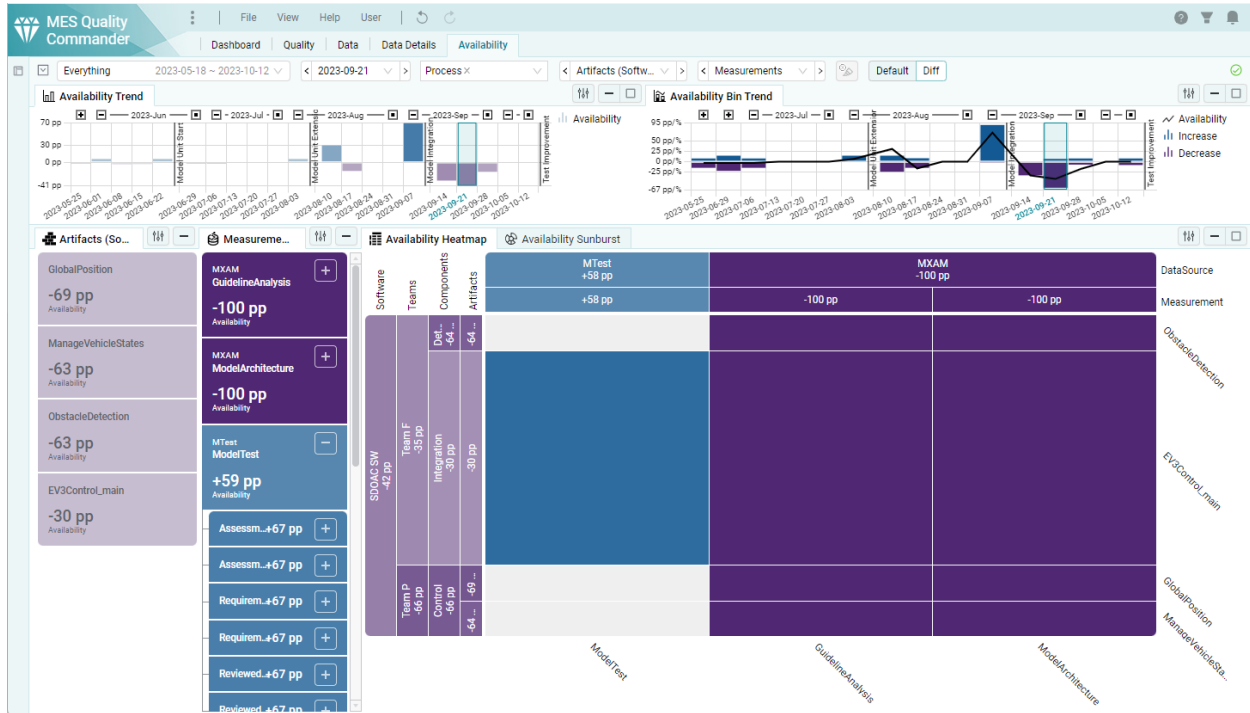
Figure 3.32: Availability Page in Diff View (Example project without propagation enabled)

The increase in available data is shown in blue and the decrease is shown in purple. In all other aspects, the behavior of the visualizations on the Availability Page is the same as on the Quality Page.

3.6 DASHBOARDS

The *Dashboard* page gives a first overview of the current status of the most important aspects of the project at a glance. It can be used as a starting point for quality evaluation and to explore the details of quality deficits in your project. MQC provides a default dashboard page (see Figure 3.33), which can also be customized based on your demands (see *Customization*).

Each visualization in the default dashboard shows a particular aspect of the current project in a compact way, leaving out any details that can be found in the main visualizations of the MQC standard pages. For that purpose, MQC provides a link to the corresponding detail page in the top-right corner of each visualization.

Marking does not apply to the dashboard pages. But instead, filtering the data via the filter panel (see *Filter Panel*) at the right-hand side may be used to concentrate on specific aspects.

Figure 3.33: Default Dashboard page

3.6.1 Customization

Utilizing the buttons on the left hand side you can easily customize the current Dashboard Page:

1. **Enable customize dashboard**

   By clicking **Enable customize dashboard** you are able to resize a visualization by hovering over the boundary of the visualization and drag it to obtain the desired size, move a visualization by drag and drop to another empty space or delete a visualization by hovering the visualization and clicking the red button at the top right corner.

2. **Add dashboard visualization**

   Open the **Add Dashboard Visualization** dialog with different groups of tiles (Project, Quality, Availability, Action and Data). A visualization can be added by selecting the visualization and clicking **Add**. If the minimum size of the tile cannot fit to the available empty spaces then the tile cannot be added.

3. **Restore dashboard**

   Restore the initial Dashboard configuration defined in the Dashboard Configuration Source.

4. **Remove all visualization in dashboard**

   Removes all visualizations from the current Dashboard page.
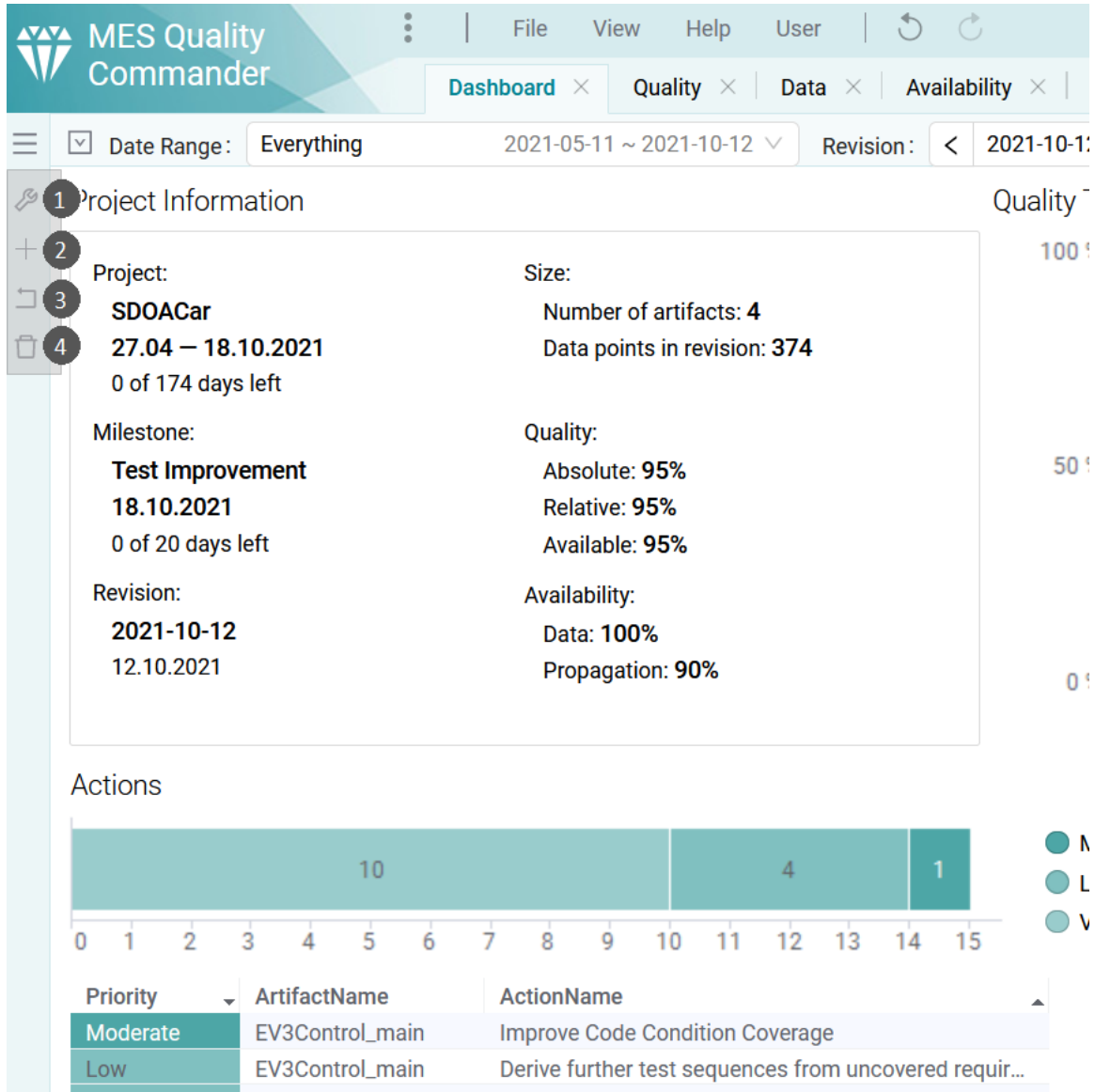
Figure 3.34: Buttons for customizing the Dashboard
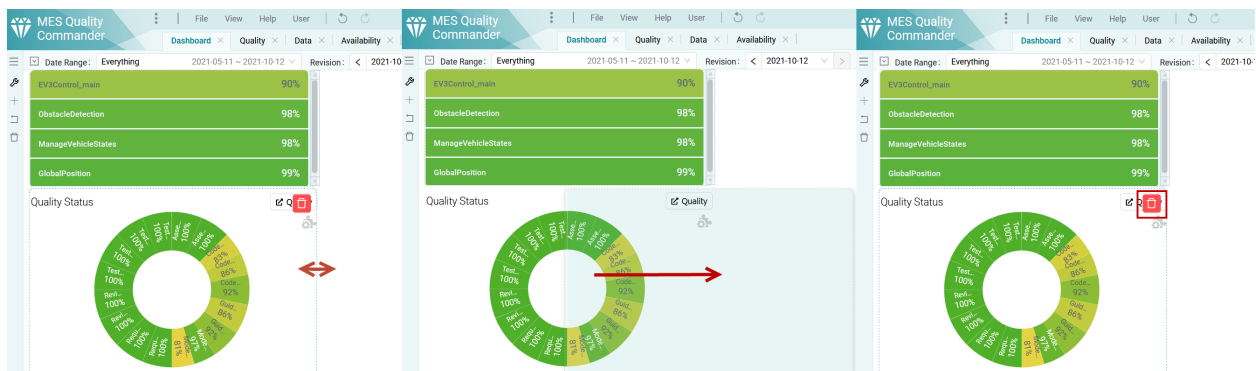
Figure 3.35: Resize (left), move (middle) and delete (right) visualization on Dashboard

Figure 3.36: Add Dashboard Tile dialog with tabs for different visualizations grouped by type.

3.7 CUSTOM PAGES

Custom pages as shown in Figure 3.37 are pages created by an editor of MQC to focus on specific informa-tion, base measures, derived measures, or quality properties.
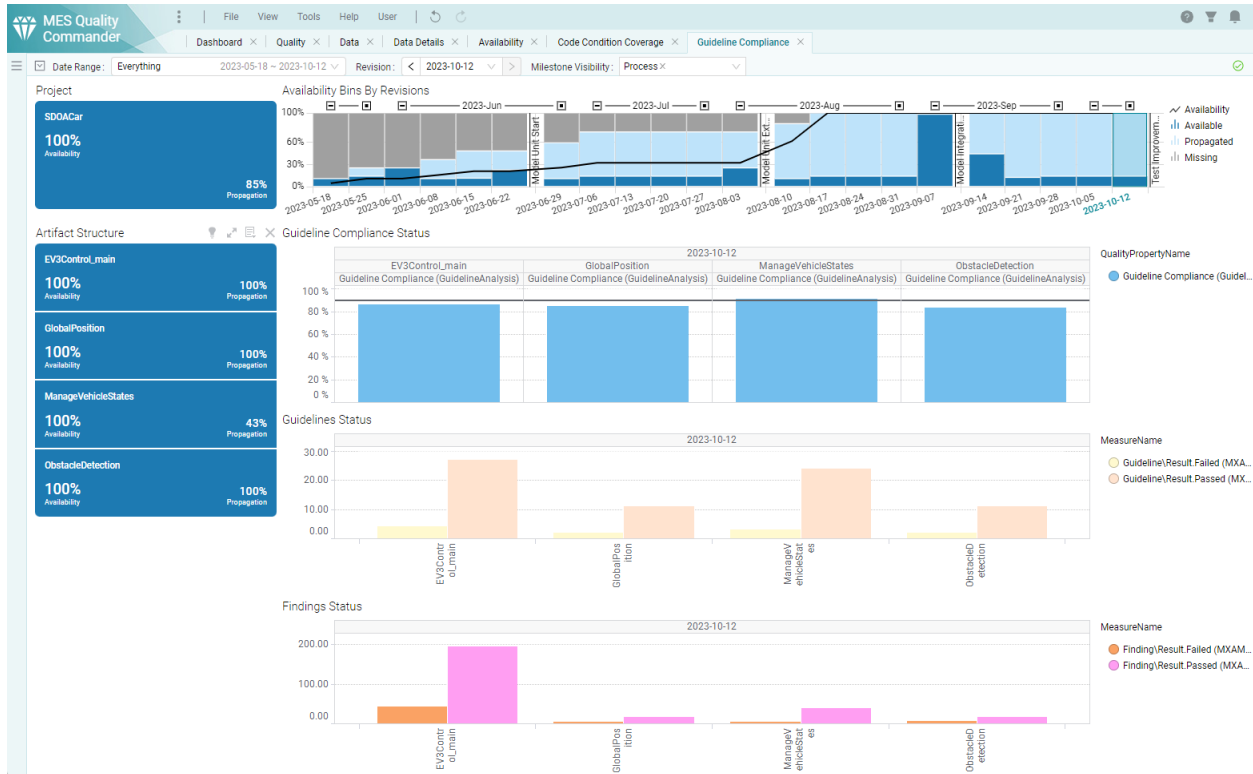
Figure 3.37: Custom page showing selected base measures and quality properties

Each custom page may show:

- multiple visualizations: trend or status

- multiple measures per visualization: quality or data

- combinations of quality and data visualizations

- expectations (targets) per measure - as separate dashed lines with the same color in trend visualiza-tions - as horizontal lines in status (bar) visualizations

- a label per measure depicting the actual value of that measure

An Artifact KPI visualization on the left-hand side may be used to focus on specific artifacts.

The toolbar for custom pages consists of the *Date Range Selection* and the *Revision Selection*.

3.8 FILTER PANEL

The filter panel opens by clicking on the filter icon in the upper right corner (see Figure 3.38). It can be used to reduce the data underlying the visualizations and metric calculations in MQC to a specific selection.

Filtering is also possible by deselecting elements of the artifact and quality structures within their KPI visualizations (read more about marking in Section 3.3). However, this is limited to the currently active levels of these two structures of these two types of structures.

The filter panel, on the other hand, allows filtering for all defined structures of the current project. In addition to all structures for the artifacts and quality properties, these are also the milestone and data structures. If several structures are defined, e.g. for the artifact or the quality properties, the filtering can be carried out here on the basis of the different structures at the same time.

By deselecting individual structure levels within the filter panel, all underlying data and quality values assigned to this level and the structure below it will no longer be taken into account when displaying the project. This is also independent of which structure settings have been made in the individual pages. Filtering affects the entire project and influences all visualizations on every page.

Figure 3.38: Example project with open filter panel

It should be noted that the deselected elements are filtered out even if they are still enabled in other parts of the filter panel. The filter selection is therefore to be understood as with an AND condition.

For an user with editor permissions, the selection of the filter panel is saved within the project. Otherwise, the filter selection is only temporary until the project is closed.

The advantage of filtering with the filter panel is, that it is a more permanent application than using the marking, which can also be used together with it. With the filter panel you can hide all the parts of the project you don't want to focus on and, together with the marking, it helps you look at details more closely. In addition, the filter panel provides a quick overview of the entire structure of the project at any time.

3.9 REPORT

MQC allows the creation of a status report summarizing the most important data in visualizations and tables. This report is designed for browsing (HTML) and print (PDF).

To create a new Report, go to *File > Create Report* in the MQC menu.

It is important to note that the active settings from the filter panel are applied for the creation of the report, opening up a more in depth customization of the displayed data.

Active markings, in contrast to filters, are not applied in the creation of the report.

3.9.1 Options

The report configuration dialog allows an extensive configuration of the contents contained in the report.

Artifact Selection

The artifacts to be included in the report can be selected either by Quality Bin (please refer to *Bins*) or by Quality Order. If you select by Quality Order you can choose a number of Artifacts with the best and/or worst aggregated quality that will be contained in the report.

Revision Selection

You can select the revision, the report will be created for, to be the last revision or a specific one.

Report Type

Three different report types are available to choose from: Full, Compact, and Short. Every report type displays relevant data and visualizations of the following categories.

Table 3.1: Included content in the report for the different report types

	Full	Compact	Short
Quality Overview	✓	✓	✓
Data Availability Overview	✓	✓ (*)	✓ (*)
Data Trend Overview	✓	-	-
Data Status Overview	✓	-	-
Artifact Details	✓	✓	-
Quality Property Details	✓	✓	-
User Configured Custom Pages	✓	✓	✓
Configuration Information	✓	-	-
Annotations	✓	✓	-

* Does not contain the "Missing Base Measures per Artifact" table.

Figure 3.39: The Create Report dialog allows the configuration of specific settings for the report.

Output Format

You can chose between two different output formats namely PDF or HTML.

After clicking on *Create* the report will open within the default viewer on your platform, where you can save it from.

3.9.2 Content

Report and Project Information

The Report Information shows the configurations of the report and the creation date. In addition it contains the project name, information regarding the due date of your milestones, the overall availability value and overall quality value.

The Project Information is a summary of the project for the selected revision.

Quality Overview

This section contains the most important visualizations and information of the Quality page in MQC. Additionally, quality status information for the selected revision per Artifact as well as per Quality Property are available in tables.

Data Availability Overview

In the Data Availability Overview the Availability Distribution shows the data availability over time.

The Availability Matrix and the Availability Status is shown only for the selected revision of the report.

Special attention is paid to missing data. Not yet imported base measures are listed per artifact and data source. Please note, this information is included only when the report type "Full" is selected.

Data Trend Overview

For the Data Trend Overview the measure trend by artifact visualizations present the base measure values per artifact across all revisions.

Data Status Overview

The Data Status Overview provides a data status visualization for each data source for the selected revision.

Artifact Details

The Artifact Details section shows, per artifact, the Quality Trend as well as the Quality Status for all quality properties. The Quality Status is only shown for the selected revision.

Additionally a table for the selected revision with the quality property values is included, which can also be used to directly navigate to the corresponding part of the Quality Property Details.

Quality Property Details

The Quality Property Details section shows the Quality Trend by Artifact as well as the measurement function, which is used to calculate the quality from the imported data for each quality property.

Additionally a table is shown that displays the data for all measures inside the measurement function for each artifact, as well as the file source from which the data was read. The table cells inside this table are either displayed white for available, blue for propagated or grey for missing.

Much like in the Artifact Details, this table can be used to directly navigate to the corresponding part of the Artifact Details section.

User Configured Custom Pages

The user configured custom pages are dependent on the configured pages in MQC (for more details refer to *Custom Pages*). The status and trend visualizations for each page are shown.

Configuration Information

This section provides an overview for all imported configurations divided into Quality Configurations, Structure Configurations and Other Configurations.

The Quality Configurations include:

- the Quality Model Configuration
- the Actions Configuration
- the Derived Measure Configuration
- the Base Measure Configuration
- the Context Categories Configuration
- the Quality Bin Configuration.

The Structure Configurations include:

- the Project Milestone Configuration
- the Artifact Structure Configuration (incl. Context Categories and Artifact Mapping).

Other Configurations include the user configured Custom Page Configuration.

Annotations

The Annotations section contains all Annotations that are valid for the selected revision. The Annotations are grouped by Artifact.

4 CONFIGURATION

4.1 CREATING AN MQC PROJECT

There are two options, how to set up an MQC project:

- *Creating a project from scratch*

- *Creating a project using a Setup Configuration*

4.1.1 Creating a project from scratch

To create a project from scratch after opening MQC, you have two options (see Figure 4.1).

- Either open the **Files and data** dialog on the left-hand side (click on the + in the top left corner) and choose `Create new Project`.

- Or use the `Create Project` option within the **Getting Started** section inside the main window of the MQC landing page (only available if MQC has been newly started).

A dialog (Figure 4.2) opens asking for the location of your data (see *Data Sources*) and allows to change the revision granularity used (see *Revisions*).

After confirming the dialog using the `Create` button, a new project is set up with the data contained in your data source(s) and defaults for all remaining settings (see *Settings*).

All default settings used for project creation may be changed later via the settings dialog (see *Settings*).

In case you would like to adapt further settings, e.g. *Switch on Propagation*, add a quality model configuration (see *Quality Model*) or project structure (see *Project Structure*), you can switch to the advanced mode of the Create Project dialog as shown in Figure 4.3 by using the switch in the top-right corner.

The advanced mode of the Create Project dialog allows you to load all configuration files and to adapt all necessary *Settings* to be already applied during project creation. This facilitates a fast and comprehensive setup process.

In case you switch back from advanced to simple mode, nevertheless all your previously made settings will be kept and applied when creating the project.

If you confirm the **Create Project** dialog without selecting a data source, an empty project is created. This might be useful in case you don't already have data for your project, but would like to do all necessary configurations in advance.
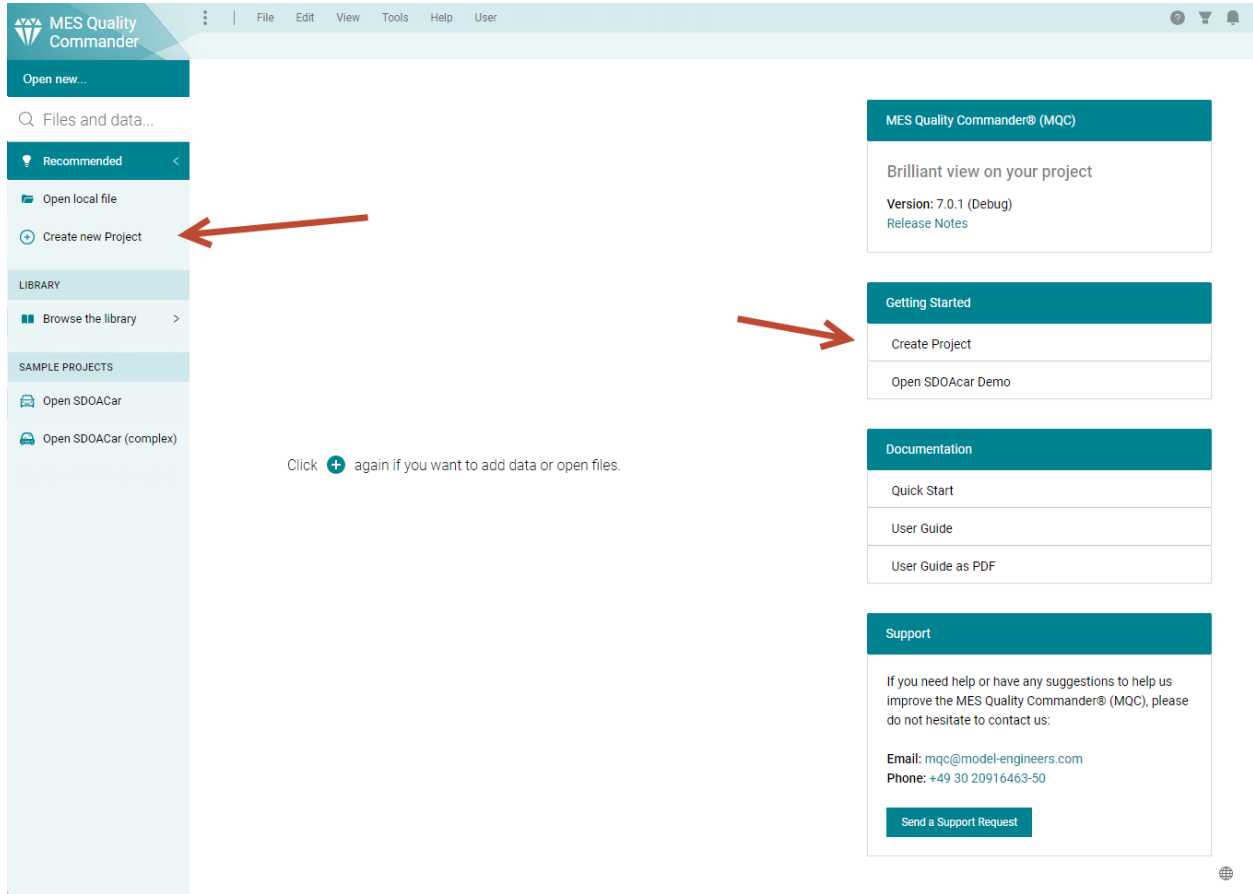
Figure 4.1: Start creating a new MQC project via the Create new Project entry in the left-hand side panel or via the menu entry in the Getting Started section inside the main window

Figure 4.2: Create Project dialog (simple mode) to initially configure the data source(s) and to set the revision granularity

Data sources can be added at any later point in time (see *Data Sources*).

4.1.2 Creating a project using a Setup Configuration

MQC provides the possibility of importing a setup configuration (see *Importing a setup configuration*) to easily set up a project. This configuration can be created in two ways:

- *Exporting a setup configuration from an existing project*,
- *Manually configuring a setup configuration*.

Exporting a setup configuration from an existing project

In MQC you can export a setup configuration file, which contains all the settings done in the current project in order to use it as backup for a fast re-creation of the current project or to use it as a template for creating other similar projects.

Open the **Settings** dialog via the configuration menu and select the `Setup Configuration` button to export the current configuration as Setup Configuration (see Figure 4.5).

Manually configuring a setup configuration

For the setup configuration the user can define:

- files and directories containing reports of tools/data sources to be imported into MQC

Figure 4.3: Create Project dialog (advanced mode) to import data source(s), quality model(s), project structure and to change the settings before creating a project.

Type	Value
Adapters	MxamMxmr,MtestMqcXml
Data	Samples\SDOACar\Data
ProjectStructure	Samples\SDOACar\Config\08_ProjectStructures_ComplexStructures.yml
QualityModel	Samples\SDOACar\Config\09_QualityModel_ComplexStructures.yml
TargetValues	Samples\SDOACar\Config\06_TargetValuesPerMilestone.xlsx
PageLayouts	PageLayouts\QualityPageLayout.yml
PageLayouts	PageLayouts\DataPageLayout.yml
PageLayouts	PageLayouts\AvailabilityPageLayout.yml
PageLayouts	PageLayouts\DataDetailsPageLayout.yml
PageLayouts	PageLayouts\PageLayoutTemplates.yml
Dashboards	Dashboards\DefaultDashboardSource.yml
PagesDashboard	Dashboard
PagesInteractive	Quality,Data,Data Details,Availability
PagesLegacy	
PagesAction	
PagesCustom	
AutomaticDataRefreshServer	true
ContextCategoriesEnabled	true
EmptyRevisionsVisible	false
MeasureValueLabelsVisible	false
Propagation	Project
QualityBin	Good #50AF28 100
QualityBin	Acceptable #F4D646 80
QualityBin	Bad #D8181C 20
RevisionGranularity	Days
RevisionGranularityCulture	en-GB
DiffForMilestones	
DataDetails	All
DataDetailsLastRevisionsNum	3
TargetValuesVisible	true

Figure 4.4: Sample of setup configuration file
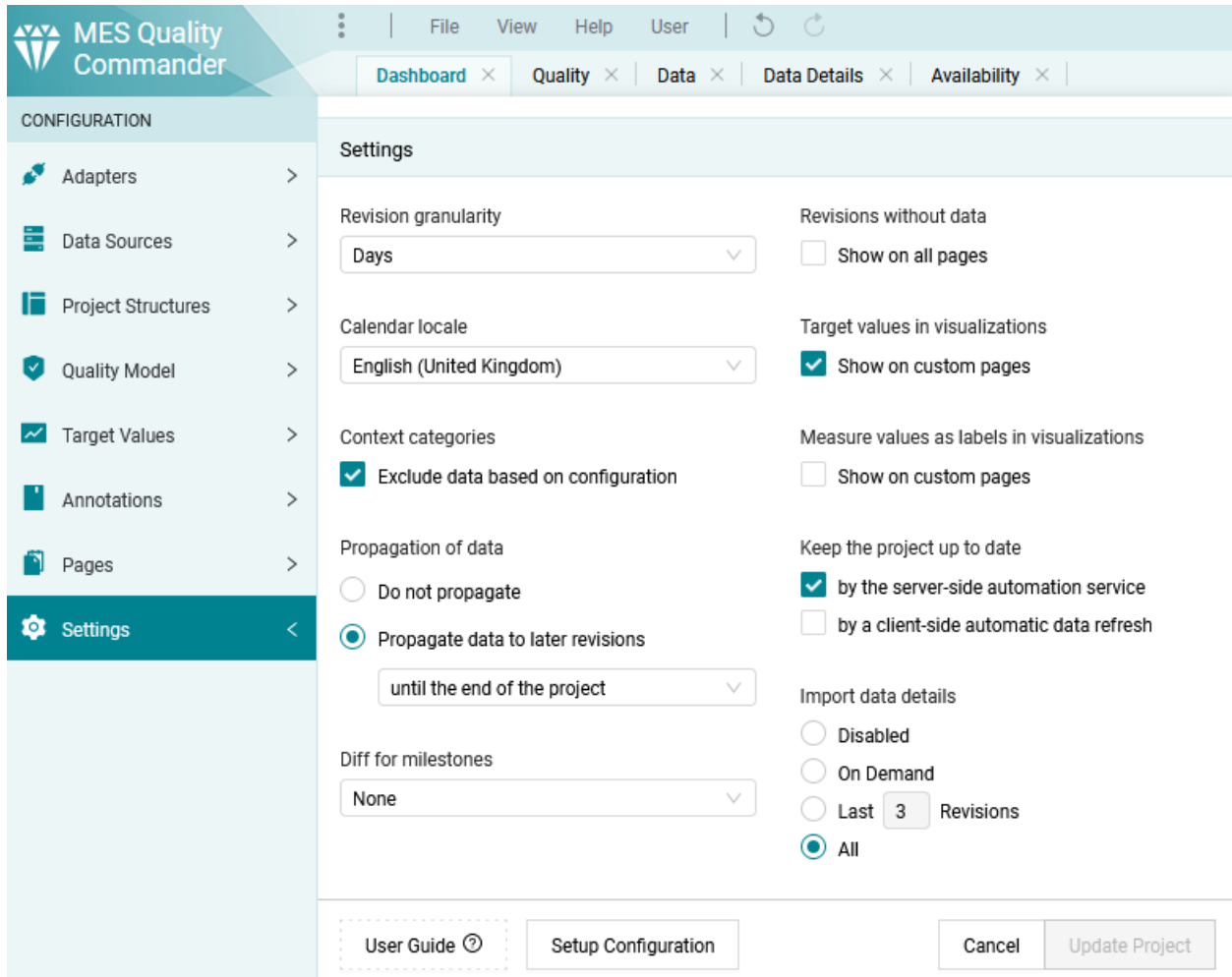
Figure 4.5: How to export Setup Configuration Excel file for current project

- names and paths of MQC configuration files, e.g. project structure

- as well as almost all options, e.g. revision granularity.

All values described in the following may be left empty. MQC applies defaults for all values, which are not configured in the setup configuration.

Errors in the configuration will be ignored. In any case a project is created and the user gets a notification via a validation dialog after the load of the setup configuration has been finished.

As shown in Figure 4.4, each configuration entry consists of a `Type`, which is fix, and a `Value`, which can be configured.

This configuration can be divided into three parts:

a. Set the paths of all files that can be imported in MQC. This contains:

- **BasePath** (optional): The first row can be optionally used to set a base path. If all files (reports as well as configuration sources) are located under the same path, this can be defined once as **BasePath** and all other path values may be configured as relative paths.

- **Data**: Data sources containing reports of tools. For each data source one row is added to the setup configuration.

    - Files and directories: The path of them has to be used as value.

    - Git Repository(ies): Provide the URL for the git repository to be used as a data source. Multiple repositories can be added as one data source (one row), if the same commit filter and file pattern configuration applies. The respective filter configurations are added as additional lines in the same cell (see *Git*).

- **ProjectStructure**: Path of configuration source file (see *Project Structure*).

- **QualityModel**: Path of configuration source file (see *Quality Model*).

- **TargetValues**: Path of configuration source file (see *Target Values*).

- **Annotations**: Path of configuration source file (see *Annotations*).

- **Dashboards**: Path of configuration source file (see *Dashboards*).

- **CustomPages**: Path of configuration source file (see *Custom Pages*).

- **CustomAdapter**: For each custom adapter, a separate row should be added and the path of the source file has to be given as value (for more detailed information please see *Developing a Custom Adapter*).

In case of multiple configuration source files of the same type, e.g. if multiple Quality Models have to be imported, for each configuration source file a separate row has to be used.

b. Set default values for MQC options. This contains:

- **Adapters**: List of base adapters which should be enabled in MQC, all entries are separated by **comma (,)**. If empty or not defined, all base adapters are enabled. For more detailed information please see *Adapters*.

- **ArtifactMappingPattern**: There should be one Excel row for each pattern (see *Artifact Mapping Patterns*). The value cell contains two lines: the first line is the *Search RegEx* and the second line is the corresponding *Replace*.

- **AutomaticDataRefreshServer**: Can be true or false (see *Keep the project up to date*).

- **ContextCategoriesEnabled**: Can be true or false (see *Context Categories*).

- **DiffRevision**: The comparison base to be used for Diff pages and visualizations. Use one of the following options:

    - "#PreviousRevision"

    - "#PreviousMilestone".

- **EmptyRevisionsVisible**: Can be true or false (see *Revisions without data*).

- **MeasureValueLabelsVisible**: Can be true or false (see *Measure values as labels in visualizations (Custom pages)*).

- **Propagation**: Can be "None", "Project" or "Milestone" (see *Propagation of data*).

- **QualityAssessmentScope**: The scope of quality assessment. Use one of the following options:

    - "Absolute"

    - "Available"

    - "Relative" (only if target values are defined).

- **QualityBin**: For each quality bin, a separate row should be added. Name, Hex Color Code and Upper Quality Boundary have to be added each on new line as value (see *Quality Bins*).

- **RevisionGranularity**: Can be "Months", "CalendarWeeks" or "Days" (see *Revision granularity*).

- **RevisionGranularityCulture**: The calendar week definition (see *Calendar locale*).

- **DataDetails**: Can be All, OnDemand or LastRevisions (see *Import data details*).

- **DataDetailsLastRevisionNum**: Number of Revisions if Data Details is LastRevisions (see *Import data details*).

- **TargetValuesVisible**: Can be true or false (see *Target values in visualizations (Custom pages)*).

c. Set the pages, which should be shown in the analysis. For each row from a group of provided pages in MQC (page type), the chosen pages should be added and separated by comma (,). Supported page types are:

- **PagesDashboard**

- **PagesInteractive**

- **PagesAction**

- **PagesLegacy**

- **PagesCustom**

For more detailed information please see *Pages*.

Importing a setup configuration

As shown in Figure 4.6, click on `Open local file` within the **Files and data** dialog on the left hand side to import a setup configuration file.
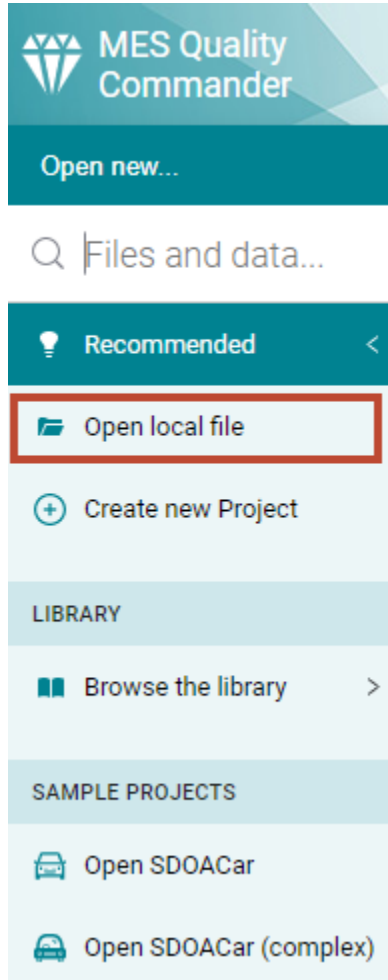
Figure 4.6: Import a Setup Configuration Excel file for easy setup of MQC

Then navigate to the folder where you have stored a previously created setup configuration file, select the file and confirm the dialog.

Alternatively, you may use the `Open Setup Configuration` button in the **Create Project** dialog to import a setup configuration file (see Figure 4.7).

In both cases, the setup configuration settings are used to pre-fill the **Create Project** dialog. This offers the possibility to check and potentially adapt the whole configuration before finally creating the project. Any misconfiguration is notified and is replaced by defaults.

If you are using the web version of MQC, before importing a setup configuration you need to ensure that the setup configuration file is located on a network drive, which is mounted at and can be accessed by the server. The **BasePath** value within the configuration file should be set to the network path of the directory where your project data is located.

Figure 4.7: Import a Setup Configuration via the Create Project dialog

4.2 MANAGING CONFIGURATIONS

Configuration data is loaded into MQC by loading configuration source files. There are different types of configuration source files to handle different groups of configuration data:

- **Project Structure configuration source file**

  Contains configuration data like expected artifacts, project milestones, and one or multiple artifact structures (see *Project Structure*).

- **Quality Model configuration source file**

  Contains configuration data like expected measures, quality properties including the corresponding measurement functions and one or multiple quality model structures (see *Quality Model*).

- **Targets configuration source file**

  Contains target values for specific data measures, or quality properties per artifact and milestone (see *Target Values*).

- **Annotations configuration source file**

  Contains annotations configuration (see *Annotations*).

- **Pages Layouts configuration source file**

  Contains the layout configuration of pages like the Quality page (see *Pages Layouts*).

- **Dashboard Pages configuration source file**

  Contains the configuration of dashboard pages (see *Dashboards*).

- **Custom Pages configuration source file**

    Contains the configuration of user specific pages (see *Custom Pages*).

The format of configuration source files can be YAML, JSON or Excel.

4.2.1 Load / Reload

All configuration source files can be loaded via the left-hand side configuration menu (see Figure 4.8). To open the menu panel, click on the icon shown in the top-right.
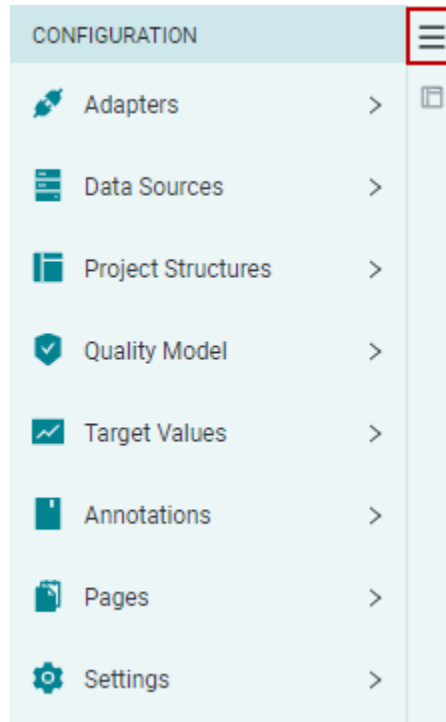
Figure 4.8: Open the configuration menu by clicking on the button in the left-hand side panel.

Then select the corresponding configuration group to add a configuration source file.

Configuration source files are loaded by using the `Add` button.

The "Add Source" Dialog allows the loading of a configuration source file from your local file system, if the MQC desktop client is used. Loading form a network path, git or selecting from the sample projects is available both on the MQC desktop client and the web player.

For some configuration source types there are additional sources available, like the suggestions for quality models or the special adapters for custom adapters.

When using Git as the source, the repository url and the branch have to be inputed.

Git repositories that have already been used for other configuration sources or for data sources are shown in a dropdown to be selected with one click.
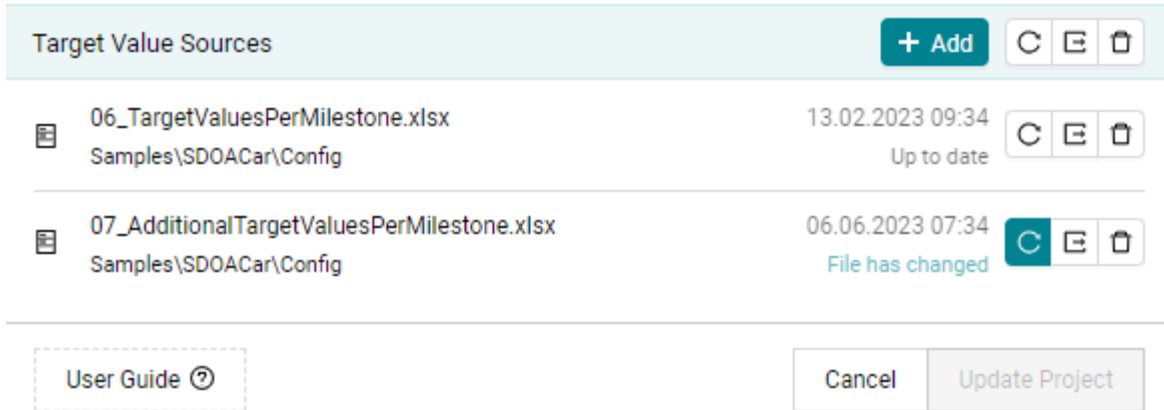
Figure 4.9: Dialog to manage one or multiple configuration source files, for example target value sources.

After cloning the git repository, by clicking on the clone button, the file tree of the git repository is shown in the state of the selected branch. A configuration source file can be choosen in the file browser.

More information about git sources can be found in the *Git* chapter of Data Sources.

Each configuration source file may be independently:

- **Reloaded**

  If after the load the file was changed at its physical location, MQC notifies about that as shown in Figure 4.9.

- **Exported**

  For more details on exporting configuration source files see *Export / Save*.

- **Deleted**

Besides, it is possible to do this for all loaded configuration source files at once by using the buttons next to the `Add` button.

Validation

When adding a new configuration source file, the content is validated for consistency and, additionally, it is checked for conflicts against the already loaded source files. Potentially, a notification popup informs about the reason(s), why an load is not possible.

4.2.2 Export / Save

Individual configuration source files are exported by using the `Export` button next to the configuration source file (see Figure 4.9). This will export the file in the format of the previous load.

To merge configurations from multiple loaded sources into a single source file, the `Export multiple` button next to `Add` button at the top of the dialog has to be used. This opens an additional **Export** dialog
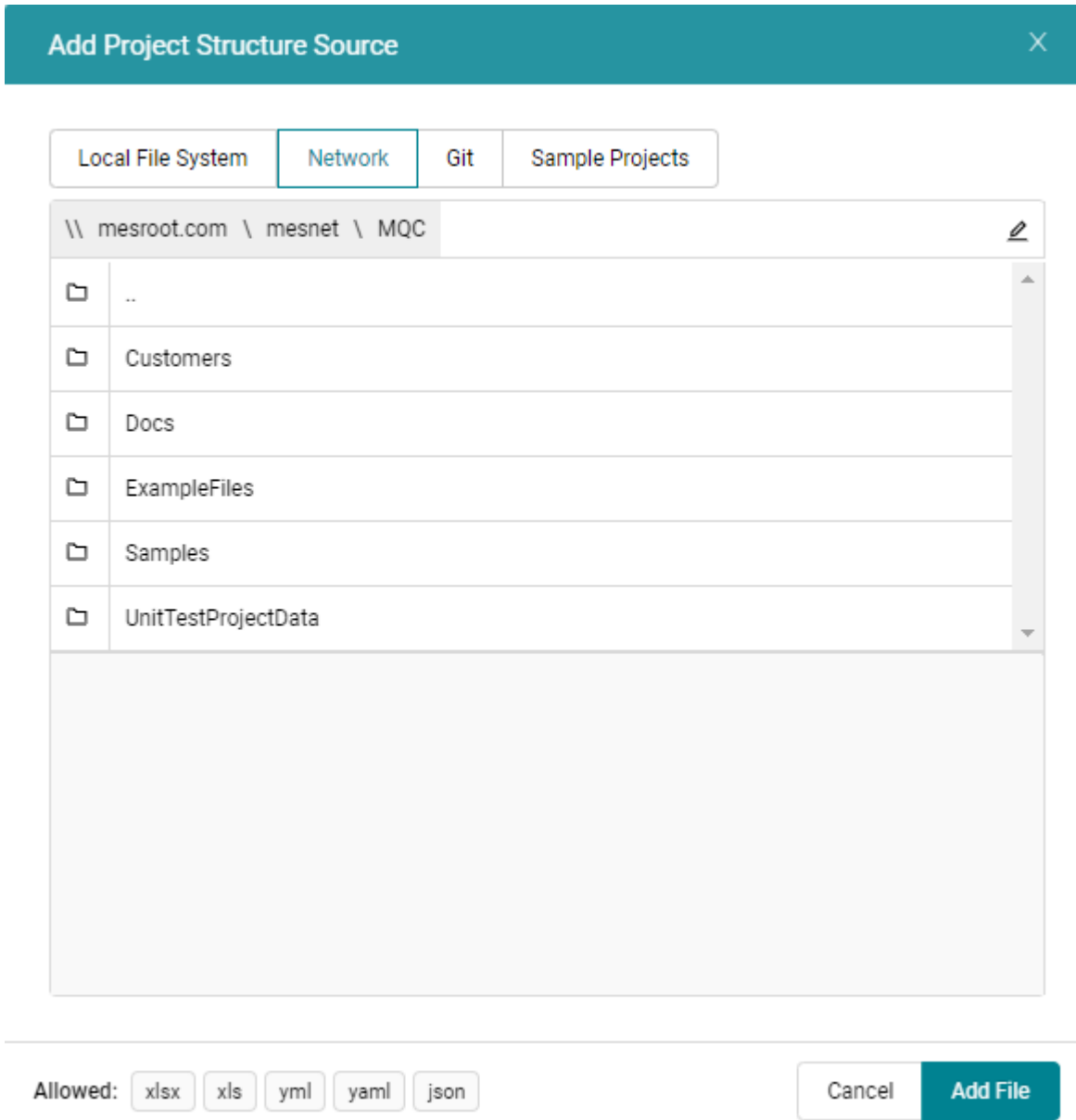
Figure 4.10: Dialog to add a configuration source file, for example a project structure source.
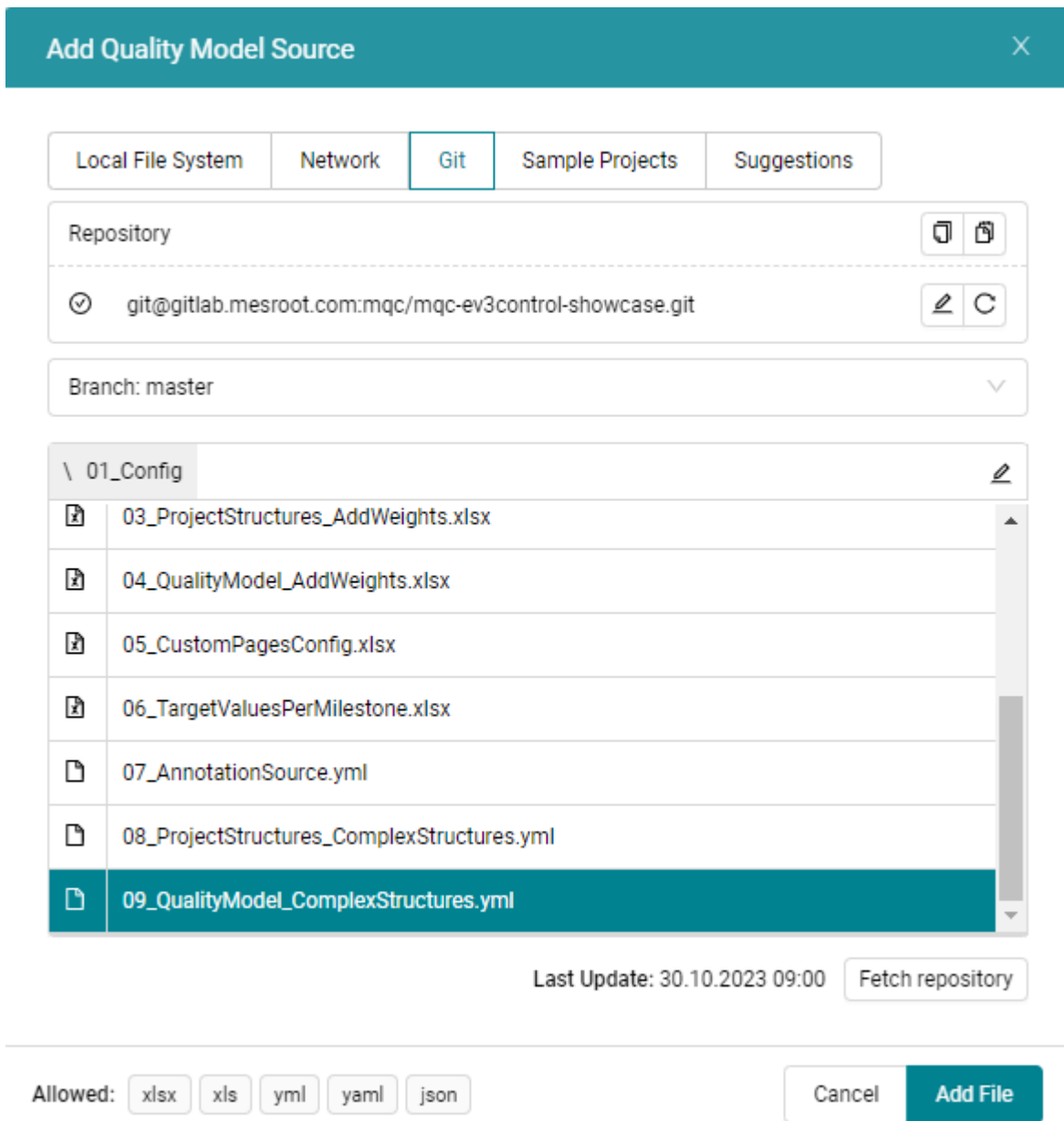
Figure 4.11: Dialog to add a configuration source file from git, for example a quality model source.

as shown in Figure 4.12. It is possible to select the relevant source files to be combined.

Additionally, this dialog provides the option to explicitly choose the file type of the combined export (YAML, JSON or Excel).
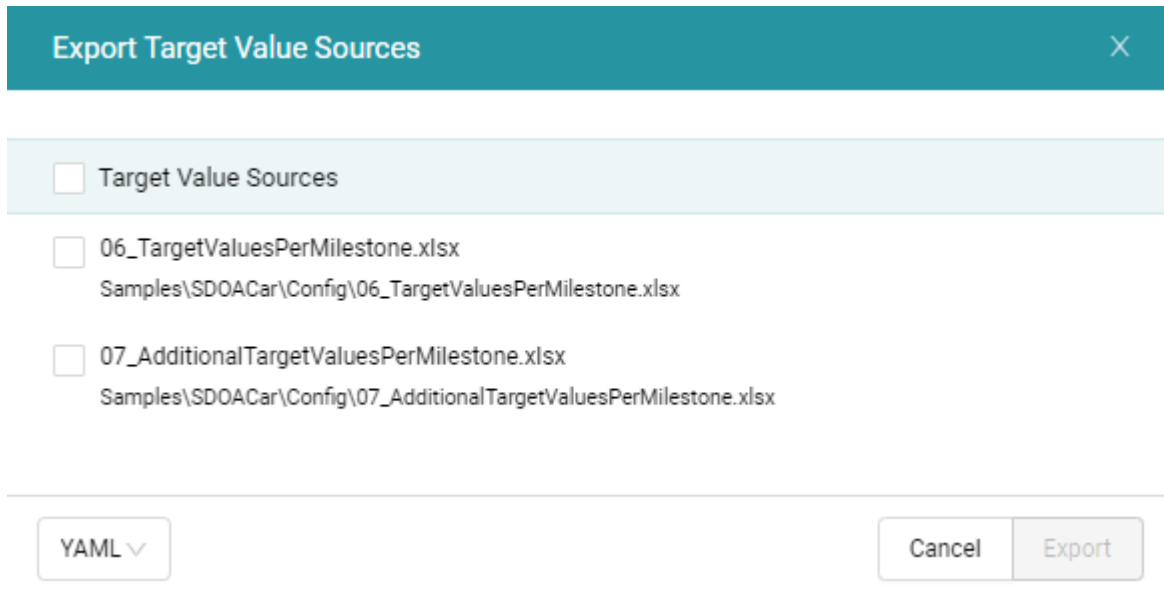
Figure 4.12: Export dialog for exporting multiple configuration sources into one file with the option to define the file format.

For configuration data that can be changed directly in MQC, a `Save/Save as` functionality is available (see Figure 4.13).
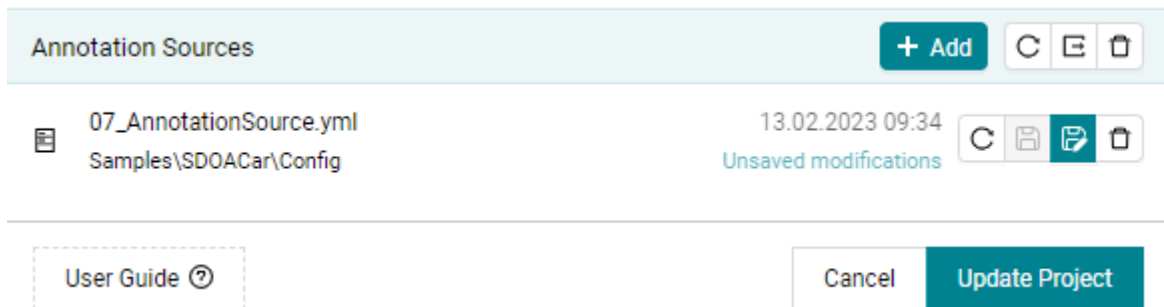
Figure 4.13: Configuration changes inside the tool can be saved to configuration source files.

This applies to:

- annotation sources

- layout sources

- dashboard sources

Any modification done by adding, deleting or refreshing configuration source files, finally, has to be applied by using the `Update Project` button.

4.3 CONFIGURATION SOURCES

4.3.1 Project Structure

MQC provides a default project structure if no Project Structure Source has been imported. The default project structure is based on the imported data:

- imported artifacts are shown with their names as read from the data reports, except a proper artifact mapping pattern was added (see *Artifact Mapping Patterns*)

- a default milestone period is used, which is starting from the date of the first imported data report and which goes until the date of the last imported data report

Project Name

MQC allows to define a dedicated project name.

Listing 4.1: Project name definition in YAML

```
Name: ProjectStructure
Project: SDOACar
```

In Excel, the project name configuration is done in a separate sheet called *ProjectArtifactStructure*.

Figure 4.14: Project name definition in Excel

Milestones

Milestones in MQC are treated as a time period. A milestone configuration consists of:

- **MilestoneName**

  The name of the milestone.

- **MilestoneStartDate**

  The start date of the milestone period. Either, the start date of the project for the first milestone, or one day after the end date of the previous milestone.

- **MilestoneDueDate**

  The end date of the milestone period respectively the date of the milestone itself.

- **MilestoneDuration**

  The milestone period duration.

Listing 4.2: Definition of milestones in YAML

```
Milestones:
- Name: Model Unit Start
  StartDate: 2021-04-27
  DueDate: 2021-06-28
  Duration: 63
- Name: Model Unit Extension
  StartDate: 2021-06-29
  DueDate: 2021-08-16
  Duration: 49
- Name: Model Integration
  StartDate: 2021-08-17
  DueDate: 2021-09-27
  Duration: 42
- Name: Test Improvement
  StartDate: 2021-09-28
  DueDate: 2021-10-18
  Duration: 21
```

In Excel, the milestone configuration is done in a separate sheet called *ProjectMilestoneStructure*.

ProjectName	MilestoneName	MilestoneStartDate	MilestoneDueDate	MilestoneDuration
SDOACar	Model Unit Start	27.04.2021	28.06.2021	63
SDOACar	Model Unit Extension	29.06.2021	16.08.2021	49
SDOACar	Model Integration	17.08.2021	27.09.2021	42
SDOACar	Test Improvement	28.09.2021	18.10.2021	21

Figure 4.15: Definition of milestones in Excel

It is not necessary to specify each of the parameters `MilestoneStartDate`, `MilestoneDueDate`, and `MilestoneDuration`.

Instead, it is sufficient to specify a project start date as start date of the first milestone phase in combination with a duration for all milestones. It is also possible to define start and end date per milestone without duration.

Milestone Structures

Milestones can be defined in sets, which can be organized in a tree structure, so you can easily navigate a lot of milestone sets and quickly enable or disable them. If a project has no need for a milestones structure, you can configure them as explained in *Milestones*.

Each set of milestones is a consecutive list of dates. Milestones from different sets can overlap or even have the same date.

When using YAML (or JSON) as configuration format:

The milestone structure consists of one or more milestone sets that contain:

- **Name**

  The name of the milestone set.

- **Description**

  An optional description of the milestone set.

- **Structures**

  Multiple milestone sets can be defined in the level below this set.

- **Milestones**

  Multiple milestones can be defined for this milestone set. The definition is identical to the unstructured *Milestones*.

There has to at least one Structure and/or Milestone defined in the milestone set.

Listing 4.3: Defining milestone sets and milestones in YAML

```
MilestoneStructures:
  - Name: 'Process'
    Milestones:
      - Name: Model Unit Start
        StartDate: 2021-04-26
        DueDate: 2021-06-27
      - Name: Model Unit Extension
        DueDate: 2021-08-15
      - Name: Model Integration
        DueDate: 2021-09-26
      - Name: Test Improvement
        DueDate: 2021-10-17
  - Name: 'ECUs'
    Structures:
      - Name: 'EV3C'
        Milestones:
          - Name: Model Implementation
            StartDate: 2021-08-15
            DueDate: 2021-08-25
          - Name: Test Implementation
            DueDate: 2021-09-05
          - Name: Test Refinement
            DueDate: 2021-09-14
          - Name: Test Completion
            DueDate: 2021-10-06
      - Name: 'MVS'
        Milestones:
          - Name: Model Implementation
            StartDate: 2021-07-01
            DueDate: 2021-07-08
          - Name: Test Implementation
```

```
                    DueDate: 2021-07-15
                  - Name: Test Refinement
                    DueDate: 2021-08-05
                  - Name: Test Completion
                    DueDate: 2021-10-06
          - Name: 'OD'
            Milestones:
                  - Name: Model Implementation
                    StartDate: 2021-04-26
                    DueDate: 2021-05-12
                  - Name: Test Implementation
                    DueDate: 2021-05-21
                  - Name: Test Refinement
                    DueDate: 2021-05-28
                  - Name: Test Completion
                    DueDate: 2021-10-06
          - Name: 'GP'
            Milestones:
                  - Name: Model Implementation
                    StartDate: 2021-06-01
                    DueDate: 2021-06-12
                  - Name: Test Implementation
                    DueDate: 2021-06-20
                  - Name: Test Refinement
                    DueDate: 2021-06-24
                  - Name: Test Completion
                    DueDate: 2021-10-06
```

In Excel a milestone structure can be configured by adding value to the `MilestonePath` Column in the *ProjectMilestoneStructure* sheet. The `MilestonePath` consists of the name-hierarchy of milestone sets concatenated by a dot. (e.g. "ECUs.GP") When using Excel as the configuration format, you cannot add a description for the milestone sets.

Artifacts

With the artifact configuration in the project structure the list of expected project artifacts is defined. A proper artifact configuration consists of:

- **ArtifactName**

  Mandatory, the name of the artifact.

- **ArtifactWeight**

  Optional, a weight to define that artifacts may have more or less influence than others. If nothing is set, a weight of 1 is used.

- **ContextCategories** (optional)

ProjectName	MilestoneName	MilestoneStartDate	MilestoneDueDate	MilestoneDuration	MilestonePath	MilestoneDescription
SDOACar	Model Implementation	15.08.2021	25.08.2021		ECUs.EV3C	
SDOACar	Test Implementation	26.08.2021	05.09.2021	11	ECUs.EV3C	
SDOACar	Test Refinement	06.09.2021	14.09.2021	9	ECUs.EV3C	
SDOACar	Test Completion	15.09.2021	06.10.2021	22	ECUs.EV3C	
SDOACar	Model Implementation	01.07.2021	08.07.2021		ECUs.MVS	
SDOACar	Test Implementation	09.07.2021	15.07.2021	7	ECUs.MVS	
SDOACar	Test Refinement	16.07.2021	05.08.2021	21	ECUs.MVS	
SDOACar	Test Completion	06.08.2021	06.10.2021	62	ECUs.MVS	
SDOACar	Model Implementation	26.04.2021	12.05.2021		ECUs.OD	
SDOACar	Test Implementation	13.05.2021	21.05.2021	9	ECUs.OD	
SDOACar	Test Refinement	22.05.2021	28.05.2021	7	ECUs.OD	
SDOACar	Test Completion	29.05.2021	06.10.2021	131	ECUs.OD	
SDOACar	Model Implementation	01.06.2021	12.06.2021		ECUs.GP	
SDOACar	Test Implementation	13.06.2021	20.06.2021	8	ECUs.GP	
SDOACar	Test Refinement	21.06.2021	24.06.2021	4	ECUs.GP	
SDOACar	Test Completion	25.06.2021	06.10.2021	104	ECUs.GP	
SDOACar	Model Unit Start	26.04.2021	27.06.2021		Process	
SDOACar	Model Unit Extension	28.06.2021	15.08.2021	49	Process	
SDOACar	Model Integration	16.08.2021	26.09.2021	42	Process	
SDOACar	Test Improvement	27.09.2021	17.10.2021	21	Process	

Figure 4.16: Defining milestone paths for milestones in excel

Context categories (see *Context Categories*) can be assigned to one or multiple artifacts in the project structure configuration source.

Listing 4.4: Assign context categories to artifacts in YAML

```
Artifacts:
  - Name: EV3Control_main
  - Name: ObstacleDetection
    ContextCategories:
      - MLC_CC
  - Name: GlobalPosition
    ContextCategories:
      - Global_CC
  - Name: ManageVehicleStates
    ContextCategories:
      - MLC_CC
```

In Excel, the assignment of context categories to artifacts is done within the *ArtifactStructure* sheet.

When no context category is assigned, i.e. the field is left empty, all available data is shown for that artifact.

Details about how to switch on or off categories in MQC can be found in *Context Categories*.

- **ArtifactPath(s)**

Optional, a list of artifact names extracted from data sources to be treated as the same artifact (see *Artifacts and Artifact Structure*).

Independent from a specific artifact mapping, MQC allows to define general artifact mapping patterns (see *Artifact Mapping Patterns*).

- **ArtifactStructure**

For details see *Artifact Structures*.

ArtifactName	ArtifactWeight	ContextCategories	StructureSoftware	StructureTeams	StructureComponents
EV3Control_main	1		SDOAC SW	Team F	Integration
ObstacleDetection	1	MLC_CC	SDOAC SW	Team F	Detection
GlobalPosition	1	Global_CC	SDOAC SW	Team P	Control
ManageVehicleStates	1	MLC_CC	SDOAC SW	Team P	Control

Figure 4.17: Assign context categories to artifacts in Excel

Listing 4.5: Defining expected artifacts in YAML

```
Artifacts:
- Name: EV3Control_main
  Weight: 1
  Paths:
  - EV3Control_main
  - EV3Control_demo_ec
  - EV3Control_demo_ec/EV3Control
  Structures:
  - Path: Software.Teams.Components
    Value: SDOAC SW.Team F.Integration
- Name: ObstacleDetection
  Weight: 1
  ContextCategories:
  - MLC_CC
  Paths:
  - ObstacleDetection
  - EV3Control_demo_ec/VehicleManager/ObstacleDetection
  - ObstacleDetection_demo_ec
  - Obstacledetection_demo_ec
  - ObstacleDetection_demo_ec/ObstacleDetection
  Structures:
  - Path: Software.Teams.Components
    Value: SDOAC SW.Team F.Detection
```

In Excel, the list of expected artifacts is configured within the *ArtifactStructure* sheet.

ArtifactName	ArtifactWeight	ContextCategories	StructureSoftware	StructureTeams	StructureComponents
EV3Control_main	1		SDOAC SW	Team F	Integration
ObstacleDetection	1	MLC_CC	SDOAC SW	Team F	Detection
GlobalPosition	1	Global_CC	SDOAC SW	Team P	Control
ManageVehicleStates	1	MLC_CC	SDOAC SW	Team P	Control

Figure 4.18: Defining expected artifacts in Excel using

Artifact Structures

Artifact structures can be used in MQC to group artifacts, which can be assigned to a specific path of such a defined structure.

When using YAML (or JSON) as configuration format:

- Multiple hierarchies can defined.

- Each artifact can be assigned to one or multiple hierarchies.

- The hierarchies can have a different amount of levels.

- A description can be added also to structure levels.

Listing 4.6: Defining artifact structures and artifacts with structure assignments in YAML

```
ArtifactStructures:
- Name: Architecture
  Path: Software.Components
  Values:
    - Name: SDOAC SW
      Description: SDOA Car Software
      Values:
        - Name: Integration
          Description: Integration Model
        - Name: Control
          Description: Control Model
- Name: Responsibility
  Path: Devisions.Teams
Artifacts:
- Name: EV3Control
  Structures:
```

(continues on next page)

```
    - Path: Software.Components
      Value: SDOAC SW.Integration
    - Path: Devisions.Teams
      Value: Development.Team F
- Name: GlobalPosition
  Description: Subsystem for the position of the vehicle
  Structures:
    - Path: Software.Components
      Value: SDOAC SW.Control
    - Path: Devisions.Teams
      Value: Development.Team P
```

In Excel, the artifact structure configuration is done within the *ArtifactStructure* sheet (as part of the artifact configuration, see *Artifacts*).

When using Excel as configuration format:

- Only one hierarchy can be defined.

- Multiple structure levels are allowed.

- Structure levels have to be defined as columns with a column name beginning with "Structure" (e.g. 'StructureSoftware' for a structure level called "Software").

- The order of the levels is defined by the order of the columns in the Excel (e.g. Software > Teams > Components).

ArtifactName	ArtifactWeight	ContextCategories	StructureSoftware	StructureTeams	StructureComponents
EV3Control_main	1		SDOAC SW	Team F	Integration
ObstacleDetection	1	MLC_CC	SDOAC SW	Team F	Detection
GlobalPosition	1	Global_CC	SDOAC SW	Team P	Control
ManageVehicleStates	1	MLC_CC	SDOAC SW	Team P	Control

Figure 4.19: Defining artifact structures and artifacts with structure assignments in Excel

4.3.2 Quality Model

MQC provides data source specific quality model configurations. These predefined source files can be directly chosen via the "Suggestions" tab within the **Add** dialog (see *Load / Reload*).

If no quality model was imported, MQC recognizes the used data sources (either based on the enabled adapters or from the data currently imported) and loads the corresponding quality model suggestions.

These suggestions contain Base Measures, Derived Measures and Quality Properties with a Quality Structure. For Data Details the suggestions also contain Finding Results and Finding Structures for DataSource Adapters that support Data Details.

Data

Base Measures

The base measures configured in the quality model define the set of expected data. When configuring base measures, MQC expects the following information:

- **DataSourceName**

  The tool/source providing a data report containing data to be imported by MQC. Examples are MXAM and TPT.

- **MeasurementName**

  A default measurement name, which is used if the measurement can't be extracted from a report.

  Typically, a measurement means the set of operations done to determine the value for a measure. For example this can be the name of a specific guideline document, which is used for the static analysis of a model, or the kind of test environment like MiL, SiL or PiL.

- **BaseMeasureName**

  The name for a group of measures, e.g. "FindingCount" for the data source MXAM.

- **VariableName**

  The name of a specific measure inside a measure group, e.g. "Passed" or "Failed". Full measure names would then be "FindingCount.Passed" or "FindingCount.Failed".

- **DefaultValue**

  The value, which should be set if no value for a specific measure can be extracted from a report, typically 0. (see default-values-configuration)

- **Aggregation** (optional)

  Combining multiple artifact paths into an artifact can result in more than one data point for the same report at a specific revision and measure. You can specify a aggregation method, sum or avg, to aggregate instead of marking one data point as duplicate and ignoring it.

  Default: None

- **Increment** (optional)

  Normally, the data imported for a revision contains absolute numbers. If the data of each revision consists only of the changes of the data (increase / decrease) then the *Increment* can be set to true. Enabling this setting results in a measure value where the imported value is added to the measure value of the previous revision.

Default: false

Default Values

MQC uses the configured defaults, if the report contains values for at least one other measure belonging to the same base measure group. Otherwise, the whole base measure group is treated as "Missing".

Listing 4.7: Defining expected base measures with there default values in YAML

```
BaseMeasures:
- Name: MXAM.GuidelineAnalysis.FindingCount.Aborted
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Canceled
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Failed
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Info
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Passed
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Repaired
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Review
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Unrepaired
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Warning
  DefaultValue: 0
- Name: MXAM.GuidelineAnalysis.FindingCount.Ignored
  DefaultValue: 0
```

In Excel, the base measure configuration is done in a separate sheet called *BaseMeasure*.

DataSourceName	MeasurementName	BaseMeasureName	VariableName	DefaultValue
MXAM	GuidelineAnalysis	FindingCount	Aborted	0
MXAM	GuidelineAnalysis	FindingCount	Canceled	0
MXAM	GuidelineAnalysis	FindingCount	Failed	0
MXAM	GuidelineAnalysis	FindingCount	Info	0
MXAM	GuidelineAnalysis	FindingCount	Passed	0
MXAM	GuidelineAnalysis	FindingCount	Repaired	0
MXAM	GuidelineAnalysis	FindingCount	Review	0
MXAM	GuidelineAnalysis	FindingCount	Unrepaired	0
MXAM	GuidelineAnalysis	FindingCount	Warning	0
MXAM	GuidelineAnalysis	FindingCount	Ignored	0

Figure 4.20: Defining expected base measures with there default values in Excel

Derived Measures

A fully qualified derived measure consists of:

- **DataSourceName**

  Optional, typically the name of the data source of the base measures used to calculate the derived measure.

- **MeasurementName**

  Optional, typically the name of the measurement of the base measures used to calculate the derived measure.

- **DerivedMeasureName**

  Mandatory, the name for a group of derived measures.

- **VariableName**

  Mandatory, the name of a specific derived measure inside a derived measure group.

- **MeasureFunction**

  Mandatory, the expression to calculate a derived measure using base measures and/or other derived measures.

  Base and derived measures must be set in square brackets `[` and `]`!

If no data source and measurement names are configured (short notation), MQC takes data source and measurement from the base and/or derived measures used to calculate the current derived measure.

This means, if the derived measure configuration is in short notation and the base measures used to calculate the derived measure are provided for multiple measurements (e.g. TestCount.Total for MiL and for SiL), the derived measure will be calculated for all the measurements as well.

To calculate a derived measure for a specific measurement only, fully qualified measure names have to be used inside the measure function.

Listing 4.8: Defining derived measures using a short notation in YAML

```
DerivedMeasures:
- Name: GuidelinesCalc.Passed
  Expression: '[GuidelineCount.Passed]+log(1+[FindingCount.Passed],2)'
- Name: GuidelinesCalc.Repaired
  Expression: '[GuidelineCount.Repaired]+log(1+[FindingCount.Repaired],2)'
- Name: GuidelinesCalc.Unrepaired
  Expression: '[GuidelineCount.Unrepaired]+log(1+[FindingCount.Unrepaired],2)'
- Name: GuidelinesCalc.Failed
  Expression: '[GuidelineCount.Failed]+log(1+[FindingCount.Failed],2)'
- Name: GuidelinesCalc.Info
  Expression: '[GuidelineCount.Passed with Infos]+log(1+[FindingCount.Info],2)'
- Name: GuidelinesCalc.Review
  Expression: '[GuidelineCount.Review]+log(1+[FindingCount.Review],2)'
```

```
- Name: GuidelinesCalc.Warning
  Expression: '[GuidelineCount.Warning]+log(1+[FindingCount.Warning],2)'
- Name: GuidelinesCalc.AbortedCanceled
  Expression: '[GuidelineCount.Aborted]+[GuidelineCount.Canceled]'
```

In Excel, the derived measure configuration is done in a separate sheet called *DerivedMeasure*.

DataSourceName	MeasurementName	DerivedMeasureName	VariableName	MeasureFunction
		GuidelinesCalc	Passed	[GuidelineCount.Passed]+log(2,1+[FindingCount.Passed])
		GuidelinesCalc	Repaired	[GuidelineCount.Repaired]+log(2,1+[FindingCount.Repaired])
		GuidelinesCalc	Unrepaired	[GuidelineCount.Unrepaired]+log(2,1+[FindingCount.Unrepaired])
		GuidelinesCalc	Failed	[GuidelineCount.Failed]+log(2,1+[FindingCount.Failed])
		GuidelinesCalc	Info	[GuidelineCount.Passed with Infos]+log(2,1+[FindingCount.Info])
		GuidelinesCalc	Review	[GuidelineCount.Review]+log(2,1+[FindingCount.Review])
		GuidelinesCalc	Warning	[GuidelineCount.Warning]+log(2,1+[FindingCount.Warning])
		GuidelinesCalc	AbortedCanceled	[GuidelineCount.Aborted]+[GuidelineCount.Canceled]

Figure 4.21: Defining derived measures using a short notation in Excel

Context Categories

A context category definition may contain:

- **name of the context category**

  mandatory, later used to be assigned to artifacts

- **expected data**

  optional, white list

- **excluded data**

  optional, black list

Both, expected and excluded data has to be defined using comma-separated lists of data sources, measurements and/or measures. Use either the full name or name patterns (including wildcards).

Listing 4.9: Configuring context categories in YAML

```
ContextCategories:
- Name: StaticAnalysisOnly
  Expect: MXAM, MXRAY
- Name: TestNoCoverage
  Expect: MXAM, MXRAY, MTest
  Exclude: MTest.*.Model*Coverage
```

In Excel, the context category configuration is done in a separate sheet called *ContextCategory*.

An empty "Expect" field means, all data is expected for that context category. To state that no data is expected at all, use the key word "none".

An empty "Exclude" field means, nothing is excluded, but still, the amount of data may be reduced by specifying expected data.

ContextCategoryName	Expect	Exclude
StaticAnalysisOnly	MXAM, MXRAY	
TestNoCoverage	MXAM, MXRAY, MTest	MTest.*.Model*Coverage

Figure 4.22: Configuring context categories in Excel

Quality

Quality Properties

When configuring quality properties, MQC expects the following information:

- **QualityPropertyName**

  Mandatory, the name of the quality property.

- **Weight**

  Optional, a weight to define that quality properties may have more or less influence than others. If nothing is set, a weight of 1 is used.

- **MeasurementFunction**

  Mandatory, the expression to calculate a quality property measure value using base measures and/or other derived measures.

  Base and derived measures must be set in square brackets [and]!

- **Description**

  Optional, an explanation of the quality property may be added to document and visualize the background concept.

If the measurement function is defined in short notation and the base and derived measures used to calculate the quality property are provided for multiple measurements (e.g. TestCount.Total for MiL and for SiL), the quality property will be calculated for all the measurements as well.

To calculate a quality property for a specific measurement only, fully qualified measure names have to be used inside the measurement function.

Listing 4.10: Quality property definition in YAML

```
QualityProperties:
- Name: Testable Requirements with Test Sequences
  Weight: 2
  Expression: [Testable Requirements with Test Sequences.Reached] /
              [Testable Requirements with Test Sequences.Total]
  Description: >
      Show the percentage of the reached testable requirements
      with test Sequences
```

In Excel, the quality property configuration is done within the *QualityModel* sheet (together with the quality structure configuration, see *Quality Structures*).

QualityPropertyName	QualityPropertyWeight	QualityMeasurementFunction	QualityPropertyDescription
Testable Requirements with Test Sequences	2	[Testable Requirements with Test Sequences.Reached] / [Testable Requirements with Test Sequences.Total]	Show the percentage of the reached testable requirements with test Sequences

Figure 4.23: Quality property definition in Excel

Bin Conditions

When configuring bin conditions, attention must be paid to the following:

- **MeasurementFunction**

  Instead of a numerical function using base measures and/or derived measures, the term `[MQC.Bin]` must be used to indicate that the quality property is calculated via a bin condition.

- **BinConditions**

  For each quality bin a conditional expression must be added. The particular quality bin is assigned, if the expression returns `true`.

  Bin conditions must be defined in the right order starting with the lowest quality bin up to the highest. This is important for the evaluation of the conditions.

  Please, ensure that all used quality bins are properly defined (see *Quality Bins*).

Listing 4.11: Using Bin Conditions to define quality properties in YAML

```
QualityProperties:
- Name: B2B
  Weight: 1
  Expression: '[MQC.Bin]'
  BinConditions:
  - Name: Bad
    Expression: >
      [B2B Statement Coverage.Ratio] is not null and
      [B2B MC/DC Coverage.Ratio] is not null and
      [B2B Test Count.Failed] is not null and
      [B2B Test Count.Error] is not null
  - Name: Acceptable
    Expression: False
  - Name: Good
    Expression: >
      [B2B Statement Coverage.Ratio] = 1 and
      [B2B Test Count.Failed] = 0 and
      [B2B Test Count.Error] = 0
```

In Excel, the bin condition configuration is split into two parts. The quality property configuration still has to be done within the *QualityModel* sheet.

Besides, the bin condition configuration is done in a separate sheet called *Bin Condition* (see Figure 4.24).

QualityPropertyName ▼	QualityPropertyWeight ▼	QualityMeasurementFunction ▼
B2B	1	[MQC.Bin]

Figure 4.24: Quality property definition using bin conditions instead of a measurement function in Excel

Quality Structures

To group quality properties in order to define different quality aspects, quality structures can be defined.

Quality properties can be assigned to a specific path of such a defined structure.

When using YAML (or JSON) as configuration format:

- Multiple hierarchies can defined.

- Each quality property can be assigned to one or multiple hierarchies.

- The hierarchies can have a different amount of levels.

- A description can be added also to structure levels, not only to quality properties.

Listing 4.12: Definition of a quality model structure and assigning quality properties to this structure in YAML

```
QualityStructures:
  - Name: Functional Suitability
    Path: Models.Characteristics.Subcharacteristics
    Values:
      - Name: Product Quality Model
        Values:
          - Name: Correctness
            Description: >
                Grouping the quality properties.

                Quality properties in this group evaluate the successful
                operation of the models.
            Values:
              - Name: Model Design
              - Name: Functional Requirements
                Description: >
                    Grouping the quality properties that evaluate the services,
                    the models must offer
              - Name: Test Sequences
              - Name: Assessments
          - Name: Completeness
            Values:
              - Name: Functional Requirements
              - Name: Test Sequences
              - Name: Assessments
```

(continues on next page)

```
                  - Name: Model Coverage
                  - Name: Code Coverage
  - Name: Quality Assurance Methods
    Path: Quality Assurance.QA Method.QA Work Product
    Values:
      - Name: Product Quality Model
        Values:
          - Name: Static Analysis
            Values:
              - Name: Guidelines Model Design
              - Name: Guidelines Model Architecture
          - Name: Test
            Values:
              - Name: Requirements
              - Name: Test Sequences
              - Name: Assessments
              - Name: Structural Coverage
QualityProperties:
  - Name: Requirements Compliance
    Weight: 3
    Expression: '[Requirements Compliance.Reached] / [Requirements Compliance.Total]'
    Structures:
      - Path: Models.Characteristics.Subcharacteristics
        Value: Product Quality Model.Correctness.Functional Requirements
      - Path: Quality Assurance.QA Method.QA Work Product
        Value: Product Quality Model.Test.Requirements
```

In Excel, the quality structure configuration is done within the *QualityModel* sheet.

When using Excel as configuration format:

- Only one hierarchy can be defined.

- Multiple structure levels are allowed.

- Structure levels have to be defined as columns with a column name beginning with "Quality" (e.g. 'QualityCharacteristics' for a structure level called "Characteristics").

- The order of the levels is defined by the order of the columns in the Excel (e.g. Models > Characteristics > Subcharacteristics).

Data Details

To enable Data Details, expected finding results have to be configured and the data details setting has to be set to All, On Demand or Last Revisions. (see *Import data details*)

QualityModel	QualityCharacteristic	QualitySubCharacteristic	QualityPropertyName
Product Quality Model	Correctness	Model Design	Guideline Compliance
Product Quality Model	Correctness	Architectural Design	Local Complexity
Product Quality Model	Correctness	Architectural Design	Modelling Depth
Product Quality Model	Correctness	Architectural Design	ModelClones
Product Quality Model	Correctness	Architectural Design	Inports
Product Quality Model	Correctness	Architectural Design	Outports
Product Quality Model	Correctness	Architectural Design	%Elementary Inputs Unused (globally)
Product Quality Model	Correctness	Functional Requirements	Requirements Compliance
Product Quality Model	Completeness	Functional Requirements	Requirements with Reviewed Testability
Product Quality Model	Completeness	Functional Requirements	Testable Requirements with Assessments
Product Quality Model	Completeness	Functional Requirements	Testable Requirements with Test Sequences
Product Quality Model	Completeness	Test Sequences	Test Sequence Work Progress
Product Quality Model	Completeness	Test Sequences	Reviewed Test Sequences
Product Quality Model	Correctness	Test Sequences	Test Sequence Compliance
Product Quality Model	Completeness	Assessments	Assessment Work Progress
Product Quality Model	Correctness	Assessments	Assessments Compliance

Figure 4.25: Definition of a quality model structure and assigning quality properties to this structure in Excel

Finding Results

The finding results are the original results of the findings read by the DataSource Adapters.

This configuration allows to define which of these results are expected, how they should be colored in the visualizations and what the severity for the order of the aggregation should be.

MQC expects the following information for finding results:

- **Name**

  The name of the result, which is always {DataSource}.{Result}.

- **Color**

  The color for the visualizations, a 6 digit Hexcode with leading #.

- **Severity**

  The severity is a number (int) that defines the order of the bins in the aggregation. Only the color of the worst result (= highest severity) is visible in most visualizations.

  In the treemap the size of the elements is based on the number of findings multiplied by the severity.

Listing 4.13: Definition of finding results for datasource MXAM in YAML

```
FindingResults:
- Name: MXAM.Ignored
  Color: '#F2F2F2'
  Severity: 1
- Name: MXAM.Passed
  Color: '#77AF35'
```

(continues on next page)

```
        Severity: 1
    - Name: MXAM.Repaired
      Color: '#A5CF63'
        Severity: 1
    - Name: MXAM.Info
      Color: '#67A1DC'
        Severity: 2
    - Name: MXAM.Warning
      Color: '#E2DC49'
        Severity: 4
    - Name: MXAM.Unrepaired
      Color: '#DFAB3C'
        Severity: 5
    - Name: MXAM.Failed
      Color: '#C03F33'
        Severity: 7
    - Name: MXAM.Review
      Color: '#864DC2'
        Severity: 8
    - Name: MXAM.Aborted
      Color: '#7F7F7F'
        Severity: 9
    - Name: MXAM.Canceled
      Color: '#595959'
        Severity: 10
```

Name	Color	Severity
MXAM.Ignored	#F2F2F2	1
MXAM.Passed	#77AF35	1
MXAM.Repaired	#A5CF63	1
MXAM.Info	#67A1DC	2
MXAM.Warning	#E2DC49	4
MXAM.Unrepaired	#DFAB3C	5
MXAM.Failed	#C03F33	7
MXAM.Review	#864DC2	8
MXAM.Aborted	#7F7F7F	9
MXAM.Canceled	#595959	10

Figure 4.26: Definition of finding results in Excel

Finding Bin Structures

The purpose for Finding Bin Structures is twofold:

- **Unified results between different DataSources**

    By grouping similar results of different data sources into a bin the amount of results can be reduced and the results simplyfied.

Example: "MXAM.Passed", "TPT.Succeeded", "TPT.Covered", "TPT.Linked" and "EmbeddedTester.Justified" could be mapped to a bin named "Passed".

The Bin Structure can then be selected in the toolbar, so that the visualizations uses the bins instead of the original finding results.

- **Visibility filter to focus on specific result types**

    To only show a subset of findings (e.g. Issues), a visibility filter can be configured.

    After a visibility bin is configured, it can be selected in the toolbar to change the visibility of finding results in the KPI, Heatmap and Treemap visualizations. The List is filtered to show only the visible findings.

Definition

MQC expects the following information for finding bin structures:

- **Name**

    The name of the structure, selectable in the toolbar.

- **Bins**

    Multiple bins, each consisting of:

    – **Name**

    The name of the bin, typically a unified result name.

    – **Color**

    The color for the visualizations, a 6 digit Hexcode with leading #.

    – **Severity**

    The severity is a number (int) that defines the order of the bins in the aggregation. Only the color of the worst result (= highest severity) is visible in most visualization.

    In the treemap the size of the elements is based on the number of findings multiplied with the severity.

    – **Results**

    List of Finding Results. (see *Finding Results*)

    Each Result can only be assigned to one bin of a structure. By using a *****, the result assignment can be a wildcard that maches multiple results.

<p align="center">Listing 4.14: Definition of finding bin structures as mapping for selection in YAML</p>

```
FindingBinStructures:
- Name: Result
  Bins:
  - Name: Passed
    Color: '#77AF35'
    Severity: 1
    Results:
```

<div align="right">(continues on next page)</div>

```
        - '*.Passed'
        - '*.Info'
        - MXAM.Ignored
        - MXAM.Repaired
    - Name: Warning
      Color: '#E2DC49'
      Severity: 4
      Results:
        - '*.Warning'
    - Name: Failed
      Color: '#C03F33'
      Severity: 7
      Results:
        - '*.Failed'
        - MXAM.Unrepaired
    - Name: Review
      Color: '#864DC2'
      Severity: 8
      Results:
        - '*.Review'
    - Name: Error
      Color: '#B40013'
      Severity: 10
      Results:
        - '*.Error'
        - MXAM.Aborted
        - MXAM.Canceled
```

StructureName	Name	Color	Severity	Results	DisabledForBin	EnabledForVisibility
Result	Passed	#77AF35	1	*.Passed *.Info MXAM.Ignored MXAM.Repaired	FALSE	FALSE
Result	Warning	#E2DC49	4	*.Warning	FALSE	FALSE
Result	Failed	#C03F33	7	*.Failed MXAM.Unrepaired	FALSE	FALSE
Result	Review	#864DC2	8	*.Review	FALSE	FALSE
Result	Error	#B40013	10	*.Error MXAM.Aborted MXAM.Canceled	FALSE	FALSE
State	Issues	#C03F33	1	*.Warning *.Failed *.Review *.Error MXAM.Unrepaired MXAM.Aborted MXAM.Canceled	TRUE	TRUE

Figure 4.27: Definition of finding bin structures in Excel

Listing 4.15: Definition of finding bin structures as visibility in YAML

```
- Name: State
  Bins:
  - Name: Issues
    Color: '#C03F33'
    Severity: 1
    Results:
    - '*.Warning'
    - '*.Failed'
    - '*.Review'
    - '*.Error'
    - MXAM.Unrepaired
    - MXAM.Aborted
    - MXAM.Canceled
    EnabledForVisibility: true
  Disabled: true
```

If Quality Model Suggestions are used, the Default Data Details Bins are loaded automatically. This configuration can also be manually added as a Quality Model, which is recommended if no custom bins or custom mapping is needed.

Finding Structures

The Structure of a Finding can be very different from another Finding. Even in the same DataSource the Structure can have a different depth.

To normalize these structures, so that they can be shown in the Heatmap, the Finding Structures can be

configured. Unnecessary structure can be removed, additional levels added and names can be freely defined.

MQC expects the following information for finding structures:

- **DataSource**

  The DataSource of the Findings of this structure.

- **Name**

  The level name of the structure.

- **Index**

  The starting index (beginning with 0) in the full structure that is imported by the DataSource Adapter.

  The "Last" keyword is replaced by the last item of the full structure. Simple math expressions are possible.

  If the Index is calculated as less than 0, it is set to 0. If it is calculated greater than the index of the last element it is set to the highest possible index.

- **IndexTo** (optional)

  Not needed if equal to **Index**. Multiple parts of the full structure will be combined with a slash (e.g. Chapter 1 / Chapter 2 / Chapter 3).

Example

For the Check "mcheck_misra_slsf_016_a" the following finding structure is read:

Table 4.1: Finding structure for finding of check "mcheck_misra_slsf_016_a"

Index	Element
0	mes_guidelines_embedded_coder_fs
1	Functional Model
2	Simulink
3	Established Design Principles
4	misra_slsf_016_a
5	mcheck_misra_slsf_016_a

Listing 4.16: Version 1: Definition of finding structures in YAML

```
FindingStructures:
- DataSource: MXAM
  Levels:
  - Name: Documents
    Index: 0
  - Name: Chapters
    Index: 1
    IndexTo: Last-2
  - Name: Guidelines
```

```
   Index: Last-1
 - Name: Checks
   Index: Last
```

Results in:

Table 4.2: Results of Version 1 of check "mcheck_misra_slsf_016_a"

Structure level	Value
Document	mes_guidelines_embedded_coder_fs
Chapter	Functional Model / Simulink / Established Design Principles
Guideline	misra_slsf_016_a
Check	mcheck_misra_slsf_016_a

Listing 4.17: Version 2: Definition of mxam finding structures with subchapters in YAML

```
FindingStructures:
- DataSource: MXAM
  Levels:
  - Name: Documents
    Index: 0
  - Name: Chapters
    Index: 1
  - Name: Subchapters
    Index: 2
    IndexTo: Last-2
  - Name: Guidelines
    Index: Last-1
  - Name: Checks
    Index: Last
```

Results in:

Table 4.3: Results of Version 2 of check "mcheck_misra_slsf_016_a"

Structure level	Value
Document	mes_guidelines_embedded_coder_fs
Chapter	Functional Model
Subchapter	Simulink / Established Design Principles
Guideline	misra_slsf_016_a
Check	mcheck_misra_slsf_016_a

The configuration is also possible in Excel, as shown below.

DataSourceName	Name	Index	IndexTo
MXAM	Documents	0	
MXAM	Chapters	1	Last-2
MXAM	Guidelines	Last-1	
MXAM	Checks	Last	
MTest	Measurements	0	
MTest	Measures	1	
MTest	Sub Measures	Last	

Figure 4.28: Definition of finding structures in Excel

Actions

For details about actions, see *Actions*.

Listing 4.18: Action definition in YAML

```
QualityProperties:
- Name: Testable Requirements with Test Sequences
  Weight: 2
  Expression: [Testable Requirements with Test Sequences.Reached] /
              [Testable Requirements with Test Sequences.Total]
  Actions:
  - Link test sequences to (covered) requirements
  - Derive further test sequences from uncovered requirements
```

In Excel, the action configuration is done in a separate sheet called *ActionList*.

QualityPropertyName	Action
Guideline Compliance	Refactor model according to guideline findings
Local Complexity	Refactor / split subsystems to smaller units
Modelling Depth	Refactor model according to overstructuring
ModelClones	Refactor cloned subsystems to library subsystems and link from library to clone groups
Inports	Refactor interfaces regarding number of inports
Outports	Refactor interfaces regarding number of outports
%Elementary Inputs Unused (globally)	Refactor interfaces regarding input effectiveness
Requirements Compliance	Check test against requirements and model
Requirements with Reviewed Testability	Review testability of requirements
Testable Requirements with Assessments	Link test assessments to (covered) requirements
Testable Requirements with Test Sequences	Link test sequences to (covered) requirements
Testable Requirements with Test Sequences	Derive further test sequences from uncovered requirements
Test Sequence Work Progress	Review test sequences
Reviewed Test Sequences	Review test sequences
Test Sequence Compliance	Check expected output against requirements and model implementation
Assessment Work Progress	Review Assessments
Reviewed Assessments	Review Assessments
Assessments Compliance	Check assessment implementation against requirements and model
Model Condition Coverage	Derive further test sequences from uncovered requirements
Model Decision Coverage	Derive further test sequences from uncovered requirements
Run-Time Checks	Improve Run-Time Checks

Figure 4.29: Defining recommended actions for improving quality properties in Excel

4.3.3 Target Values

To configure target values in MQC, the following information is necessary:

- **ArtifactName**

  The name of the artifact a target value shall be applied to.

- **MilestoneName**

  The name of the milestone a target value shall be reached at.

- **MeasureName[.TargetName]**

  The name of the measure, data or quality property, a target value shall be applied to, which is optionally followed by a specific target value name:

  - Targets for data measures (fully qualified):

    DataSourceName.MeasurementName.BaseMeasureName.VariableName[.TargetName]

  - Targets for quality properties:

    QualityPropertyName[.TargetName]

- **TargetValue**

  The value, e.g. a percentage value for quality properties, that is expected to be reached at the date of the configured milestone.

If the configured measure name does not contain a target name at the end, a default target name **"Target"** is applied by MQC instead.

It is not necessary to define a target for each configured milestone (see *Calculation of Target Values per Revision*).

Listing 4.19: Defining target values per measure, per artifact and per milestone in YAML

```
Artifacts:
- Name: ManageVehicleStates
  Milestones:
  - Name: Model Unit Start
    Targets:
    - Name: Guideline Compliance (GuidelineAnalysis)
      Value: 80
    - Name: Requirements Compliance
      Value: 60
  - Name: Model Unit Extension
    Targets:
    - Name: Guideline Compliance (GuidelineAnalysis)
      Value: 80
    - Name: Requirements Compliance
      Value: 80
- Name: EV3Control_main
```

(continues on next page)

```
Milestones:
- Name: Model Unit Start
  Targets:
  - Name: Guideline Compliance (GuidelineAnalysis)
    Value: 0
    - Name: Requirements Compliance
      Value: 0
  - Name: Requirements with Reviewed Testability
    Value: 0
```

In Excel, the target value configuration is done in a sheet called *Target Values*.

ArtifactName	MilestoneName	Guideline Compliance (GuidelineAnalysis)	Requirements Compliance	Requirements with Reviewed Testability	Testable Requirements with Assessments
ManageVehicleStates	Model Unit Start	80	60	80	80
ManageVehicleStates	Model Unit Extension	80	80	90	90
ManageVehicleStates	Model Integration	90	100	100	100
ManageVehicleStates	Test Improvement	90	100	100	100
EV3Control_main	Model Unit Start	0	0	0	0
EV3Control_main	Model Unit Extension	0	0	0	0
EV3Control_main	Model Integration	90	90	90	90
EV3Control_main	Test Improvement	90	100	100	100
GlobalPosition	Model Unit Start	80	60	80	80
GlobalPosition	Model Unit Extension	80	80	90	90
GlobalPosition	Model Integration	90	100	100	100
GlobalPosition	Test Improvement	90	100	100	100
ObstacleDetection	Model Unit Start	80	60	80	80
ObstacleDetection	Model Unit Extension	80	80	90	90
ObstacleDetection	Model Integration	90	100	100	100
ObstacleDetection	Test Improvement	90	100	100	100

Figure 4.30: Defining target values per measure, per artifact and per milestone in Excel

Details about how to switch on or off targets in MQC can be found in *Target values in visualizations (Custom pages)*.

4.3.4 Annotations

For a proper description of necessary configuration parameters, refer to *Annotations*.

Mandatory parameters are:

- Title

- Artifact

- Quality Property

Listing 4.20: Defining annotation source in YAML

```
Authors:
- Model Engineering Solutions GmbH
Annotations:
- Title: 80% because halted for technical reasons
  Author: mqc_admin
  Artifact: EV3Control_main
  QualityProperty: Code Condition Coverage
  ValidFrom: 2021-10-11
  ValidTo: 2021-10-17
  ConditionType: Bin
  ConditionBins:
    - Acceptable
  TargetType: Quality
  TargetValue: 80.00
- Title: Bad because not enough tests
  Author: mqc_admin
  Artifact: GlobalPosition
  QualityProperty: Testable Requirements with Test Sequences
  ValidFrom: 2021-10-11
  ValidTo: 2021-10-17
  ConditionType: Bin
  ConditionBins:
    - Good
  TargetType: Bin
  TargetValue: Bad
- Title: Because Model Coverage - Decision under 80%
  Description: see Data Origins
  Author: mqc_admin
  Artifact: ObstacleDetection
  QualityProperty: Model Decision Coverage
  ValidFrom: 2021-10-11
  ValidTo: 2021-10-17
```

(continues on next page)

```
ConditionType: Bin
ConditionBins:
  - Acceptable
TargetType: Quality
TargetValue: 75.00
```

All other configurations are optional and can be added respectively edited later directly in MQC (see *Annotations*).

4.4 SETTINGS
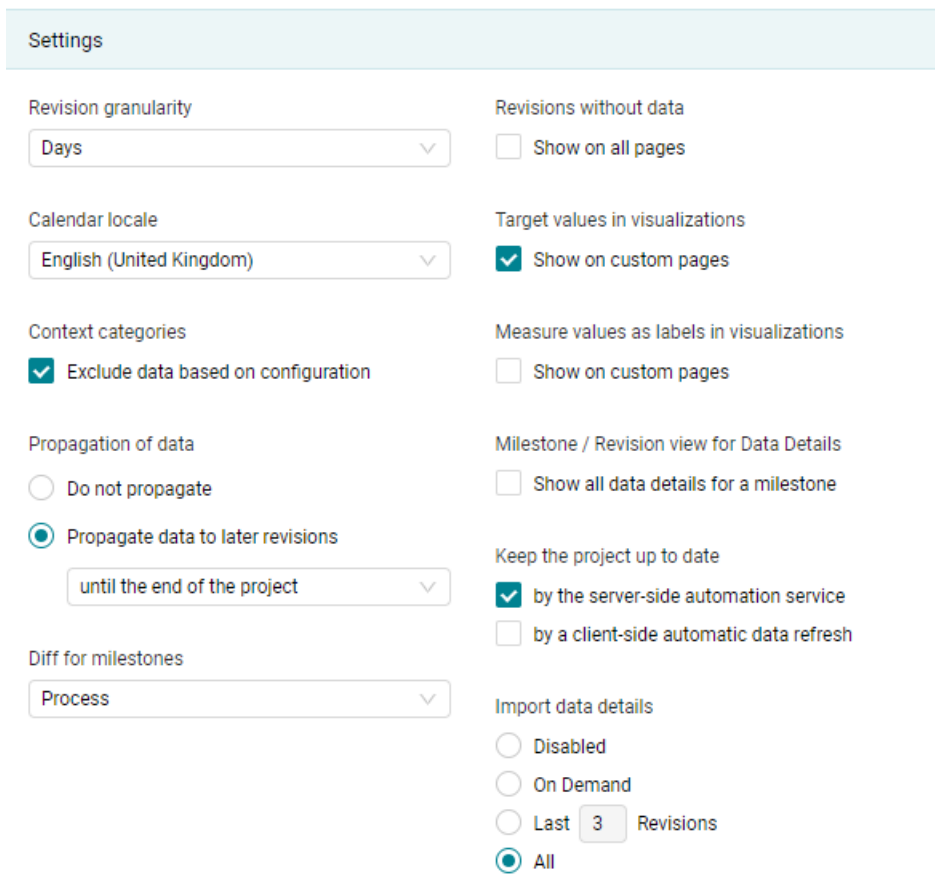
Figure 4.31: The settings dialog in MQC

4.4.1 Revision granularity

This setting defines the degree of compactions of the revisions for which data exists to better visualize your trend visualizations. The revision granularity can be selected depending on your needs.
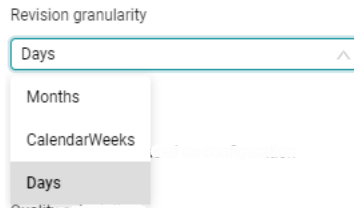
Figure 4.32: The default revision granularity is *Days*, but it might be useful to change it to *Calendar-Weeks* or *Months*

4.4.2 Calendar locale

The calendar locale definition shows the culture information that was used during the creation of the current project.

The formatting of dates in dialogs and the date range selection in the toolbar are based on the calendar locale.

The locale is also respected in the calulation of revision names, when the revision granularity is set to calendar weeks (see *Revisions*).
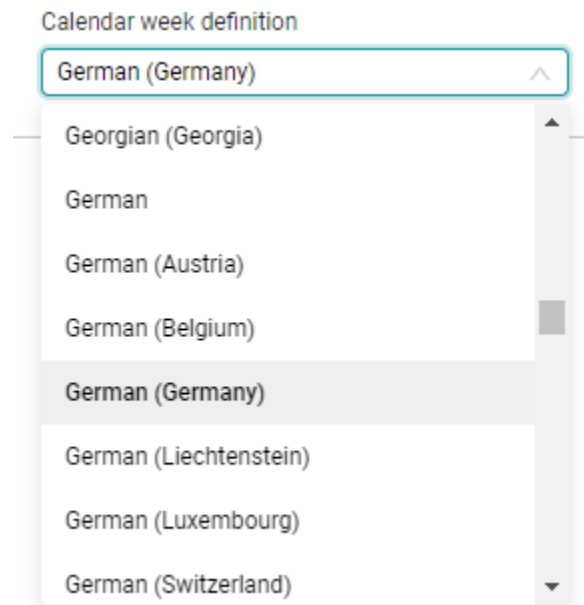
Figure 4.33: Select the calendar locale

If an MQC project is shared between areas with different locales, the calendar locale, which was initially determined during the creation of the project, is kept. This allows a consistent view on data and quality.

4.4.3 Context Categories

Per default the usage of Context Categories is disabled, hence, all data is expected and shown for all artifacts. To apply the context category configuration, enable this setting.

With context categories enabled, for each artifact only expected data is shown in all visualizations (see Figure 4.34). All other data is excluded (white areas of the matrix). By this, you can easily distinguish between not expected data and data that is really missing. (see context categories in *Artifacts*)
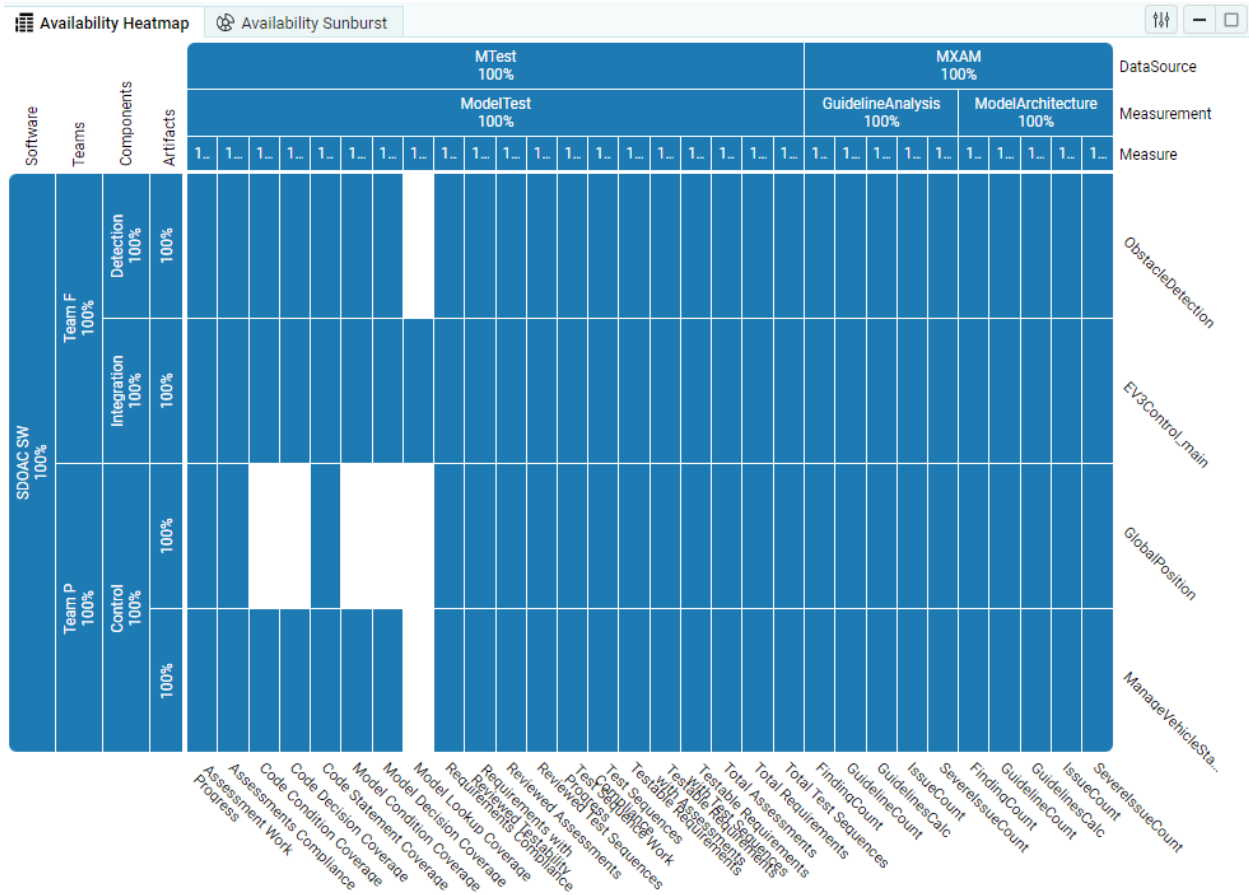
Figure 4.34: Availability Heatmap showing expected data (Context Categories enabled).

Additionally, only expected data is used for calculating quality for an artifact.

4.4.4 Propagation of data

MQC offers the feature to propagate data, that is missing at certain revisions, by using data previously imported instead of importing the data again.

The propagation can be enabled for the whole project, or as propagation between milestones. (see *Data Propagation*)

4.4.5 Diff for milestones

The Diff View (see diff-mode) shows the differences between revisions.

By selecting a milestone structure for "Diff for milestones", an additional comparison is created that compares each revision to the last milestone of this structure.

The Diff base can be switched in the toolbar between the Revision and Milestone comparisons.
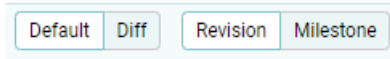
Figure 4.35: Toolbar with Diff Base next to View

4.4.6 Revisions without data

By default MQC hides all empty revisions, for which no data was imported.

This setting allows you to display these configured but empty revisions for all artifacts and measures within this project.

4.4.7 Target values in visualizations (Custom pages)

To make imported *Target Values* visible in the corresponding visualizations on custom pages (see *Custom Pages*), enable this setting.

Targets per measure are shown as:

- separate dashed lines with the same color in trend visualizations
- horizontal lines in status (bar) visualizations
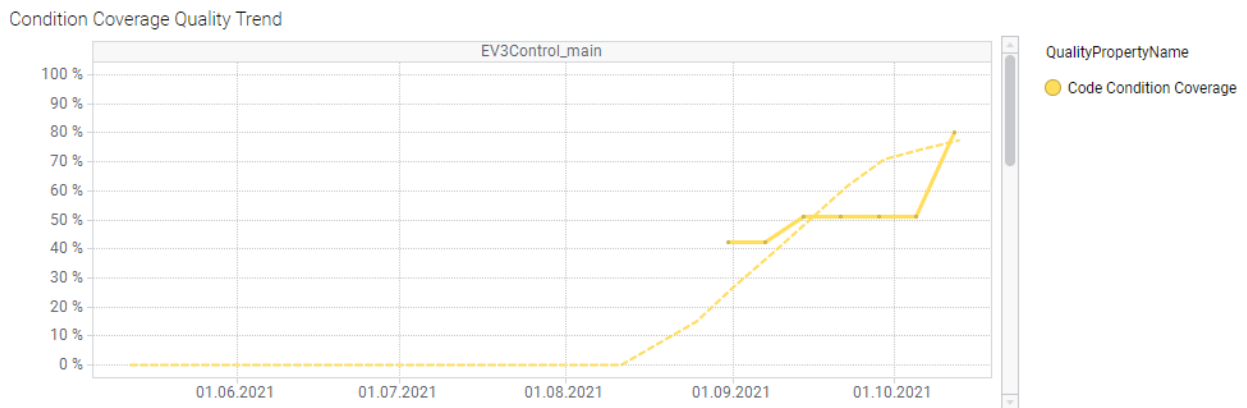
Figure 4.36: Trend visualization showing a quality property with a corresponding target value

Guideline Compliance Status
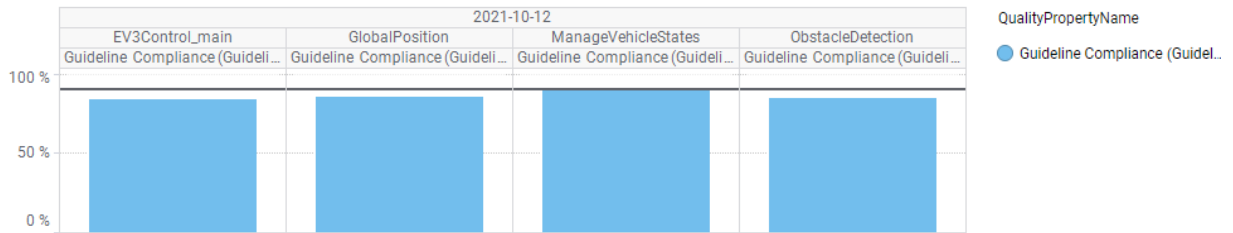
Figure 4.37: Status visualization showing measures with a corresponding target values

Guidelines Status
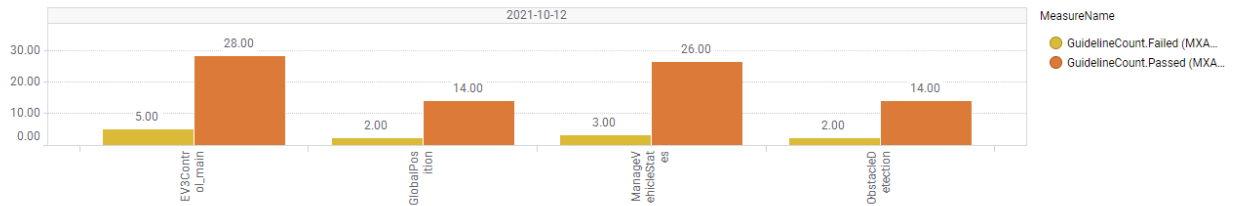
Condition Coverage Quality Trend
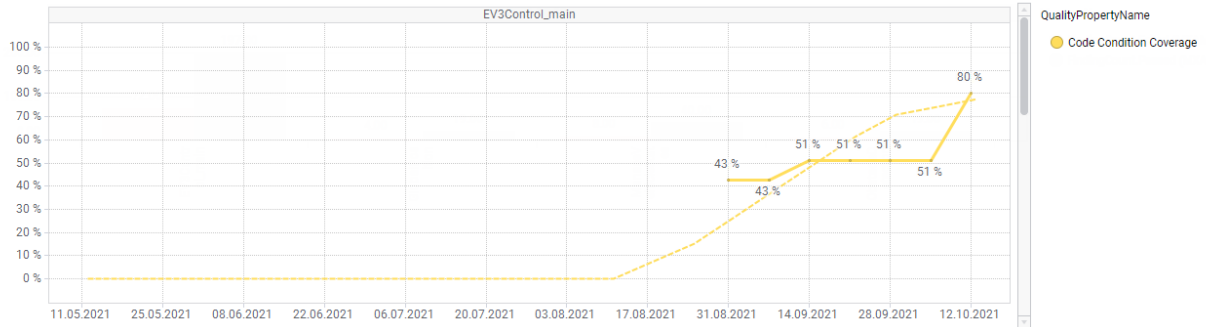
Figure 4.38: Trend visualization and Status visualization showing value labels

4.4.8 Measure values as labels in visualizations (Custom pages)

Value labels can be made visible for status and trend visualizations in custom pages. (see *Custom Pages*)

Enabled value labels are also included in the visualizations of a created report. In a Report value labels provide an alternative to the information that is normally only available via tooltips.

4.4.9 Milestone / Revision view for Data Details

By checking "Show all data details for a milestone" the revision selection on the data details page gets replaced with a milestone selection.

The status visualizations show all findings since the last milestone up and including to the selected milestone.

This setting is useful if the data was configured as incremental and the findings are therefore also only imported incrementally.

4.4.10 Keep the project up to date

- **by the server-side automation service**

  MQC projects with this setting enabled and saved in the server library are updated periodically to fetch the latest data changes.

  Background server-side updates are only executed if new or changed data was detected. This ensures that all relevant projects are always kept up to date.

  See *Update Projects on Server* for details on how to add a job to the "Automation Services" on the MQC Server to periodically check and update MQC projects.

- **by a client-side automatic data refresh**

  MQC projects open in the desktop client and web player use a background monitoring to detect changes in the data of the projects.

  While this setting is disabled, the user is informed of updates by a change in the *Data Import State*. The user can then update the project by clicking on the "Refresh Data" button provided there.

  While this setting is enabled, any change detected in the data results in a directly executed update of the project in the client, while the user is interrupted in his work and has to wait until the project has been updated.

4.4.11 Import data details

If Data Details are not disabled, they are imported by the DataSource adapters that can read findings. (at the moment only the MXAMmxmr Adapter supports reading findings)

- **Disabled**

  Data Details are disabled.

The DataSource Adapters do not import findings. The Data Details page and Data Details visualizations are not available.

- **On Demand**

  Data Details will not be imported by default, but can be loaded on demand later.

  The Data Details visualizations show 'Not Loaded' findings, while the List shows a warning message, if data details have not been imported.

  By clicking on the **Imported data details** button, the data details to be imported can be configured with the combination of artifacts, datasources and revisions.

Figure 4.39: Imported Data Details Dialog

- **Last [Number] Revisions**

  Data Details are imported and transformed for all artifacts for a number of the last revisions.

- **All**

  Data Details are imported and transformed for all artifacts and all revisions.

4.5 DATA SOURCES

To import data from reports in MQC, the data has to be provided as files. These files can be located in the local file system, a network path or a git repository.
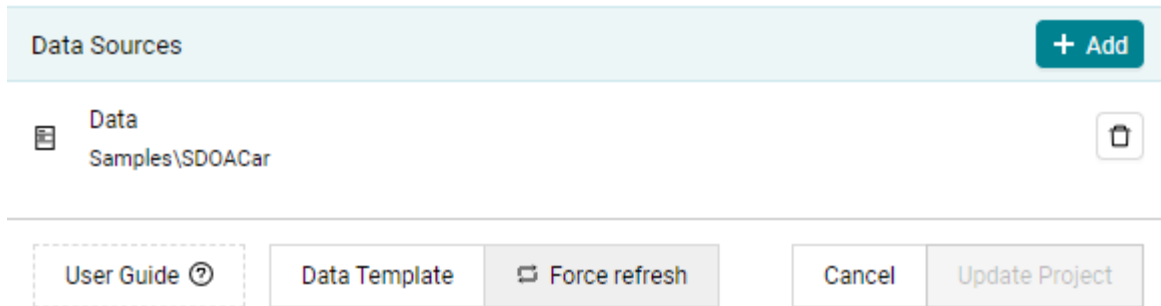
Figure 4.40: Data sources in MQC. By clicking on "Add" a new data source can be configured.

A new data source can be added through a browser in the data source dialog. The type of the data source has to be selected first and then the respective directory, files or git repository can be added. You can add multiple data sources to your project.

Each data source is constantly monitored for updates and changes. Whenever e.g. a new report appears in a data source (new file in directory or new commit in version control system), it will be detected by MQC. Either the user is notified about the changes or the project is automatically updated.

4.5.1 Local File System / Network / Sample Projects

The local file system and network types function similarly. A directory, and therefore all subdirectories and files within, or a single file can be selected as the data source. At any point, the allowed file extensions that can be read by MQC are displayed below.

Navigating through the file system works like a windows explorer. Double-clicking a directory opens it. The current path is displayed at the top of the browser and can be used to navigate upwards in the file system. You can also add the path manually.

The local file system of the server is not accessible when using the web player.

You have also direct access to the MQC Showcase source files under `Sample Projects` (see Figure 4.41).

4.5.2 Version Control Systems

MQC also allows version control systems to be used as a data source. Currently only Git is supported.

Git

After selecting Git as data source type, a Git repository URL can be added. In case you have multiple repositories with a similar structure, you can configure them as one data source. Use the `Add Repository` button to extend the list of repository URLs or paste a list of repositories to the edit field, which automatically leads to multiple lines (see Figure 4.44).

Each repository is checked then for availability and accessibility separately. If an authentication is required, you will be asked for your credentials.
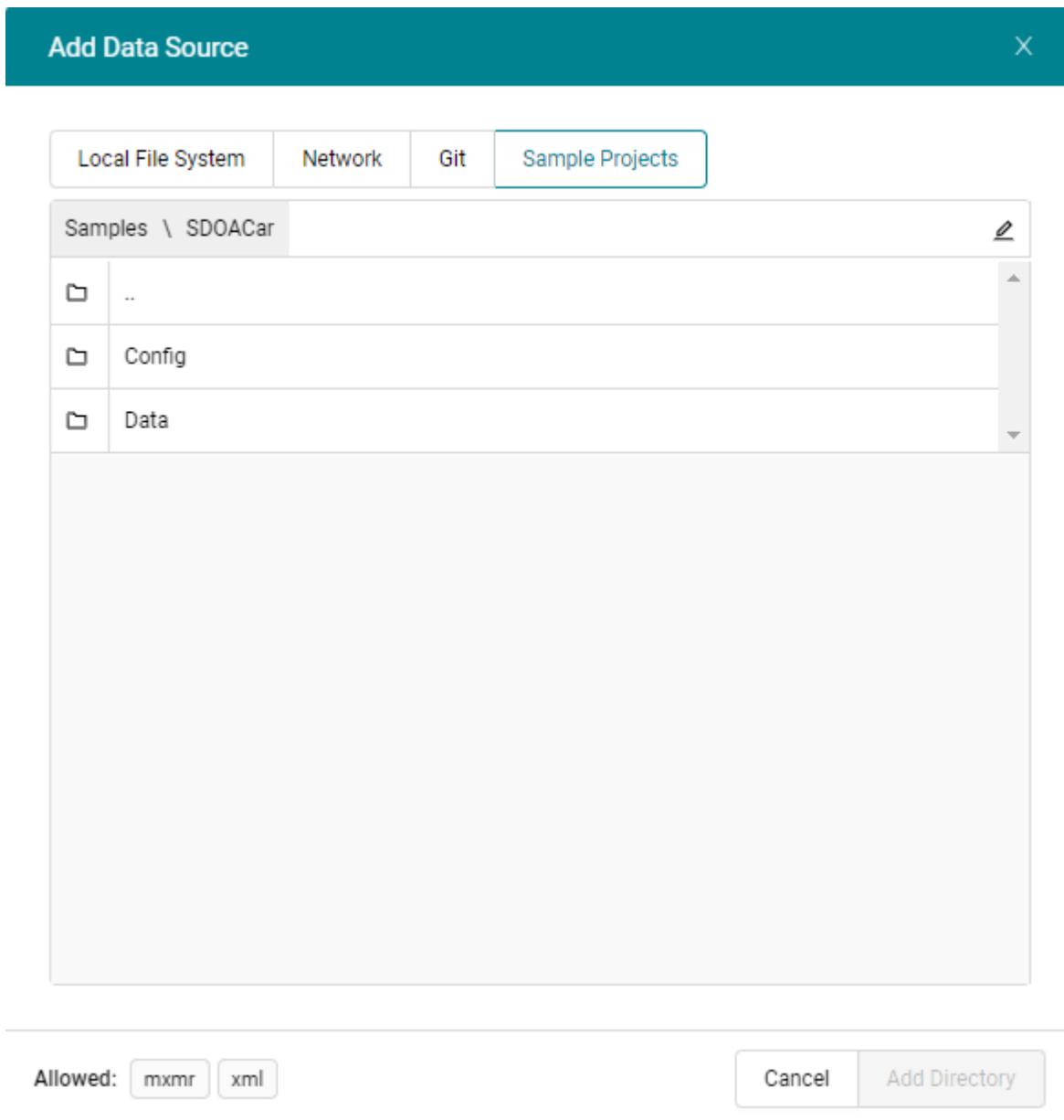
Figure 4.41: The Browser to add a new Data Source
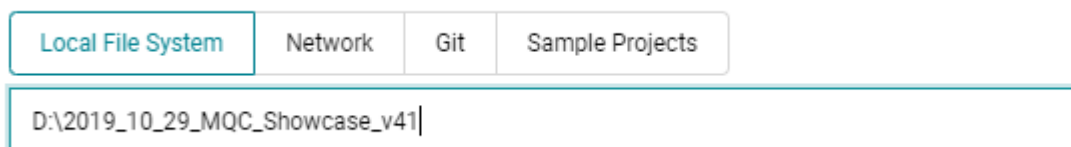
Figure 4.42: Local File System selected. Add/Edit the path manually.
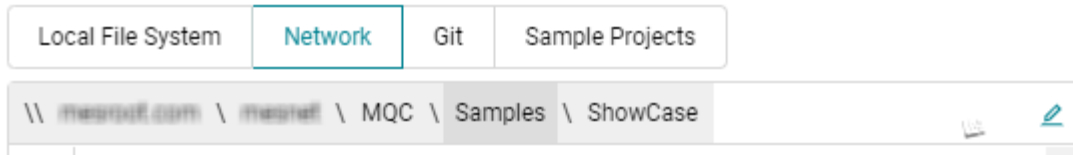
Figure 4.43: Network selected. The path can be used to navigation upwards by click.

If git is not installed, locally (if using the MQC desktop client) or on the server (if using the web player), an error is shown.

Git for windows can be found here: https://git-scm.com/downloads

Git repositories can be added by ssh or https urls.

An rsa private key can be provided for authentication of ssh urls if it is not configured as the default ssh key for the current user.

For an https url the authentication has to be done by entering the username and password, if it is not already stored in the windows credential store.
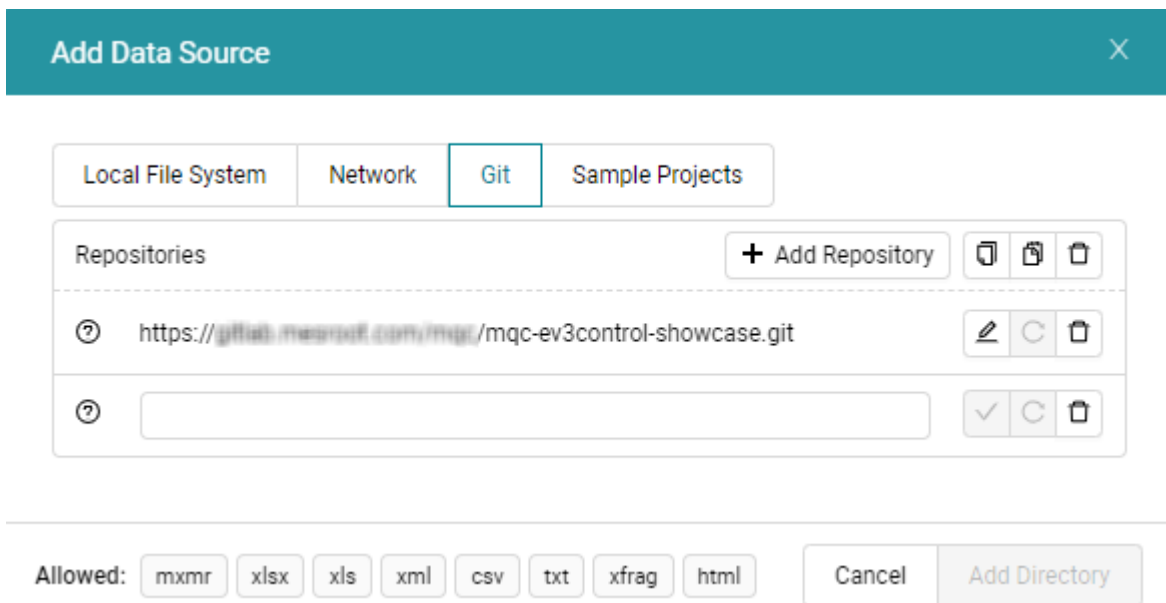
Figure 4.44: Git Repositories dialog to add, edit and remove git repository(ies) to be configured as one data source.

If the configured repository(ies) contain multiple branches, you can select a branch via the drop down next to the list of repository URLs.

Only branches that exist in all repositories can be selected, when multiple Git repositories are given.

Since your commit history might contain commits not relevant for your MQC quality analysis, you have the possibility of filtering these commits in the Commit Filters section of this dialog (see Figure 4.45).

The Commit Filters section provides three possible targets to filter, namely, `Message`, `Author` and `Tag`. You

Figure 4.45: Configuration options to add a commit filter.

can further choose to either include or exclude the commits matching this filter by clicking on the toggle button below `Apply`. The green colored + stands for include and the red colored − stands for exclude. Define a suitable regular expression in the `RegEx` text box. All the filters to the same target are connected with an `OR` whereas different targets are connected with an `AND`.

Figure 4.46 shows a filter that matches all commits `From the Author "Jenkins Slave" AND with (Message "Static Analysis" OR Message "Dynamic Analysis")`.
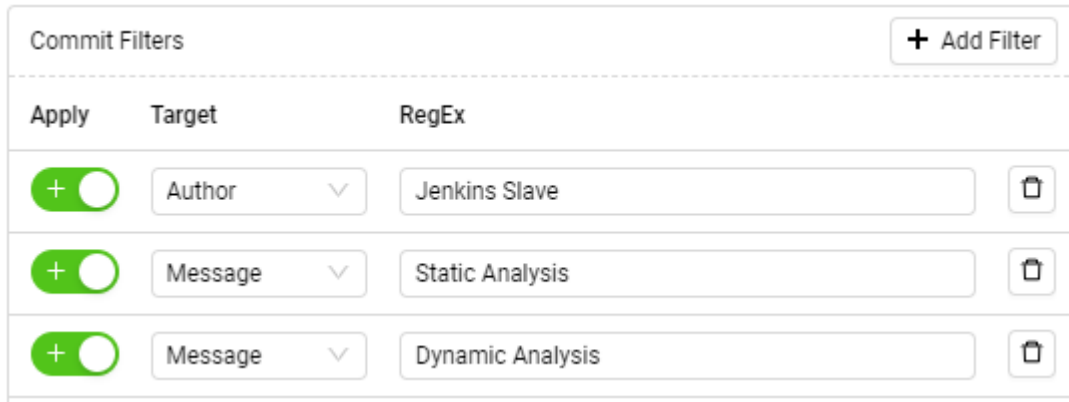
Figure 4.46: Commit Filters with "AND" between different targets and "OR" between the same targets.

Additionally, MQC offers the possibility to filter by time (see Figure 4.45). This allows to consider only commits of a specific time frame. First and last commit time can be configured separately by:

- using the start and end date of the project (as defined by the milestones within a project structure configuration, which is also the default if nothing is set at all)
- selecting a specific date via the date picker.

If no project structure configuration has been added, all commits are fetched from the repository(ies).

If a commit appears outside the configured project time frame, you will be notified about such commits within the Notifications panel. After adapting the project structure configuration, a data refresh is necessary to fetch the relevant commits from the repository.

With the checkbox shown in Figure 4.47 you can define which date and time is used as the report date-time:

- Git commit date (checked)
- Report creation date time as contained in the report itself (unchecked)

In addition to the commit filters, it is also possible to select specific folders or files relevant to your analysis. MQC provides two possibilities to do this.

If you select the `Regular Expressions` option in the drop down, then you can add multiple regular expressions similar to the Commit Filters section.

If you select `Browser Selection` option, one file or folder can be chosen as the data source, similar to selecting directories from a local or a network drive as data source.

Figure 4.47: Enable checkbox to use commit time as report date.

The `Browser Selection` is only available when using a single Git repository as data source. For multiple repositories only `Regular Expressions` could be applied. Additionally, to be able to navigate within and to select specific folders, it is necessary to fetch the repository first by using the `Fetch repository` button in the left-right corner (see Figure 4.50).

Since the version 6.3, MQC supports the partial checkout feature (sparse checkout) provided by Git. Using sparse checkout, no full clone of the git repository is performed. Only the relevant files and directories based on the defined filters are downloaded from the server.

Using the sparse checkout has to be enabled by an administrator. The corresponding configuration is described in the Server Administration Guide.

If sparse checkout is enabled, no browser selection is possible. Additionally, it is not possible to define "exclude" filters for directories as shown in Figure 4.49. Instead of regular expressions for directory paths, the names of the directories have to be used as filter.

If a git data source has been added, a monitoring service periodically checks if a new commit is made on the remote. The default for the check interval is set to 10 seconds. In case of multiple repositories, the supervision timer is stepwise increased for performance reasons based on the number of repositories. I.e. If you have configured more than 100 repositories, the check happens only every 60 seconds.

When data is refreshed, the git repository is updated and all remote changes are fetched and applied to the MQC project. In case some hidden changes were made in your Git repository, use the `Force Refresh` button (see Figure 4.40) to not just apply changes to a project but to re-import all data anew.

4.6 ADAPTERS

Adapters can be Enabled / Disabled in the Adapters Dialog available in the MQC application. By disabling unnecessary File Adapters the speed of importing huge data can be improved.

The allowed file extensions for the Data Sources depend on the enabled Adapters.

Not all available adapters are shown in the dialog. To add a special adapter click on `Add` and then `Special` to add the needed adapter.

Figure 4.48: Selection menu to choose the folder or file path of relevant data by file system or add patterns to match relevant folder or file paths.

Figure 4.49: Regular Expression pattern matching for selecting data source paths.

Figure 4.50: Browser Selection for choosing data source path.
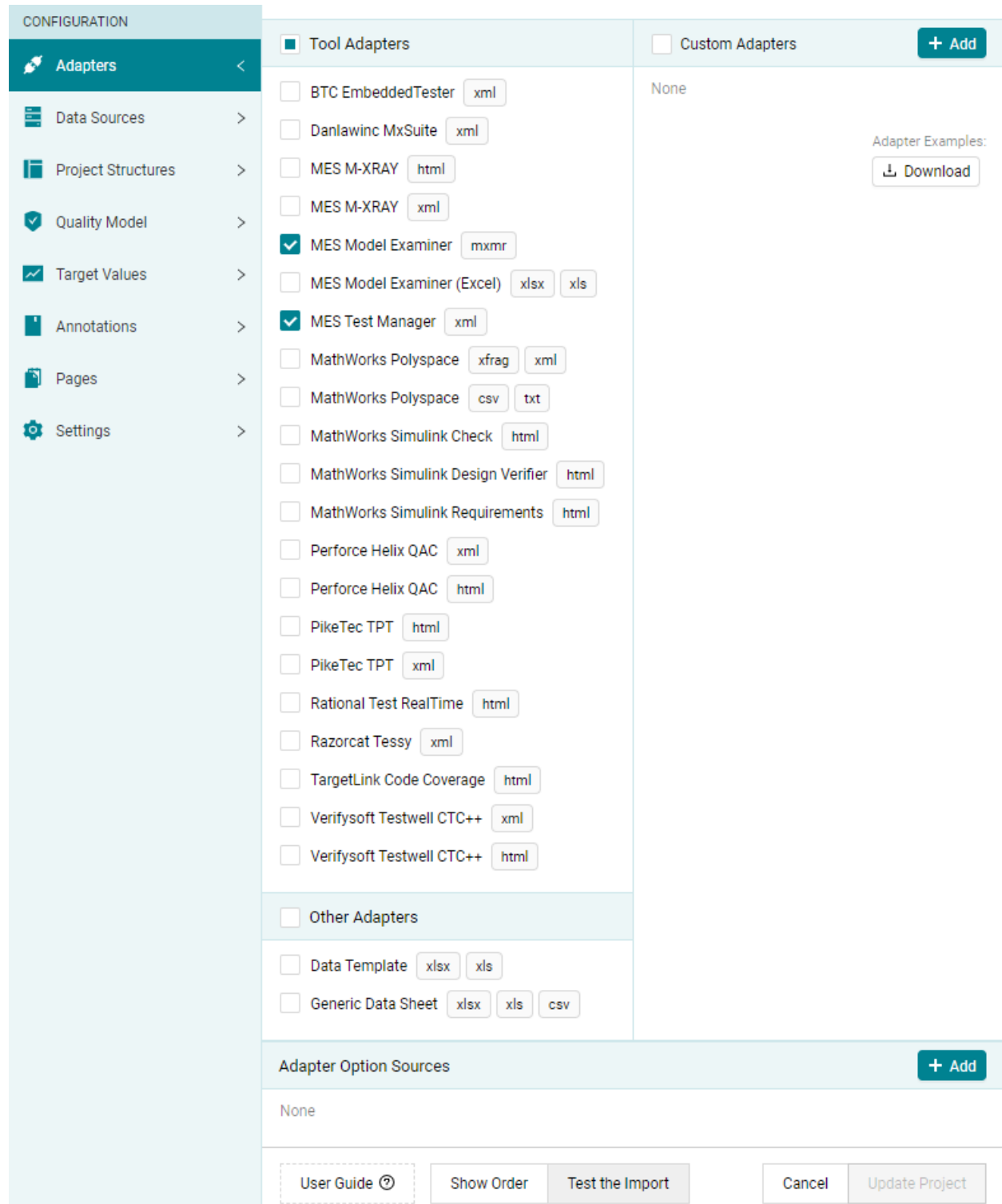
Figure 4.51: Enabling / Disabling File Adapters in the Adapter Dialog
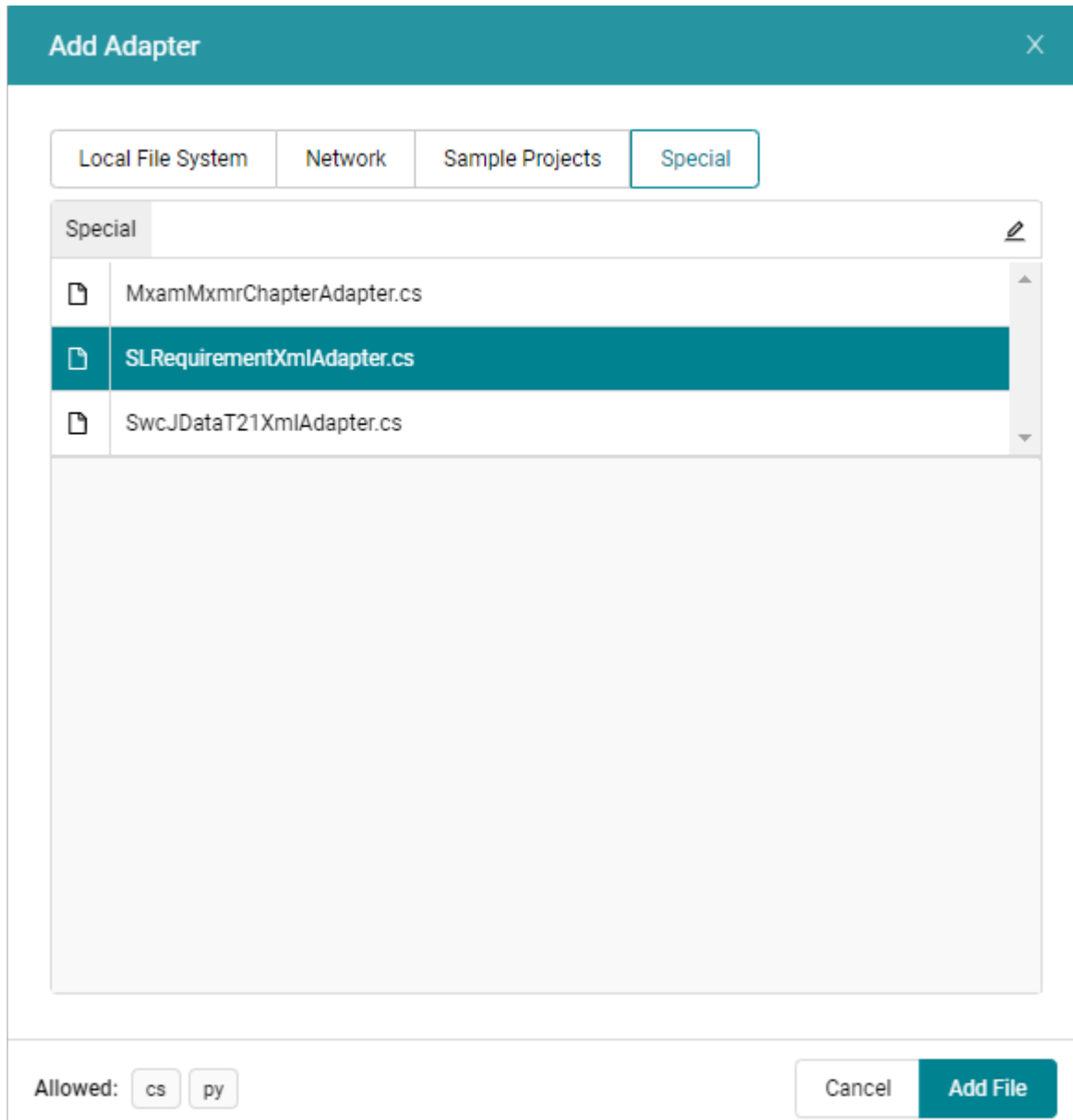
Figure 4.52: Adding a special adapter

4.6.1 File Adapters

The following tools are supported by MQC with default adapters:

- *MES Model Examiner® (MXAM)*
- *MES M-XRAY® (MXRAY)*
- *MES Test Manager® (MTest)*
- *PikeTec TPT*
- *Razorcat Tessy*
- *MathWorks Polyspace*
- *BTC EmbeddedTester*
- *Verifysoft Testwell CTC++*
- *Danlawinc MxSuite*
- *MathWorks Simulink Check*
- *Simulink Design Verifier*
- *Perforce Helix QAC*
- *TargetLink Code Coverage*
- *Rational Test RealTime (RTRT)*
- *MathWorks Simulink Requirements*

CSV files can be imported with the *Generic data sheet* by using Adapter Options.

If you want to import manually collected data use the provided Excel Template and let the *Manual data import (Excel Template)* import it into MQC.

MES Model Examiner® (MXAM)

MQC supports two types of MXAM report formats:

- *MXMR*
- *Excel*

MXMR

The following example of an MXMR report describes which information is imported by MQC's MXAM adapter to MQC:

- from the `<RReport>` element, specifically the `date` element: `ReportDateTime`
- from the first `<subcomponents>` element inside an `<artifacts>` element, the attribute `name` as `ArtifactName` and the attribute `path` as `ArtifactPath`

MXAM provides guideline and finding result data for each artifact in the MXMR Report. Therefore, each artifact section will be parsed to get the information of all findings and guideline results.

```xml
<artifacts result="Aborted" adapterId="com.modelengineers.mxam.tooladapter.matlab"␣
→storageNature="Tool artifact">
  <structure>
    <subComponents name="ExPol" path="ExPol">
    </subComponents>
  </structure>
  ...
  <summary itemType="Findings">
    <statistic resultType="Review" count="3"/>
    <statistic resultType="Failed" count="66"/>
    <statistic resultType="Info" count="28"/>
    <statistic resultType="Passed" count="12"/>
    <statistic resultType="Ignored" count="2"/>
  </summary>
  ...
  <summaries itemType="Guidelines">
    <statistic resultType="Review" count="1"/>
    <statistic resultType="Failed" count="5"/>
    <statistic resultType="Passed with Infos" count="24"/>
    <statistic resultType="Passed" count="11"/>
  </summaries>
</artifacts>
```

The adapter reads those measures to be found in the Findings and Guidelines header: `<summary itemType="Findings">` and `<summaries itemType="Guidelines">`. Please note that an MXAM report can contain various artifacts and for each artifact MQC reads out the Findings and Guidelines Summary, that are saved as `Finding\Result` and `Guideline\Result`, respectively:

- `Review`

- `Failed`

- `Info` (for Finding\Result) and `Passed with Infos` (for Guideline\Result)

- `Passed`

- `Ignored`

- `Aborted`

- `Canceled`

- `Repaired`

- `Unrepaired`

- `Warning` (for Finding\Result) and `Warnings` (for Guideline\Result)

The adapter also reads the `Model Architecture` chapter data separately with `ModelArchitecture` as MeasurementName. For this, the adapter extracts all findings related to this chapter and aggregates the finding results from the `result` attribute for each artifact to define the `Finding\Result` measure. To get

the `Guideline\Result` measure, the adapter determines the worst finding for each artifact and guideline (e.g. `MXRAY_COMPLEXITY_LOCAL`) and aggregates again equal results.

```
<findings xsi:type="MatlabReport:RMatlabFinding" result="Passed" path="ExPol" name=
↪"ExPol" qualifier="Model"
        checkTreePath="mes_guidelines_embedded_coder_fs/Model Architecture/mes_arch_
↪1301/matlab_mxray_1301"
        check="//@project/@documents.0/@chapters.1/@guidelines.0/@checks.0"␣
↪artifact="//@artifacts.0"
        artifactStructureComponent="//@artifacts.0/@structure/@subComponents.0"␣
↪ignoreComment=""
        parentPath="ExPol" repairInfo="">
    <properties key="Mask type" value="" visible="false"/>
    <properties key="Block type" value="" visible="false"/>
    <properties key="MessageParameter" value="[local complexity, 22]" visible="false"/
↪>
    <properties key="metric" value="MXRAY_COMPLEXITY_LOCAL" visible="false"/>
    <message messageId="BoundCheck_GOOD" messageText="The local complexity is 22.">
        <messageParameter>local complexity</messageParameter>
        <messageParameter>22</messageParameter>
    </message>
    <linkAction>
        <properties key="label" value="Open Model" visible="true"/>
        <properties key="link" value="matlab:open_system('EV3Control_demo_ec');"␣
↪visible="true"/>
    </linkAction>
    <elementIdentifier xsi:type="XMatlab:XMatlabElementIdentifier"
                    elementIdentifier="ExPol,&#xA;ExPol,&#xA;Model,&#xA;BoundCheck_
↪GOOD:
                                    The local complexity is 22."
    path="ExPol" name="ExPol" qualifier="Model" artifactName="ExPol" version="5.1.0.
↪xMessage" sid="">
        <message messageId="BoundCheck_GOOD" messageText="The local complexity is 22.
↪">
            <messageParameter>local complexity</messageParameter>
            <messageParameter>22</messageParameter>
        </message>
    </elementIdentifier>
</findings>
```

MQC reads and shows the Model Architecture results separately, it is not contained in the overall Guideline\Result and Finding\Result measure values.

MQC extracts all findings, generates a separate Data Detail for each finding and reads:

- `Check` as `Subject`

- the value of `check` attribute, which is used to fetch the path of the check inside the guideline document tree the finding belongs to, and stores it as `Subject Path`

- `path` attribute of findings element as `Artifact Inner Path`

- `Result` as `Measure`

- `result` attribute of findings element as `State`

- `messageText` or `text` attribute of message element as `Issue Message`

Excel

MQC accepts an Excel file as a valid MXAM report file, if it at least contains the `Project Overview` sheet. All guideline results and finding results are read then from the `Findings` sheet taking into account that multiple `Findings` sheets may be present (e.g. in addition `Findings 2`, `Findings 3` etc).

MQC reads

- from the `Project Overview` sheet:

    - the `ReportDateTime` as stored in the row that contains the string "Generated at:"

- from the `Findings` sheets EACH row as an MXAM finding, where:

    - the `Check ID` column is used to extract the guideline name

    - the `Objectives` column is used to extract the `MeasurementName`

    - the `Artifact` column is read as `ArtifactName`

    - the `Result` column is read as `VariableName`.

To extract the guideline name from the entry of the `Check ID` column, MQC splits the given path into its parts, where the last part is the check and the part next to the last indicates the guideline. If for example the column contains the string

```
mes_first_set_modeling_guidelines_fs/Layout and Design/Modeling of Data Flow/
misra_slsf_030_abc/mcheck_misra_slsf_030_ab
```

- `mes_first_set_modeling_guidelines_fs` is the name of the document

- `Layout and Design` is a chapter inside the document

- `Modeling of Data Flow` is a subsection of the above chapter

- `misra_slsf_030_abc` is the guideline and

- `mcheck_misra_slsf_030_ab` is the check.

If the column `Objectives` is not existing, MQC instead checks for the column name `Check Type` to extract the measurement name. If multiple entries are contained, MQC uses the first entry as measurement name.

If the MXAM Excel report contains findings for different subsystems for the same artifact, MQC tries to extract a common prefix of all artifact paths as `ArtifactName`.

After reading all rows, the extracted findings (each row equals one MXAM finding) are aggregated and stored as MXAM measures:

- the sum of all findings with the same measurement name, the same artifact name and the same variable name is stored as `MeasurementName.FindingCount.VariableName = RowCount`, e.g. `Functionality.FindingCount.Passed = 212`.

To extract the `GuidelineCount` measure:

- MQC first groups all findings according to the same measurement name, the same artifact name and the same guideline name

- then for each group MQC takes the variable name of the worst finding as guideline result, e.g. `Failed`

- and afterwards counts the occurrence of findings with the same measurement name, the same artifact name and the same variable name as `MeasurementName.GuidelineCount.VariableName`, e.g. `Functionality.GuidelineCount.Failed = 15`.

MES M-XRAY® (MXRAY)

MQC supports the Standard XML MXRAY Report file.

The following information is extracted by the MQC/ MXRAY adapter:

- from the `<Timestamp>` element

  - `ReportDateTime`

- from the `<SubsystemQualityOverview>` header all included elements, usually these are:

  - `Local Complexity`

  - `Level`

  - `%Elementary Inputs Unused (globally)`

  - `Cyclomatic Complexity`

  - `Inports`

  - `Outports`

  For each of these Measures the variables `Good`, `Acceptable` and `Bad` are imported. Furthermore for `Local Complexity` the variables `LowerBoundOfAcceptable` and `LowerBoundOfBad` are read .

- from the `GlobalValueSummary` element

  - `Global Complexity (Ref0)`

  - `Global Complexity (Ref1)`

  - `Global Complexity (RefN)`

  - `Global Complexity Stateflow (Ref0)`

  - `Global Complexity Stateflow (Ref1)`

  - `Global Complexity Stateflow (RefN)`

- from the `CloneGroups` element

  - `NumberOfDetectedCloneGroups`

  - `NumberOfSubsystemsAnalyzed`

  - `NumberOfUniqueSubsystemsInAllCloneGroups`

- `NumberOfSubsystemsInAllCloneGroups`

In MQC `CloneGroups.NumberOfUniqueSubsystemsInAllCloneGroups` is shown as `CloneGroups.Bad`:

`Bad = NumberOfUniqueSubsystemsInAllCloneGroups`

Additionally `CloneGroups.NumberOfSubsystemsAnalyzed` is used together with `CloneGroups.NumberOfUniqueSubsystemsInAllCloneGroups` to calculate a value for `CloneGroups.Good`:

`Good = NumberOfSubsystemsAnalyzed - NumberOfUniqueSubsystemsInAllCloneGroups`

MES Test Manager® (MTest)

The MQC-MTest adapter supports the HTML format for MTest Report files.

MQC expects a test catalog html file, where the file name starts with "**Testcatalog_**". In addition, it reads data from the assessment catalog as well as from the requirement catalog if located in the same folder as the test catalog.

Extracting the data from a zip archive is also supported. Then the archive name must start with "**Testcatalog_**" as well.

The following information is extracted by the adapter:

From the `<table class="infoHeaderTable">` table MQC extracts the value of `Generated at:` as `ReportDateTime`.

```
<body>
  <h1>MTest Sequence Catalog: MTest Demo Autopilot_ec Batch Project</h1>
    <div class="infoHeader">
    <table class="infoHeaderTable">
      <tr><td>MTest Project: </td><td>MTest Demo Autopilot_ec Batch Project</td></tr>
      <tr><td>Project Author: </td><td>MTest Demo User</td></tr>
      <tr><td>Generated at: </td><td>2023-04-12 21:20</td></tr>
    </table>
  </div>
```

For each catalog file the artifact names are extracted from the list of test objects. Each artifact is a concatenated string using test model and test object values, e.g. "autopilot_demo_ec/Autopilot_Mode_Logic".

```
<ol>
  <li>Test Model: <a href="...">autopilot_demo_ec</a>
    <ol>
      <li>Test Object: <a href="...">Autopilot_Mode_Logic</a>
        <ol>
          <li>Test Group: <a href="...">Test001 – Altitude mode logic functionality␣
→testing using MTCD (text-based)</a></li>
          <li>Test Group: <a href="...">Test002 – Altitude mode logic functionality␣
→testing using MTCD (Excel-based)</a></li>
```

(continues on next page)

```
      <li>Test Group: <a href="...">Test003 - Altitude mode logic functionality␣
→testing via Variation in MTCD (text-based)</a></li>
      </ol>
    </li>
    <li>Test Object: <a href="...">Glide_Slope_Control</a>
      <ol>
        <li>Test Group: <a href="...">Test001 - Glide slope controller␣
→functionality testing using MTCD (text-based)</a></li>
        <li>Test Group: <a href="...">Test002 - Glide slope controller␣
→functionality testing using MTCD (Excel-based)</a></li>
        <li>Test Group: <a href="...">Test003 - Glide slope controller␣
→functionality testing via variation test case in MTCD (text-based)</a></li>
      </ol>
    </li>
  </ol>
  </li>
</ol>
```

Test Catalog

MQC extracts all test case information from the third table of each test group section.

```
<h3>1.1.1 Test Group: Test001 - Altitude mode logic functionality testing using MTCD␣
→(text-based)</h3>
<table width="100%" class="contentCheck">...</table>
<table class="content" width="100%">...</table>
<table width="100%" border="1" cellpadding="5">
  <tr><th>Test Sequence</th><th>Linked Items</th><th>Test Sequence Description</th>
→<th>Evaluation Results</th></tr>
  <tr>
    <td>
      <a id="Mod_autopilot_demo_ec_Autopilot_Mode_Logic_Test001_TSeq001"></a>
      <a href="...">01: TS_AP_ML_001</a></b>:<span class="passed">Passed</span><br />
      <em class="small">autopilot_demo_ec/<br />Autopilot_Mode_Logic/<br />Test001</
→em><br /><br />
      <b>Work Status:</b>reviewed
    </td>
    <td>
      <b>Requirements</b><br />
      <a class="normal" href="...">AP_ML_req_01</a><br /><br />
      <b>Assessments</b><br />
      <a class="normal" href="...">assmnt_AP_ML_req_01</a><br />
      <a class="normal" href="...">mars_AP_ML_req_01</a><br /><br />
    </td>
    <td>
      <b>Glideslope armed 01</b><br />
```

```html
    Motivation<br />
    ...
  </td>
  <td>
    <b>Assessment:</b><span class="passed">Passed</span><br />
    <b>Regression:</b><span class="none">None</span><br />
    <b>Back2Back:</b><span class="passed">Passed</span><br />
    <b>evaluated by:</b><br />MTest Demo User (12-Apr-2023 21:05:01)<br />
  </td>
</tr>
<tr>...
</table>
```

MQC iterates over each row of the table and counts the number of the same results. Afterwards the measures of all test groups belonging to the same artifact (test object) are aggregated by sum.

- For `Testcase\Result` the content of the first column is checked, e.g. `<span class="passed">Passed</span>`.

  The following numbers are collected:

  - `Testcase\Result.Passed`

  - `Testcase\Result.Review`

  - `Testcase\Result.Failed`

  - `Testcase\Result.Error`

  - `Testcase\Result.Untriggered`

  - `Testcase\Result.Inactive`

  - `Testcase\Result.Unknown`

  - `Testcase\Result.None`

  - `Testcase\Result.Total`

- For `Testcase\Work Status` the content of the first column is checked, e.g. `<b>Work Status:</b>reviewed`.

  The following numbers are collected:

  - `Testcase\Work Status.Undefined`

  - `Testcase\Work Status.Created`

  - `Testcase\Work Status.Reqchanged`

  - `Testcase\Work Status.Described`

  - `Testcase\Work Status.Inwork`

  - `Testcase\Work Status.Rejected`

  - `Testcase\Work Status.Completed`

- `Testcase\Work Status.Reviewed`

- `Testcase\Work Status.Total`

- For `Testcase\Requirement Coverage`, MQC checks, if the second column contains any require-ments for a test case.

  The following numbers are collected:

  - `Testcase\Requirement Coverage.Covered` (number of testcases with linked require-ments)

  - `Testcase\Requirement Coverage.Uncovered` (number of testcases not linked to any re-quirement)

  - `Testcase\Requirement Coverage.Total` (number of all listed testcases)

MQC extracts all structural coverage information from the first table of each test object section starting with the 8th row (previous rows contain the test case summary for the corresponding test object).

```html
<h2>1.1 Test Object: Autopilot_Mode_Logic</h2>
<table width="100%" class="coverage">
  <tr><td colspan="8" class="tableabove">Test Sequence State</td></tr>
  <tr>...</tr>
  <tr>...</tr>
  <tr>...</tr>
  <tr>...</tr>
  <tr><td colspan="8" class="tableabove">Structural Coverage</td></tr>
  <tr><th>Tool</th><th>Metric</th><th colspan="2">Percentage</th><th>Target</th><th>
→Total</th><th>Reached</th><th>Unreached</th></tr>
  <tr>
    <td rowspan="3"><a href="...">Model Coverage</a></td>
    <td>Condition</td>
    <td>96.4%</td>
    <td><table class="metricbar">...</table></td>
    <td>80%</td>
    <td>28</td>
    <td>27</td>
    <td>1</td>
  </tr>
  <tr>
    <td>Decision</td>
    <td>81.2%</td>
    <td>...
  </tr>
</table>
```

The following numbers are collected from `Reached`, `Unreached` and `Total` columns, each with the vari-ables `Covered`, `Uncovered` and `Total`:

- `Model Coverage\Condition`

- `Model Coverage\Decision`

- Model Coverage\Statement

- Model Coverage\Lookup

- Code Coverage\Condition

- Code Coverage\Decision

- Code Coverage\Statement

Assessment Catalog

MQC extracts all assessment information from the third table of each test object section.

```html
<h2>1.1 Test Object: Autopilot_Mode_Logic</h2>
<table width="100%" class="coverage">
<table width="100%" class="coverage">
<table width="100%" border="1" cellpadding="5">
  <tr><th>Assessment</th><th>Linked Items</th><th>Assessment Description</th><th
→width="60%"> Test Sequence Results</th></tr>
  <tr>
    <td>
      <a id="Mod_autopilot_demo_ec_Autopilot_Mode_Logic_assmnt_AP_ML_req_01"></a><b>
      <a href="...">assmnt_AP_ML_req_01</a> :</b>
      <span class="failed">Failed</span><br />
      <em class="normal">(Autopilot_Mode_Logic)<br /></em><br />
      ...
      <b>Scope:</b>       global<br /><br />
      <b>Work Status:</b>reviewed<br />
    </td>
    <td>
      <b>Requirements</b><br />
      <a href="...">AP_ML_req_01</a><br /><br />
      <b>Test Sequences</b><br />
      <a href="...">01: TS_AP_ML_001</a><br />
      <a href="...">02: TS_AP_ML_002</a><br />
      ...
    </td>
    <td>
      <b>Check state Armed</b><br />Checks if Armed is true when Glide Slope Enable
→and Glide Slope Signal are both true.
    </td>
    <td>
      <details>
        <summary><b>Test Sequences:</b><span class="failed">Failed</span></summary>
        <div><br />
          <b>Altitude mode logic functionality testing using MTCD (text-based)
→(Test001)</b><br />
          <span class="failed"></span><a href="...">3: TS_AP_ML_003 - Glideslope
→coupled 01</a>: NOT ok. Not Armed when Glide Slope Enable and Glide Slope Signal
→are both true.<br />
```

```html
        <span class="failed"></span><a href="...">4: TS_AP_ML_004 – Glideslope␣
→coupled 02</a>:  NOT ok. Not Armed when Glide Slope Enable and Glide Slope␣
→Signal are both true.<br />
        ...
      </div>
    </details><br />
    <details>
      <summary><b>Test Sequences:</b><span class="passed">Passed</span></summary>
      <div><br />
        <b>Altitude mode logic functionality testing using MTCD (text-based)␣
→(Test001)</b><br />
        <span class="passed"></span><a href="...">1: TS_AP_ML_001 – Glideslope␣
→armed 01</a>: Ok.<br />
        <span class="passed"></span><a href="...">2: TS_AP_ML_002 – Glideslope␣
→armed 02</a>: Ok.<br />
        ...
      </div>
    </details><br />
  </td>
 </tr>
 <tr>...
</table>
```

MQC iterates over each row of the table and counts the number of the same results.

- For `Assessment\Result` the content of the first column is checked, e.g. `<span class="failed">Failed</span>`.

  The following numbers are collected:

    - `Assessment\Result.Passed`

    - `Assessment\Result.Review`

    - `Assessment\Result.Failed`

    - `Assessment\Result.Error`

    - `Assessment\Result.Untriggered`

    - `Assessment\Result.Inactive`

    - `Assessment\Result.Unknown`

    - `Assessment\Result.None`

    - `Assessment\Result.Total`

- For `Assessment\Work Status` the content of the first column is checked, e.g. `<b>Work Status:</b>reviewed<br />`.

  The following numbers are collected:

    - `Assessment\Work Status.Undefined`

- **–** `Assessment\Work Status.Created`

- **–** `Assessment\Work Status.Reqchanged`

- **–** `Assessment\Work Status.Described`

- **–** `Assessment\Work Status.Inwork`

- **–** `Assessment\Work Status.Rejected`

- **–** `Assessment\Work Status.Completed`

- **–** `Assessment\Work Status.Reviewed`

- **–** `Assessment\Work Status.Total`

- For `Assessment\Requirement Coverage`, MQC checks, if the second column contains any requirements for an assessment.

  The following numbers are collected:

  - **–** `Assessment\Requirement Coverage.Covered` (number of assessments with linked requirements)

  - **–** `Assessment\Requirement Coverage.Uncovered` (number of assessments not linked to any requirement)

  - **–** `Assessment\Requirement Coverage.Total` (number of all listed assessments)

Requirement Catalog

MQC extracts all requirement information from the forth table of each test object section.

```html
<h2>1.1 Test Object: Autopilot_Mode_Logic</h2>
<table width="100%" class="contentCheck">...</table>
<table width="100%" class="coverage">...</table>
<table width="100%" class="coverage">...</table>
<table width="100%" border="1" cellpadding="5">
  <tr><th>Requirement</th><th>Linked Items</th><th>Requirement Description</th><th>
→Additional Document Fields</th></tr>
  <tr>
    <td>
      <a id="Mod_autopilot_demo_ec_Autopilot_Mode_Logic_AP_ML_req_01"></a>
      <b>AP_ML_req_01:</b>
      <span class="failed">Failed</span> <br />
      <em class="normal">(Autopilot_Mode_Logic)<br /> </em><br />
      <b>Testability</b><br /><b>Status:</b> Yes<br />
      <b>Review Status:</b> reviewed (derived)<br />
    </td>
    <td>
      <b>Assessments</b><br />
      <a class="normal" href="...">assmnt_AP_ML_req_01</a><br />
      <a class="normal" href="...">mars_AP_ML_req_01</a><br /><br />
```

<div align="right">(continues on next page)</div>

```
    <b>Test Sequences</b><br />
    <a class="normal" href="...">01: TS_AP_ML_001</a><br />
    <a class="normal" href="...">02: TS_AP_ML_002</a><br />
    ...
  </td>
  <td>Glide-slope control is armed when its Glide Slope Enable and Glide Slope␣
↪Signal are both true.</td>
  <td>
    <b>Requirement Type</b><br />functional requirement<br /><br />
    <b>Requirement Status</b><br />reviewed<br /><br />
    <b>Original Testability</b><br />Test<br /><br />
  </td>
 </tr>
 <tr>...
</table>
```

MQC iterates over each row of the table and counts the number of the same results.

- For `Requirement\Result` the content of the first column is checked, e.g. `<span class="failed">Failed</span>`.

  The following numbers are collected:

  - `Testcase\Result.Passed`

  - `Testcase\Result.Review`

  - `Testcase\Result.Failed`

  - `Testcase\Result.Error`

  - `Testcase\Result.Untriggered`

  - `Testcase\Result.Inactive`

  - `Testcase\Result.Unknown`

  - `Testcase\Result.None`

  - `Testcase\Result.Total`

- For `Requirement\Assessment Coverage`, MQC checks, if the second column contains any assessments for a requirement.

  The following numbers are collected:

  - `Requirement\Assessment Coverage.Covered` (number of requirements with linked assessments)

  - `Requirement\Assessment Coverage.Uncovered` (number of requirements not linked to any assessment)

  - `Requirement\Assessment Coverage.Total` (number of all listed requirements)

- For `Requirement\Testcase Coverage`, MQC checks, if the second column contains any test sequences for a requirement.

  The following numbers are collected:

  - `Requirement\Testcase Coverage.Covered` (number of requirements with linked testcases)

  - `Requirement\Testcase Coverage.Uncovered` (number of requirements not linked to any testcase)

  - `Requirement\Testcase Coverage.Total` (number of all listed requirements)

- For `Requirement\Testability` the content of the first column is checked, e.g. `<b>Testability</b><br /><b>Status:</b> Yes<br />`.

  The following numbers are collected:

  - `Requirement\Testability.Testable` (number of requirements with testability status `Yes`)

  - `Requirement\Testability.Untestable` (number of requirements with testability status `No`)

  - `Requirement\Testability.Total` (number of all listed requirements)

PikeTec TPT

MQC supports two types of TPT report formats:

- *XML*

- *HTML*

XML

The following TPT XML report example shows, where the TPT XML adapter extracts the expected measures from:

```
<Header ExecutionConfig="Lights Control MATLAB" ExecutionDate="14:47:58 10.05.2016"
        TptFileName="D:\requirements.tpt" TptVersion="8u2">
  <Property Name="Model Under Test" Value="D:\matlab-platform\lights_control_simulink.
↪mdl"/>
  <Property Name="System Under Test" Value="lights_control_simulink/lights_control"/>
  <Platform History="100" Name="MATLAB-Platform" Stepsize="10000" Timeout="60000000">
    <Property Name="MATLAB Version" Value="MATLAB 8.4"/>
  </Platform>
</Header>
```

MQC extracts from the `<Header.. >` element

- `ExecutionDateTime` (stored in MQC as `ReportDateTime`)

- `SystemUnderTest Value` (stored in MQC as `ArtifactName`), in this case "lights_control_simulink/lights_control"

The TPT XML adapter only stores the name of the subsystem instead of the complete path.

The `Measurement` as additional structuring element (see *Quality Computation*), is used here to distinguish between test results (same base measure and variable name) for different contexts, e.g. MiL or SiL.

The measurement name is expected as part of the file name or read from the `Name` attribute of the `<Platform>` element. In both cases, MQC expects the following syntax: `_SIL_` or `_MIL_`.

Testcases

```
...
<Summary AssessmentDuration="2.518" ExecutionDuration="2.078">
  <ExecutionSummary Errors="0" Failed="6" Inconclusive="0" Succeeded="5" Tests="11"/>
</Summary>
...
<Testcases Number="26">
  <Group Name="...">
    <Group Name="...">
      <Testcase Description="" FullName="testcase 1" Id="8296" Name="testcase 1"
→Requirements="SPEC-4 SPEC-15" Status="Stable" Version="2"/>
      <Testcase Description="" FullName="testcase 2" Id="8352" Name="testcase 2"
→Requirements="SPEC-16 SPEC-4 SPEC-9" Status="Stable" Version="2"/>
      <Testcase Description="" FullName="testcase 3" Id="8541" Name="testcase 3"
→Requirements="SPEC-13" Status="Stable" Version="2"/>
    </Group>
    <Group Name="...">
      <Group Name="...">
        <Testcase Description="" FullName="testcase 4" Id="8310" Name="testcase 4"
→Requirements="SPEC-17 SPEC-18 SPEC-9" Status="Stable" Version="2"/>
        <Testcase Description="" FullName="testcase 5" Id="8432" Name="testcase 5"
→Requirements="SPEC-17 SPEC-18 SPEC-8" Status="Stable" Version="2"/>
        ...
      </Group>
  </Group>
    ...
    </Group>
    ...
</Testcases>
```

The following data is extracted:

- testcase results are read from the categories of `<ExecutionSummary>` element

  - `Tests` as `TestCase\Result.Total`

  - `Succeeded` as `TestCase\Result.Passed`

  - `Failed` as `TestCase\Result.Failed`

  - `Errors` as `TestCase\Result.Error`

- **–** Inconclusive as `TestCase\Result.Inconclusive`
- from all `<Testcase.. >` elements MQC calculates the sums of the respective `Status` attribute values as

  - **–** `Testcase\Work Status.Stable`

  - **–** `Testcase\Work Status.Reviewed`

  - **–** `Testcase\Work Status.To Be Reviewed`

  - **–** `Testcase\Work Status.In Progress`

  - **–** `Testcase\Work Status.Modified`

  - **–** `Testcase\Work Status.New`

  - **–** `Testcase\Work Status.Total`

- additionally, MQC counts all `<Testcase.. >` elements where the `Requirements` attribute has values and stores it as

  - **–** `Testcase\Requirement Coverage.Covered` (number of testcases with requirements)

  - **–** `Testcase\Requirement Coverage.Uncovered` (number of testcases without requirements)

  - **–** `Testcase\Requirement Coverage.Total` (number of all testcases)

For each `<Testcase.. >` element, MQC reads the following data details.

- Testcase Result details:

  - **–** `Testcase` as `Subject`

  - **–** `TPT/{MeasurementName}/Testcase/{TestcaseGroupName}/{TestcaseName}` as `Subject Path`

    (whereas `{TestcaseGroupName}` is a concatenated string of all parent `Group` elements a testcase is assigned to, separated by `/`)

  - **–** `ArtifactPath` as `Artifact Inner Path`

  - **–** `Result` as `Measure`

  - **–** from the `TestcaseInformation` element with the attribute `Testcase="{id}"` extract the value of the `Result` attribute as `State`

- Testcase Requirement Coverage details:

  - **–** `Testcase` as `Subject`

  - **–** `TPT/{MeasurementName}/Testcase/{TestcaseGroupName}/{TestcaseName}` as `Subject Path`

    (whereas `{TestcaseGroupName}` is a concatenated string of all parent `Group` elements a testcase is assigned to, separated by `/`)

  - **–** `ArtifactPath` as `Artifact Inner Path`

– `Requirement Coverage` as `Measure`

– `Covered` (if any requirement is listed in the `Requirements` attribute), otherwise `Uncovered` as `State`

• Testcase Work Status details (if there is a `Status` attribute):

– `Testcase` as `Subject`

– `TPT/{MeasurementName}/Testcase/{TestcaseGroupName}/{TestcaseName}` as `Subject Path`

(whereas `{TestcaseGroupName}` is a concatenated string of all parent `Group` elements a test-case is assigned to, separated by `/`)

– `ArtifactPath` as `Artifact Inner Path`

– `Work Status` as `Measure`

– `New | In Progress | Modified | To Be Reviewed | Reviewed | Stable` as `State`

(based on the value of the `Status` attribute)

Structural Coverages

Structural coverage information is read from the `<Header><Platform><CoverageData>` elements.

```
<Header>
  <Property Name="System under Test" Value=""/>
  <Property Name="Revision" Value=""/>
  <Platform History="100" Name="C Platform" Stepsize="10000" Timeout="60000000">
    <Property Name="Platform Mapping" Value="&lt;none&gt;"/>
    <CoverageData Coverage="0.29" CoverageType="Decision" ToolName="CTC++"␣
→ToolVersion=""/>
    <CoverageData Coverage="0.16" CoverageType="Condition" ToolName="CTC++"␣
→ToolVersion=""/>
    <CoverageData Coverage="0.21" CoverageType="MC/DC" ToolName="CTC++" ToolVersion="
→"/>
  </Platform>
</Header>
```

• depending on the measurement, the coverages provided by the report are read as:

– `Model Coverage` in case of `MiL`, e.g. `Model Coverage\{CoverageType}`

– `Code Coverage` in case of `SiL`, e.g. `Code Coverage\{CoverageType}`

• the following coverage types are read by MQC:

– `Model Coverage\Condition`

– `Model Coverage\Decision`

– `Model Coverage\MC/DC`

– `Code Coverage\Condition`

> – `Code Coverage\Decision`

> – `Code Coverage\MC/DC`

- the following measures are collected per coverage type:

> – `Total` (always assumed to be 100)

> – `Covered` (value of the `Coverage` attribute multiplied by 100)

> – `Uncovered` (Total - Covered)

Requirements

The adapter prioritizes obtaining comprehensive data for requirements from the HTML report. This ensures that the adapter can access the most complete information available for requirements, enhancing the accuracy and reliability of the generated reports.

It first attempts to locate the `overview.html` file and extracts the requirement data from there (described in *Requirements*). If the `overview.html` file is not found, the adapter retrieves the data from the XML report instead.

```
<Requirements>
  <Requirement Id="SPEC-12" Text="Functional requirements"/>
  <Requirement Id="SPEC-13" State="SUCCESS" Text="If light_switch is ON, then␣
↪headlight shall immediately be ON."/>
</Requirements>
<Testcases Number="11">
  <Testcase Description="..." Requirements="SPEC-6 SPEC-13"/>
</Testcases>
<Assesslets Number="22">
  <Group Name="Assesslets">
    <Assesslet Id="1" Name="..." Requirements="SPEC-13"/>
  </Group>
</Assesslets>
...
<Summary AssessmentDuration="454.739" ExecutionDuration="3667.713">
  <ExecutionSummary Errors="0" Failed="0" Inconclusive="0" Succeeded="209" Tests="209
↪"/>
  <RequirementsSummary Errors="0" Failed="0" Inconclusive="0" Not-Covered="12" Passed=
↪"291" Requirements="303"/>
</Summary>
```

If the xml report contains a `<RequirementsSummary>` element (since TPT version 18), the requirement information is read from there. Otherwise, MQC iterates over all `<Requirements>` elements and counts the corresponding `State` attribute values.

The following data is extracted:

- `Requirement\Result.Passed`

- `Requirement\Result.Failed`

- `Requirement\Result.Error`

- `Requirement\Result.Inconclusive` (including number of requirements with `State == 'Not-Covered'`)

- `Requirement\Result.Total`

To extract the number of requirements linked to testcases, MQC counts all requirements listed as value in the `Requirements` attribute in any `<Testcase.. >` element:

- `Requirement\Testcase Coverage.Covered` (number of requirements linked to any testcase)

- `Requirement\Testcase Coverage.Uncovered` (Total - Covered)

- `Requirement\Testcase Coverage.Total`

To extract the number of requirements linked to assesslets, MQC counts all requirements listed as value in the `Requirements` attribute in any `<Assesslet.. >` element:

- `Requirement\Assesslet Coverage.Covered` (number of requirements linked to any assesslet)

- `Requirement\Assesslet Coverage.Uncovered` (Total - Covered)

- `Requirement\Assesslet Coverage.Total`

For each `<Requirement.. >` element, MQC reads the following data details.

- Requirement Result details:

  - `Requirement` as `Subject`

  - `TPT/{MeasurementName}/Requirement/{RequirementId}` as `Subject Path`

  - `ArtifactPath` as `Artifact Inner Path`

  - `Result` as `Measure`

  - the value of the `State` attribute as `State`, or `Uncovered` if the `State` attribute does not exist

- Requirement Testcase Coverage details:

  - `Requirement` as `Subject`

  - `TPT/{MeasurementName}/Requirement/{RequirementId}` as `Subject Path`

  - `ArtifactPath` as `Artifact Inner Path`

  - `Testcase Coverage` as `Measure`

  - `Covered` as `State` (if the value of the `Id` attribute is listed in the `Requirements` attribute of any `Testcase` elements, otherwise `Uncovered`)

- Requirement Assesslet Coverage details:

  - `Requirement` as `Subject`

  - `TPT/{MeasurementName}/Requirement/{RequirementId}` as `Subject Path`

  - `ArtifactPath` as `Artifact Inner Path`

  - `Assesslet Coverage` as `Measure`

- **–** `Covered` as `State` (if the value of the `Id` attribute is listed in the `Requirements` attribute of any `Assesslet` elements, otherwise `Uncovered`)

Assesslets

As for requirements, the adapter prioritizes obtaining comprehensive data for assesslets from the HTML report. It first attempts to locate the `overview.html` file and extracts the assesslet data from there (described in *Assesslets*). If the `overview.html` file is not found, the adapter retrieves the data from the XML report instead.

```xml
<Assesslets Number="22">
  <Group Name="...">
    <Assesslet Id="1" Name="..." Requirements="SPEC-13" Status="Stable" Version="2"/>
    <Assesslet Id="2" Name="..." Requirements="SPEC-6" Status="Stable" Version="2"/>
    ...
  </Group>
  <Group Name="...">
    <Group Name="...">
      <Assesslet Id="14" Name="..." Requirements="SPEC-16 SPEC-4 SPEC-9" Status=
↪"Stable" Version="2"/>
      ...
    </Group>
    <Group Name="...">
      <Assesslet Id="19" Name="..." Requirements="SPEC-17 SPEC-18 SPEC-9" Status=
↪"Stable" Version="1"/>
      <Assesslet Id="20" Name="..." Requirements="SPEC-17 SPEC-18 SPEC-8" Status=
↪"Stable" Version="2"/>
      ...
    </Group>
    ...
  </Group>
</Assesslets>
<Testresult>
  <TestcaseInformation AssessmentDuration="1.371" ExecutionDate="12:52:49 12.11.2020"␣
↪ExecutionDuration="0.1" ReportFile="...\report.html" Result="PASSED"␣
↪SimulationDuration="6.0" Testcase="184" TimeoutOccurred="false">
    <AssessletInformation AssessResult="PASSED" Assesslet="1"/>
    <AssessletInformation AssessResult="PASSED" Assesslet="2"/>
    <AssessletInformation AssessResult="INCONCLUSIVE" Assesslet="3"/>
    ...
  </TestcaseInformation>
  <TestcaseInformation AssessmentDuration="0.903" ExecutionDate="12:52:50 12.11.2020"␣
↪ExecutionDuration="0.099" ReportFile="...\report.html" Result="PASSED"␣
↪SimulationDuration="16.0" Testcase="77" TimeoutOccurred="false">
    <AssessletInformation AssessResult="PASSED" Assesslet="1"/>
    ...
  </TestcaseInformation>
  ...
</Testresult>
```

The following data is extracted:

- from all `<Assesslet..>` elements MQC calculates the sums of the respective `Status` attribute values as

  - `Assesslet\Work Status.Stable`

  - `Assesslet\Work Status.Reviewed`

  - `Assesslet\Work Status.To Be Reviewed`

  - `Assesslet\Work Status.In Progress`

  - `Assesslet\Work Status.Modified`

  - `Assesslet\Work Status.New`

  - `Assesslet\Work Status.Total`

- assesslet results are read from the `AssessletResult` attributes of the `<TestResult><TestcaseInformation><AssessletInformation>` elements

- `Assesslet\Result.Error`

- `Assesslet\Result.Failed`

- `Assesslet\Result.Inconclusive`

- `Assesslet\Result.Passed`

- `Assesslet\Result.Total`

- additionally, MQC counts all `<Assesslet.. >` elements where the `Requirements` attribute has values and stores it as

- `Assesslet\Requirement Coverage.Covered` (number of assesslets with requirements)

- `Assesslet\Requirement Coverage.Uncovered` (number of assesslets without requirements)

- `Assesslet\Requirement Coverage.Total` (number of all assesslets)

For each `<Assesslet.. >` element, MQC reads the following data details.

- Assesslet Result details:

  - `Assesslet` as `Subject`

  - `TPT/{MeasurementName}/Assesslet/{AssessletGroupName}/{AssessletName}[ID="{Id}"]` as `Subject Path`

    (whereas `{AssessletGroupName}` is a concatenated string of all parent `Group` elements an assesslet is assigned to, separated by `/`)

  - `ArtifactPath` as `Artifact Inner Path`

  - `Result` as `Measure`

  - from the `AssessletInformation` element with the attribute `Assesslet="{id}"` extract the value of the `AssessResult` attribute as `State`

- Assesslet Requirement Coverage details:

- – `Assesslet` as `Subject`

- – `TPT/{MeasurementName}/Assesslet/{AssessletGroupName}/`
  `{AssessletName}[ID="{Id}"]` as `Subject Path`

  (whereas `{AssessletGroupName}` is a concatenated string of all parent `Group` elements an
  assesslet is assigned to, separated by `/`)

- – `ArtifactPath` as `Artifact Inner Path`

- – `Requirement Coverage` as `Measure`

- – `Covered` as `State` (if any requirement is listed in the `Requirements` attribute, otherwise
  `Uncovered`)

- Assesslet Work Status details (if there is a `Status` attribute):

  - – `Assesslet` as `Subject`

  - – `TPT/{MeasurementName}/Assesslet/{AssessletGroupName}/`
    `{AssessletName}[ID="{Id}"]` as `Subject Path`

    (whereas `{AssessletGroupName}` is a concatenated string of all parent `Group` elements an
    assesslet is assigned to, separated by `/`)

  - – `ArtifactPath` as `Artifact Inner Path`

  - – `Work Status` as `Measure`

  - – `New | In Progress | Modified | To Be Reviewed | Reviewed | Stable` as
    `State`

    (based on the value of the `Status` attribute)

HTML

The TPT HTML adapter extracts the same measures as the TPT XML adapter. Data is mainly read from file
`overview.html`.

```
<body>
  <div class="header">TPT Report: Overview</div>
  <div class="table-caption">Test Summary</div>
  <table>
    <tr><td><span>TPT File</span></td><td><span>Model01.tptz</span></td></tr>
    <tr><td><span>TPT Version</span></td><td><span>15u3</span></td></tr>
    <tr><td><span>Date</span></td><td><span>07-Dec-2020</span></td></tr>
    <tr><td><span>System under Test</span></td><td><span>Model01</span></td></tr>
    ...
  </table>
  ...
  <p class="timestamp">Creation Time: 16:53:28 23.09.2021, TPT Version: 16u2</p>
</body>
```

The `ReportDateTime` is fetched from the `<p class="timestamp">` element at the end of the html body.

The `System under Test` from the first table of the report is read as artifact name. It may also be the case that `System under Test` is shorten to `SUT`. If neither the first nor the second can be found, the name of the artifact is taken from `TPT File` (file name without extension).

```
<table>
  <caption>Platform Information</caption>
  <tr><td><span>Platform Mapping</span></td><td><span>FromInterfaceImport</span></td>
↪</tr>
  <tr><td><span>MATLAB Version</span></td><td><span>MATLAB 9.3</span></td></tr>
  <tr><td><span>Variable 'TestRun'</span></td><td><span>SIL</span></td></tr>
  <tr><td><span>Variable 'iS_MIL'</span></td><td><span>0</span></td></tr>
</table>
```

The measurement name is expected as part of the file name. If it can't be extracted from there, MQC checks if the report contains a table with the caption `Platform Information` and reads the platform name from the row with the inner text `Variable 'TestRun'`.

Testcases

```
<table class="table-requirement-overview-new">
  <caption>Test Case Summary</caption>
  <tr>
    <td><span>Test Case Name</span></td>
    <td><span>Result</span></td>
    <td><span>ID</span></td>
    <td><span>Execution</span></td>
    <td><span>Assessment</span></td>
    <td><span>Duration</span></td>
    <td><span>Revision Status</span></td>
    <td><span>Directory</span></td>
    <td><span>Comment</span></td>
  </tr>
  <tr>
    <td class="generated_120" colspan="8">
      <div class="icon16x16 icon-group"></div>
      <span class="scenariogroup-new">change switch</span>
    </td>
  </tr>
  <tr>
    <td>
      <div class="icon16x16 icon-testcase"></div>
      <span class="passed">OFF ON</span>
    </td>
    <td>
      <div class="icon16x16 icon-success"></div>
    </td>
```

(continues on next page)

```html
      <td>
        <span>184</span>
      </td>
      <td>
        <span>12:52:49 12.11.2020</span>
      </td>
      <td>
        <span>12:52:49 12.11.2020</span>
      </td>
      <td>
        <span>1.471s</span>
      </td>
      <td>
        <span>Stable</span>
      </td>
      <td>
        <a href="FUSION_Platform\000_change_switch\000_OFF_ON" target="_blank">
          <span>FUSION_Platform\000_change_switch\000_OFF_ON</span>
        </a>
      </td>
      <td/>
    </tr>
    <tr>
      ...
</table>
```

Testcase data is mainly fetched from the table with the caption `Test Case Summary` contained in the file `overview.html`. The adapter iterates over all rows of the table and calculates the sum of the respective testcase results by types read from the `Result` column:

- `Testcase\Result.Error`
- `Testcase\Result.Failed`
- `Testcase\Result.Inconclusive`
- `Testcase\Result.Passed`
- `Testcase\Result.Total`

Accordingly, the testcase work status is fetched from the `Revision Status` column:

- `Testcase\Work Status.Stable`
- `Testcase\Work Status.Reviewed`
- `Testcase\Work Status.To Be Reviewed`
- `Testcase\Work Status.In Progress`
- `Testcase\Work Status.Modified`
- `Testcase\Work Status.New`

- `Testcase\Work Status.Total`

The requirement coverage, means, the number of testcases linked to requirements, is calculated by counting all testcases contained in the list of testcase results of at least one requirement (see *Requirements*):

- `Testcase\Requirement Coverage.Covered`

- `Testcase\Requirement Coverage.Uncovered`

- `Testcase\Requirement Coverage.Total`

For each row of this table with all column values, MQC reads the following data details.

- Testcase Result details:

  - `Testcase` as `Subject`

  - `TPT/{MeasurementName}/Testcase/{TestcaseGroupName}/{TestCaseName}` as `Subject Path`

    (whereas `{TestcaseGroupName}` is a concatenated string of the contents of the `Test Case Name` column of all upper rows with folder icon, separated by `/`)

  - `ArtifactPath` as `Artifact Inner Path`

  - `Result` as `Measure`

  - value of `Result` column as `State`

- Testcase Requirement Coverage details:

  - `Testcase` as `Subject`

  - `TPT/{MeasurementName}/Testcase/{TestcaseGroupName}/{TestCaseName}` as `Subject Path`

    (whereas `{TestcaseGroupName}` is a concatenated string of the contents of the `Test Case Name` column of all upper rows with folder icon, separated by `/`)

  - `ArtifactPath` as `Artifact Inner Path`

  - `Requirement Coverage` as `Measure`

  - `Covered` as `State` (if the testcase is linked to any requirement, see *Requirements*, otherwise `Uncovered`)

- Testcase Work Status details (if there is a `Revision Status` column):

  - `Testcase` as `Subject`

  - `TPT/{MeasurementName}/Testcase/{TestcaseGroupName}/{TestCaseName}` as `Subject Path`

    (whereas `{TestcaseGroupName}` is a concatenated string of the contents of the `Test Case Name` column of all upper rows with folder icon, separated by `/`)

  - `ArtifactPath` as `Artifact Inner Path`

  - `Work Status` as `Measure`

- New | In Progress | Modified | To Be Reviewed | Reviewed | Stable   as State

    (based on the value of the `Revision Status` column)

Structural Coverages

```
<table class="coverage-new">
  <caption>Coverage Information</caption>
  <tr><td><span>Type</span></td><td><span>Coverage [TER%]</span></td><td><span>Report
→</span></tr>
  <tr><td><span>Condition</span></td><td><span>16.0%</span></td><td/></tr>
  <tr><td><span>Decision</span></td><td><span>29.0%</span></td><td/></tr>
  <tr><td><span>Function</span></td><td><span>100.0%</span></td><td/></tr>
  <tr><td><span>MC/DC</span></td><td><span>21.0%</span></td><td/></tr>
  <tr><td><span>Multicondition</span></td><td><span>18.0%</span></td><td/></tr>
  <tr><td><span>Statement</span></td><td><span>29.0%</span></td><td/></tr>
</table>
```

Coverage data is fetched from the table with the caption `Coverage Information`. MQC reads the first column as measure name and the second column as measure value:

- depending on the measurement, the coverages provided by the report are read as:

    - `Model Coverage` in case of `MiL`, e.g. `Model Coverage\{Type}`

    - `Code Coverage` in case of `SiL`, e.g. `Code Coverage\{Type}`

- the following coverage types are read by MQC:

    - `Model Coverage\Condition`

    - `Model Coverage\Decision`

    - `Model Coverage\MC/DC`

    - `Code Coverage\Condition`

    - `Code Coverage\Decision`

    - `Code Coverage\MC/DC`

- the following measures are collected per coverage type:

    - `Total` (always assumed to be 100)

    - `Covered` (value of the `Coverage` column)

    - `Uncovered` (Total - Covered)

Requirements

Requirement data is mainly fetched from the table where the caption contains `Requirement Coverage Summary`.

```
<table>
  <caption>Requirement Coverage Summary (only linked requirements)</caption>
  <tr>
    <td><span>Requirement Coverage</span></td>
    <td><span>Passed</span></td>
    <td><span>Failed</span></td>
    <td><span>Execution Error</span></td>
    <td><span>Inconclusive</span></td>
    <td><span>Not Covered</span></td></tr>
  <tr>
    <td>
     <span>10 out of 13 requirements are linked to test cases</span><br/>
     <span>6 out of 13 requirements are covered</span>
    </td>
    <td><span>6</span></td>
    <td><span>0</span></td>
    <td><span>0</span></td>
    <td><span>0</span></td>
    <td><span>7</span></td>
  </tr>
</table>
```

The following data is extracted:

- `Requirement\Result.Passed`

- `Requirement\Result.Failed`

- `Requirement\Result.Error`

- `Requirement\Result.Inconclusive` (including number of requirements with state `Not Covered`)

- `Requirement\Result.Total` (sum of all result values)

To distinguish between requirements linked to test cases and those linked to assesslets, MQC additionally reads `requirement.html`. Here, MQC fetches the data from the two tables

- `Requirements Results`

  This table contains all requirements linked to test cases. Additionally it contains requirements not linked at all, which are assumed to be not testable.

- `Requirements Assesslet Results`

  This table contains all requirements linked to assesslets.

Reading both tables, MQC creates a list of requirements, each requirement containing a list of linked test-cases, assesslets or even both.

This list of requirements is used to extract the following measures:

- `Requirement\Testcase Coverage.Covered` (number of reqirements with at least one linked testcase)

- `Requirement\Testcase Coverage.Uncovered` (total - covered)

- `Requirement\Testcase Coverage.Total` (number of all testable requirements)

- `Requirement\Assesslet Coverage.Covered` (number of reqirements with at least one linked assesslet)

- `Requirement\Assesslet Coverage.Uncovered` (total - covered)

- `Requirement\Assesslet Coverage.Total` (number of all testable requirements)

For each requirement from these two tables, MQC reads the following data details.

- Requirement Result details (if the `State` column is `Passed`, `Failed`, `Error` or `Inconclusive`):

  - `Requirement` as `Subject`

  - `TPT/{MeasurementName}/Requirement/{RequirementID}` as `Subject Path`

  - `ArtifactPath` as `Artifact Inner Path`

  - `Result` as `Measure`

  - the value of the `State` column as `State`

- Requirement Testcase Coverage details:

  - `Requirement` as `Subject`

  - `TPT/{MeasurementName}/Requirement/{RequirementID}` as `Subject Path`

  - `ArtifactPath` as `Artifact Inner Path`

  - `Testcase Coverage` as `Measure`

  - `Covered` as `State` (if any test case is listed in the `Test Case Results` column of the `Requirements Results` table, otherwise `Uncovered`)

- Requirement Assesslet Coverage details:

  - `Requirement` as `Subject`

  - `TPT/{MeasurementName}/Requirement/{RequirementID}` as `Subject Path`

  - `ArtifactPath` as `Artifact Inner Path`

  - `Assesslet Coverage` as `Measure`

  - `Covered` as `State` (if any assesslet is listed in the `Assesslet Results` column of the `Requirements Assesslet Results` table, otherwise `Uncovered`)

Assesslets

```
<table class="table-requirement-overview-new">
  <caption>Assesslet Summary</caption>
  <tr>
    <td/>
    <td><span>Assesslet Name</span></td>
    <td><span>Revision Status</span></td>
```

(continues on next page)

```
    <td><span>Passed</span></td>
    <td><span>Failed</span></td>
    <td><span>Execution Error</span></td>
    <td><span>Inconclusive</span></td>
    <td><span>Total</span></td>
  </tr>
  <tr>
    <td class="image-cell">
      <div class="icon16x16 icon-success"></div>
    </td>
    <td class="passed">
      <span>headlight hold if switch AUTO hysterese bright [ID=7]</span>
    </td>
    <td>
      <span>Stable</span>
    </td>
    <td class="number-cell">
      <span>4</span>
    </td>
    <td class="number-cell"/>
    <td class="number-cell"/>
    <td class="number-cell">
      <span>7</span>
    </td>
    <td class="number-cell">
      <span>11</span>
    </td>
  </tr>
  ...
</table>
```

Assesslet data is mainly fetched from the table with the caption `Assesslet Summary` contained in the file `overview.html`. The adapter iterates over all rows of the table and calculates the sum of the respective assesslet results by taking the worst result per row (if the assesslet has been tested for multiple testcases):

- `Assesslet\Result.Error`

- `Assesslet\Result.Failed`

- `Assesslet\Result.Inconclusive`

- `Assesslet\Result.Passed`

- `Assesslet\Result.Total`

Accordingly, the assesslet work status is fetched from the `Revision Status` column:

- `Assesslet\Work Status.Stable`

- `Assesslet\Work Status.Reviewed`

- `Assesslet\Work Status.To Be Reviewed`

- `Assesslet\Work Status.In Progress`

- `Assesslet\Work Status.Modified`

- `Assesslet\Work Status.New`

- `Assesslet\Work Status.Total`

The requirement coverage, means, the number of assesslets linked to requirements, is calculated by count-
ing all assesslets contained in the list of assesslet results of at least one requirement (see *Requirements*):

- `Assesslet\Requirement Coverage.Covered`

- `Assesslet\Requirement Coverage.Uncovered`

- `Assesslet\Requirement Coverage.Total`

For each row of this table with all column values, MQC reads the following data details.

- Assesslet Result details:

    - `Assesslet` as `Subject`

    - `TPT/{MeasurementName}/Assesslet/{AssessletName}` as `Subject Path`

    - `ArtifactPath` as `Artifact Inner Path`

    - `Result` as `Measure`

    - value of first column as `State`

- Assesslet Requirement Coverage details:

    - `Assesslet` as `Subject`

    - `TPT/{MeasurementName}/Assesslet/{AssessletName}` as `Subject Path`

    - `ArtifactPath` as `Artifact Inner Path`

    - `Requirement Coverage` as `Measure`

    - `Covered` as `State` (if the assesslet is linked to any requirement, see *Requirements*), otherwise
      `Uncovered`)

- Assesslet Work Status details (if there is a `Revision Status` column):

    - `Assesslet` as `Subject`

    - `TPT/{MeasurementName}/Assesslet/{AssessletName}` as `Subject Path`

    - `ArtifactPath` as `Artifact Inner Path`

    - `Work Status` as `Measure`

    - `New | In Progress | Modified | To Be Reviewed | Reviewed | Stable` as
      `State`

      (based on the value of the `Revision Status` column)

Razorcat Tessy

From the extract of the Tessy Example XML report,

```
<report success="notok" tessy_version="4.0.15" xml_version="3">
    <info date="2018-08-23" time="16:20:30+0200"/>
    ...
    <tessyobject id="1024" level="0" name="Testsuite" success="notok" type="project">
        <testcase_statistics notexecuted="0" notok="11" ok="54" success="notok" total=
→"65"/>
        <coverage>
            <c0 notreached="0" percentage="100" reached="142" success="ok" total="142"/
→>
            <c1 notreached="0" percentage="100" reached="113" success="ok" total="113"/
→>
            <mcdc notreached="0" percentage="100" reached="117" success="ok" total="117
→"/>
        </coverage>
        ...
    </tessyobject>
</report>
```

MQC reads

- from the main (`report`) header:

    - `tessy_version` (stored in MQC as `TessyReportVersion`)

- from the `<info..>` header:

    - `date` and `time` (stored in MQC as `ReportDateTime`)

- from the `<tessyobject..>` header:

    - `name` (stored in MQC as `ArtifactName`)

    - `type` (stored in MQC as `TessyObjectType`)

- from the `<testcase_statistics..>` header:

    - `ok` as `Testcase\Result.Passed`

    - `notok` as `Testcase\Result.Failed`

    - `notexecuted` as `Testcase\Result.Unexecuted`

    - sum of these three value as `Testcase\Result.Total`

- from the `<coverage>` `<c0..>` header:

    - `reached` as `Code Coverage\Statement.Covered`

    - `notreached` as `Code Coverage\Statement.Uncovered`

    - `total` as `Code Coverage\Statement.Total`

- from the `<coverage>` `<c1..>` header:

- **–** `reached` as `Code Coverage\Branch.Covered`

- **–** `notreached` as `Code Coverage\Branch.Uncovered`

- **–** `total` as `Code Coverage\Branch.Total`

- from the `<coverage> <mcdc..>` header:

  - **–** `reached` as `Code Coverage\MC/DC.Covered`

  - **–** `notreached` as `Code Coverage\MC/DC.Uncovered`

  - **–** `total` as `Code Coverage\MC/DC.Total`

MathWorks Polyspace

MQC supports two types of Polyspace report formats:

- *XML*

- *Text*

XML

If a Polyspace Xml report is created, MQC reads out from the xml-file:

- `ReportDateTime`: from element `PubDateTime`

- `ArtifactPath`: from element `Subtitle`

- `ArtifactName`: same as `ArtifactPath`

The xml-file refers to several xfrag-files in the `Polyspace-doc` directory to be found on the same level as the xml-file.

From the `image-000-chapter.xfrag`-file, MQC extracts

- `BaseMeasureName`: from the `title` elements of the tables

- `VariableName`: first `entry` element of each table body row

- `MeasureValue`: second `entry` element of each table body row

```
<table>
  <title>Coding Rules Summary - MISRA-C Checker</title>
  <tgroup>
    <tbody>
      <row><entry>Violations</entry><entry>52</entry></row>
      <row><entry>Pass/Fail</entry><entry>-</entry></row>
    </tbody>
  </tgroup>
</table>
<table>
  <title>Run-Time Checks Summary</title>
```

```
  <tgroup>
    <tbody>
      <row><entry>Number of Red Checks</entry><entry><emphasis role="red">2</emphasis>
→</entry></row>
      <row><entry>Number of Gray Checks</entry><entry><emphasis role="gray">10</
→emphasis></entry></row>
      <row><entry>Number of Orange Checks</entry><entry><emphasis role="orange">13</
→emphasis></entry></row>
      <row><entry>Number of Green Checks</entry><entry><emphasis role="green">205</
→emphasis></entry></row>
      <row><entry>Proven</entry><entry>100.0%</entry></row>
      <row><entry>Pass/Fail</entry><entry>-</entry></row>
    </tbody>
  </tgroup>
</table>
<table>
  <title>Global Variable Summary</title>
  <tgroup>
    <tbody>
      <row><entry>Used non-shared variable</entry><entry>51</entry></row>
    </tbody>
  </tgroup>
</table>
```

The extracted data is then modified and transformed as follows:

- If title = `Run-Time Checks Summary`

    - `Number of Red Checks` stored as `Run-Time Checks.Major`

    - `Number of Gray Checks` stored as `Run-Time Checks.Minor`

    - `Number of Orange Checks` stored as `Run-Time Checks.Moderate`

    - `Number of Green Checks` stored as `Run-Time Checks.Good`

    - `Percentage of Proven` stored as `Run-Time Checks.Proven`

    - `Pass/Fail` stored as `Run-Time Checks.Pass_Fail`

- If title = `Coding Rules Summary - MISRA-C Checker`

    - `Violations` stored as `MISRA-C Checker.Violations`

- If title = `Global Variable Summary`

    - `Used non-shared variable` stored as `Global Variable.Used non-shared variable`

    - `Unused variable` stored as `Global Variable.Unused variable`

Text

If a Polyspace tab-separated Text report is created, MQC reads from this txt-file:

- `ReportDateTime`: time stamp of the last modification of the txt-file
- `ArtifactPath`: extracted from `File` column (see below)
- `ArtifactName`: same as `ArtifactPath`
- `BaseMeasureName`: extracted from `Family` column
- `VariableName`: extracted from `Color`, `Information`, "Check" and `Comment` column based on the BaseMeasureName
- `MeasureValue`: aggregated count per `Color`

To get the `ArtifactPath`, MQC extracts the file paths of all files used to create the report from the `File` column. MQC then obtains the common prefix from these file paths and takes the last directory from that prefix. This is stored as `ArtifactPath`.

For example if paths read from the "File" column are as follows:

```
C:\dev\Models\GlobalPosition\TLProj\TL_GlobalPosition\GlobalPosition.h
C:\dev\Models\GlobalPosition\TLSim\Rte_GlobalPosition.h
C:\dev\Models\GlobalPosition\TLSim\TL_GlobalPosition_fri.h
```

In this example the common prefix is `C:\dev\Models\GlobalPosition` and MQC extracts the last common directory `GlobalPosition` as the artifact path.

After reading, the extracted data is then modified and transformed as follows:

- If `Family` equals `Run-time Check`:
    - `Color` = `Red` is stored as `Run-Time Checks.Major`
    - `Color` = `Red` and `Comment` is not empty is stored as `Run-Time Checks.Major with Comments`
    - `Color` = `Gray` is stored as `Run-Time Checks.Minor`
    - `Color` = `Gray` and `Comment` is not empty is stored as `Run-Time Checks.Minor`
    - `Color` = `Orange` is stored as `Run-Time Checks.Moderate`
    - `Color` = `Orange` and `Comment` is not empty is stored as `Run-Time Checks.Moderate`
    - `Color` = `Green` is stored as `Run-Time Checks.Good`
- If `Family` equals `Global Variable`:
    - `Check` = `Unused variable` is stored as `Global Variable.Unused variable`
    - `Color` = `Used non-shared variable` is stored as `Global Variable.Used non-shared variable`
- `Family` starts with `MISRA C`:

- Information = `Category:Mandatory` is stored as `MISRA-C Checker.Mandatory`

- Information = `Category:Mandatory` and `Comment` is not empty is stored as `MISRA-C Checker.Mandatory with Comments`

- Information = `Category:Required` is stored as `MISRA-C Checker.Required`

- Information = `Category:Required` and `Comment` is not empty is stored as `MISRA-C Checker.Required with Comments`

- Information = `Category:Advisory` is stored as `MISRA-C Checker.Advisory`

- Information = `Category:Advisory` and `Comment` is not empty is stored as `MISRA-C Checker.Advisory with Comments`

- Sum of `Category:Mandatory`, `Category:Required` and `Category:Advisory` is stored as `MISRA-C Checker.Violations`

All the base measures with the post fix `with Comments` are always a subset of the root measure. For example, `Run-Time Checks.Major` is the count of all run-time checks which are "red", with or without comments. Thus, `Run-Time Checks.Major with Comments` is always less or equal to the value of `Run-time Check.Major`.

BTC EmbeddedTester

The BTC EmbeddedTester adapter reads the data from HTML report files of this dynamic test tool.

MQC reads out:

```html
<h4>Report Base</h4>
<table class="table dualTable" id="report_base_table">
    <tr>
        <th>Property</th>
        <th>Value</th>
    </tr>
    <tr>
        <td>Scope</td>
        <td>EV3Control_main</td>
    </tr>
    <tr>
        <td>Path</td>
        <td>EV3Control_demo_ec/EV3Control</td>
    </tr>
    ...
</table>
```

- From the `Report Base` table:

    - The value of `Scope` property stored in MQC as `ArtifactName`

    - The value of `Path` property stored in MQC as `ArtifactPath`

```
<h4>General</h4>
<table class="table dualTable">
  <tr>
      <th>Property</th>
      <th>Value</th>
  </tr>
  ...
  <tr>
      <td>Profile Creation Date</td>
      <td>Jan 30, 2024, 11:56:47 AM</td>
  </tr>
  ...
</table>
```

- From the `General` table:

    - The value of `Profile Creation Date` stored in MQC as `ReportDateTime`

```
<h4>Test Execution Results Overview</h4>
<table class="table">
    <tr>
      <th></th>
      <th>TestCase</th>
      <th>Length</th>
      <th>Requirement (report base)</th>
      <th>Scope</th>
      <th>Folder</th>
      <th>Result (TL MIL)</th>
      <th>Result (SIL)</th>
    </tr>
    <tr>
      <td style="text-align:center"><span class="icon-sdc" title="No problems during␣
↪simulation"></span></td>
      <td>TC-REQ_SC_1_2</td>
      <td>5</td>
      <td><a class="link"> REQ_SC_1_2</a></td>
      <td>SDOCAR</td>
      <td>Default Test Cases</td>
      <td><div class="ok-glyph ok-color">Passed</div></td>
      <td><div class="ok-glyph ok-color">Passed</div></td>
    </tr>
    <tr>
      <td style="text-align:center"><span class="icon-sdc" title="No problems during␣
↪simulation"></span></td>
      <td><a class="link">TC-REQ_SC_2_1</a></td>
      <td>11</td>
      <td><a class="link">REQ_SC_2_1</a></td>
      <td>SDOCAR</td>
```

```
      <td>Default Test Cases</td>
      <td><div class="error-glyph error-color">Failed</div></td>
      <td><div class="error-glyph error-color">Failed</div></td>
    </tr>
</table>
```

- In the `Test Execution Results Overview` table, distinct columns are allocated for test case results corresponding to various measurements.

  MQC generates these measures for each measurement based on the outcomes of the respective test cases:

  – Number of `Passed` test cases as `Testcase\Result.Passed`

  – Number of `Failed` test cases as `Testcase\Result.Failed`

  – Number of `Error` test cases as `Testcase\Result.Error`

  – Number of `No Verdict` test cases as `Testcase\Result.No Verdict`

  – Number of test cases as `Testcase\Result.Total`

  – The count of test cases containing item(s) in the `Requirement` column as `Testcase\Requirement Coverage.Covered`

  – The count of test cases with no item listed in the `Requirement` column as `Testcase\Requirement Coverage.Uncovered`

  – The count of test cases as `Testcase\Requirement Coverage.Total`

  For each row and measurement, MQC reads two data details.

  For the first Data Detail reads:

  – `Testcase` as `Subject`

  – `EmbeddedTester/{MeasurementName}/Testcase/{Folder}/{TestCase}` as `Subject Path`

  – `ArtifactPath` as `Artifact Inner Path`

  – `Result` as `Measure`

  – `Passed | Failed | Error | No Verdict` (based on the value in the measurement column) as `State`

  For the second Data Detail reads:

  – `Testcase` as `Subject`

  – `EmbeddedTester/{MeasurementName}/Testcase/{Folder}/{TestCase}` as `Subject Path`

  – `ArtifactPath` as `Artifact Inner Path`

  – `Requirement Coverage` as `Measure`

- – `Covered` (if any requirement is listed in the Requirement column), otherwise `Uncovered` as State

```
<h3>2. Requirements Traceability</h3>
<table class="table">
    <tr>
        <th>Requirement ID</th>
        <th>Description</th>
        <th>Test Cases</th>
        <th>Covered</th>
        <th >Passed (TL MIL)</th>
        <th >Passed (SIL)</th>
    </tr>
    <tr id="2yR_req">
        <td>REQ_SC_1_1</td>
        <td>We ramp down the ultrasound sensor signal value.</td>
        <td>1</td>
        <td><div class="ok-glyph ok-color">Covered</div></td>
        <td>100%</td>
        <td> 100%</td>
    </tr>
    <tr id="2yV_req">
        <td>REQ_SC_2_1</td>
        <td>IStart with a zero input signal for usSensorFront. Drive revers by setting␣
→remoteCtrlCmd.</td>
        <td>1</td>
        <td><div class="ok-glyph ok-color">Covered</div></td>
        <td>0%</td>
        <td>0%</td>
    </tr>
</table>
```

- In the `Requirements Traceability` table, there are distinct columns indicating the status of requirements passed for different measurements. The status is represented as *0%* if it failed and *100%* if it passed. MQC generates these measures for each measurement based on the results and coverage of the requirements:

  - – Number of `100%` requirements as `Requirement\Result.Passed`

  - – Number of `0%` requirements as `Requirement\Result.Failed`

  - – Number of `100%` or `0%` requirements as `Requirement\Result.Total`

  - – Number of requirements marked as `Covered` in the `Covered` column as `Requirement\Testcase Coverage.Covered`

  - – Number of requirements marked as `Not covered` in the `Covered` column as `Requirement\Testcase Coverage.Uncovered`

  - – Number of requirements as `Requirement\Testcase Coverage.Total`

  For each row and measurement, MQC reads two data details.

For the first Data Detail reads :

- `Requirement` as `Subject`

- `EmbeddedTester/{MeasurementName}/Requirement/{Requirement ID}` as `Subject Path`

- `ArtifactPath` as `Artifact Inner Path`

- `Result` as `Measure`

- `Passed` (if the `Passed` column is `100%`) or `Failed` if it is `0%` as `State`

For the second Data Detail reads:

- `Requirement` as `Subject`

- `EmbeddedTester/{MeasurementName}/Requirement/{Requirement ID}` as `Subject Path`

- `ArtifactPath` as `Artifact Inner Path`

- `Testcase Coverage` as `Measure`

- `Covered` (if the `Covered` column is `Covered`), otherwise `Uncovered` as `State`

```
<table>
  <tr>
    <th></th>
    <th>Properties</th>
    <th colspan="2">Handled</th>
    <th colspan="2">Covered</th>
    <th colspan="2">Unreachable (n/inf)</th>
    <th colspan="2">Justified</th>
  </tr>
  <tr>
    <td>Statement (STM)</td>
    <td>302</td>
    <td>90</td>
    <td>29.8%</td>
    <td>90</td>
    <td>29.8%</td>
    <td>0</td>
    <td>0.0%</td>
    <td>0</td>
    <td>0.0%</td>
  </tr>
</table>
```

- In all tables under the `Code Analysis` section, the SiL (Software-in-Loop) coverage data is retrieved. The first column is interpreted as the measure name, while the subsequent columns are interpreted as variables:

    - `Handled`

    – Covered

    – Unreachable

    – Justified

```html
<h2>Back-to-Back Testing</h2>
<div><h3>1. Back-to-Back Test</h3>
  <h3>Test Results Overview</h3>
  <h3 >TL MIL vs SIL</h3>
  <div>
     <h4>Comparison Results</h4>
     <table class="table">
         <tr>
           <th></th>
           <th>Name</th>
           <th>Length</th>
           <th>Result</th>
         </tr>
         <tr>
           <td>#1</td>
           <td>
               <span class="icon-wdc" title="Warnings during comparison"></span>:
               TL MIL_Stage1_TempValue <br>
               <span class="icon-sdc" title="No problems during comparison"></span>:
               SIL_Stage1_TempValue
           </td>
           <td> 1</td>
           <td><div class="ok-glyph ok-color"> Passed</div> </td>
         </tr>
     </table>
  </div>
</div>
```

- In the `Back-to-Back Testing` table, MQC interprets these measures for the `B2B` measurement based on the values in the 'Result' column:

  – Number of `Passed` test cases as `Testcase\Result.Passed`

  – Number of `Failed` test cases as `Testcase\Result.Failed`

  – Number of test cases as `Testcase\Result.Total`

  For each row MQC reads data detail:

  – `Testcase` as `Subject`

  – `EmbeddedTester/B2B/Testcase/{Name}` as `Subject Path`

  – `ArtifactPath` as `Artifact Inner Path`

  – `Result` as `Measure`

– `Passed | Failed | Error | No Verdict` (based on the value in the `Result` column) as State

Verifysoft Testwell CTC++

MQC supports two types of Verifysoft Testwell CTC++ report formats:

- *XML*

- *HTML*

XML

MQC will read from the XML file:

```xml
<ctc_xml_report>
  <header_info>
    <ctcpost_version>8.0.1</ctcpost_version>
    <copyright>Copyright (c) 1993-2013 Testwell Oy</copyright>
    <copyright>Copyright (c) 2013-2016 Verifysoft Technology GmbH</copyright>
    <report_generated>Fri Jun 05 15:22:09 2020</report_generated>
  </header_info>
  <file name="D:\source\Application\DeviceLayer\test_object.cpp">
    <file_type>source</file_type>
    <instrumentation_mode>multicondition</instrumentation_mode>
    <instrumentation_timestamp>Fri Jun 05 15:21:42 2020</instrumentation_timestamp>
    <sym_rewrite_count>0</sym_rewrite_count>
    <sym_update_count>0</sym_update_count>
    <data_rewrite_count>0</data_rewrite_count>
    <data_update_count>0</data_update_count>
    <file_summary>
      <functions>77</functions>
      <lines>200</lines>
      <measurement_points>16</measurement_points>
      <ter>68</ter>
      <hits>343</hits>
      <all>505</all>
      <statement_ter>92</statement_ter>
      <statement_hits>5310</statement_hits>
      <statement_all>5752</statement_all>
      <statement_na_functions>0</statement_na_functions>
    </file_summary>
  </file>
</ctc_xml_report>
```

MQC extracts the following information, stores and transforms it to the MQC data structure:

- from the `<header_info>` header:

- **–** `<report_generated>` (stored in MQC as `ReportDateTime`)
- from each `<file>` header:
    - **–** `name` (stored in MQC as `ArtifactName`)
    - **–** from the `<file_summary>` header:
        - `*` `<lines>` (stored in MQC as `Code Coverage\{Source Line}.Total`)
        - `*` `<measurement_points>` (stored in MQC as `Code Coverage\{Measurement Point}.Total`)
        - `*` when both `<xxx_hits>` and `<xxx_all>` exist
            - · `<xxx_hits>` (stored in MQC as `Code Coverage\{xxx}.Covered`)
            - · `<xxx_all>` (stored in MQC as `Code Coverage\{xxx}.Total`)
            - · `<xxx_all>`-`<xxx_hits>` (stored in MQC as `Code Coverage\{xxx}.Uncovered`)
            - · The capitalised form of `xxx` is used in MQC.

HTML

MQC will read from the HTML file:

```html
<html>
  <head>
    <title>CTC++ Coverage Report - Files Summary</title>
  </head>
  <body>
  <table>
    <tr><td class="info">Symbol file(s)</td><td class="info">:</td><td class="info">
     c:\temp\Autopilot_ec\Test\Test_autopilot_demo_ec\Autopilot_Mode_
→Logic\Test001\TSeq001\CTCCov\Data_sil_ec\
     MON.sym (Thu May 21 05:35:10 2020)</td></tr>
    <tr><td class="info">Listing produced at</td><td class="info">:</td><td class=
→"info">
     Thu May 21 05:44:18 2020</td></tr>
    <tr><td class="info">Coverage view</td><td class="info">:</td><td class="info">
     Reduced to decision coverage</td></tr>
    <tr><td class="info">Input listing</td><td class="info">:</td><td class="info">
     D:\temp\Autopilot_ec\Test\Test_autopilot_demo_ec\Autopilot_Mode_
→Logic\CTCCov\Data_sil_ec\profile.txt</td></tr>
    <tr><td class="info">HTML generated at</td><td class="info">:</td><td class="info
→">
     Thu May 21 05:44:18 2020</td></tr>
    <tr><td class="info">Structural threshold</td><td class="info">:</td><td class=
→"infob">100 %</td></tr>
    <tr><td class="info">Statement threshold</td><td class="info">:</td><td class=
→"infob">100 %</td></tr>
```

(continues on next page)

```
</table><br>
<table>
   <thead>
      <tr><th>TER %</th><th>-</th><th colspan="2">decision</th><th>TER %</th><th>-</
→th>
         <th colspan="2">statement</th><th>File</th></tr>
   </thead>
   <tbody>
      <tr><td class="dirb" colspan="9"><a name="a1"></a>Directory: C:\temp\Autopilot_
→ec\Test\
         Test_autopilot_demo_ec\Autopilot_Mode_Logic\Autopilot_Mode_Logic_sil_sil_ec_
→ert_rtw</td></tr>
      <tr><td class="below">98 %</td><td class="below">-</td><td class="below">(41/42)
→</td>
         <td width="115"><img border="1" src="red.gif" width="98" height="5" alt="">
          <img border="1" src="white.gif" width="2" height="5" alt="">
         </td><td class="below">99 %</td><td class="below">-</td><td class="below">(86/
→87)</td>
         <td width="115"><img border="1" src="red.gif" width="99" height="5" alt="">
          <img border="1" src="white.gif" width="1" height="5" alt="">
         </td><td><a href="indexD1.html" class="underline">Autopilot_Mode_Logic_sil_sil_
→ec.c</a></td></tr>
      <tr><td class="above">100 %</td><td class="above"></td><td class="above">(0/0)</
→td>
         <td width="115"><img border="1" src="blue.gif" width="100" height="5" alt=""></
→td>
         <td class="above">100 %</td><td class="above"></td><td class="above">(0/0)</td>
         <td width="115"><img border="1" src="blue.gif" width="100" height="5" alt=""></
→td>
         <td><a href="indexD2.html" class="underline">Autopilot_Mode_Logic_sil_sil_ec_
→data.c</a></td></tr>
      <tr><td class="belowb">98 %</td><td class="below">-</td><td class="below">(41/
→42)</td>
         <td width="115"><img border="1" src="red.gif" width="98" height="5" alt="">
          <img border="1" src="white.gif" width="2" height="5" alt=""></td>
         <td class="belowb">99 %</td><td class="below">-</td><td class="below">(86/87)</
→td>
         <td width="115"><img border="1" src="red.gif" width="99" height="5" alt="">
          <img border="1" src="white.gif" width="1" height="5" alt=""></td>
         <td class="dirb"><a title="C:\temp\R2018B\Autopilot_ec\Test\Test_autopilot_
→demo_ec\Autopilot_Mode_Logic\
         Autopilot_Mode_Logic_sil_sil_ec_ert_rtw">DIRECTORY OVERALL</a></td></tr>
      <tr><td class="ruler" colspan="9"> </td></tr>
      <tr><td class="belowb">98 %</td><td class="below">-</td><td class="below">(41/
→42)</td><td width="115">
         <img border="1" src="red.gif" width="98" height="5" alt="">
         <img border="1" src="white.gif" width="2" height="5" alt=""></td>
```

```
      <td class="belowb">99 %</td><td class="below">-</td><td class="below">(86/87)</
→td>
      <td width="115"><img border="1" src="red.gif" width="99" height="5" alt="">
       <img border="1" src="white.gif" width="1" height="5" alt=""></td>
      <td><a href="index0.html" class="underlineb">OVERALL</a></td></tr>
    </tbody>
  </table><br>
 </body>
</html>
```

MQC extracts the following information, stores and transforms it to the MQC data structure:

- from the first `<table>` header:

  - read the last `<td>` of the `<tr>` with the first `<td>` equal to *'Listing produced at'* (stored in MQC as `ReportDateTime`)

- from the second `<table>` header:

  - from `<thead>` start from third `<th>` and every three `<th>` is capitalised and add to the `Code Coverage\` as `MeasureName`.

  - from each `<tr>`:

    * read last `<td>` as `<ArtifactName>`

    * from `<td>` correspondent to `<th>` read values for that measure in such a way that the first number is fetch as `Covered`, second as `Total` and the difference between these two as `Uncovered` variable . (e.g for second row in sample report above, *(41/42)* is read as Code Coverage\Decision.Covered = 41 , Code Coverage\Decision.Total = 42 and Code Coverage\Decision.Uncovered = 1)

In reading rows from second table we will ignore `<tr>` with just one `<td>` or has *class="dirb"* or *class="ruler"* or when last `<td>` is equal to *"DIRECTORY OVERALL"* or *"OVERALL"*.

Danlawinc MxSuite

MQC reads data from the XML file named *Report.RegResults.xml*:

```
<MxVDevReportFile>
  <Report>
    <Date>20200423</Date>
    <Time>15:39</Time>
    <ProgramVersion>3.41.1.45984</ProgramVersion>
  </Report>
  <Regression>
    <Overview>
      <Project>
        <Name>ModelTestMiL</Name>
        <Description>
```

```
      </Description>
      <Folder>C:\ModelTest\ModelTestMiL</Folder>
      <FileName>ModelTestMiL.mxp</FileName>
      <ScenarioFolder>.\TestCases\</ScenarioFolder>
    </Project>
  </Overview>
  <StatisticsTotals Total="1" PercentTotal="100" Passed="0" PercentPassed="0"␣
↪Failed="1" PercentFailed="100"
                  Skipped="0" PercentSkipped="0" RunTimeError="0"␣
↪PercentRunTimeError="0"
                  Missing="0" PercentMissing="0">
    <Testcases Total="1" PercentTotal="100" Passed="0" PercentPassed="0" Failed="1"␣
↪PercentFailed="100"
             Skipped="0" PercentSkipped="0" RunTimeError="0" PercentRunTimeError=
↪"0"
             Missing="0" PercentMissing="0" />
  </StatisticsTotals>
 </Regression>
</MxVDevReportFile>
```

MQC reads out

- from the `<Report>`

  - `<Date>` and `<Time>` (stored in MQC as `ReportDateTime`)

- from each `<Regression>` header:

  - `<Overview><Project><Name>` (stored in MQC as `ArtifactName`)

  - from `<StatisticsTotals>` header:

    * `Total` (stored in MQC as `Scenarios\Result.Total`)

    * `Passed` (stored in MQC as `Scenarios\Result.Passed`)

    * `Failed` (stored in MQC as `Scenarios\Result.Failed`)

    * `Skipped` (stored in MQC as `Scenarios\Result.Unexecuted`)

    * `RunTimeError` (stored in MQC as `Scenarios\Result.Error`)

    * `Missing` (stored in MQC as `Scenarios.Missing`)

  - from `<Testcases>` header:

    * `Total` (stored in MQC as `Testcases\Result.Total`)

    * `Passed` (stored in MQC as `Testcases\Result.Passed`)

    * `Failed` (stored in MQC as `Testcases\Result.Failed`)

    * `Skipped` (stored in MQC as `Testcases\Result.Unexecuted`)

    * `RunTimeError` (stored in MQC as `Testcases\Result.Error`)

* `Missing` (stored in MQC as `Testcases\Result.Missing`)

MathWorks Simulink Check

MQC reads from the HTML file:

```
<div class="ReportContent" id="Model02">
  <table class="AdvTableNoBorder" width="100%" border="0">
    <tr>
      <td align="left" valign="top"><b>System: <font color="#800000">Model02</font></
→b></td>
      <td align="right" valign="top"><b>Current run: <font color="#800000">22-Oct-
→2020 18:13:17</font></b></td>
    </tr>
  </table>
  <b>Run Summary</b></font><br/>
  <table class="AdvTableNoBorder" width="60%" border="0">
    <tr>
      <th align="left" valign="top"><b>Pass</b></th>
      <th align="left" valign="top"><b>Fail</b></th>
      <th align="left" valign="top"><b>Warning</b></th>
      <th align="left" valign="top"><b>Not Run</b></th>
      <th align="left" valign="top"><b>Total</b></th>
    </tr>
    <tr>
      <td align="left" valign="top" >  <img src="task_passed.png" /> 38</td>
      <td align="left" valign="top" >  <img src="task_failed.png" /> 8</td>
      <td align="left" valign="top" >  <img src="task_warning.png" /> 7</td>
      <td align="left" valign="top" >  <img src="icon_task.png" /> 363</td>
      <td align="left" valign="top" >416</td>
    </tr>
  </table>
</div>
```

MQC extracts the following information, stores and transforms it to the MQC data structure:

- the `ArtifactName` from `System:`

- the `ReportDateTime` from `Current Run:`

- the complete `Run Summary` from the second table inside the `<div class="ReportContent">` tag:

  - `Pass` as `GuidelineCount.Passed`

  - `Fail` as `GuidelineCount.Failed`

  - `Warning` as `GuidelineCount.Warning`

  - `Not Run` as `GuidelineCount.Not Run`

  - `Total` as `GuidelineCount.Total`

Simulink Design Verifier

```
<div class="titlepage">
  <div>
    <div><h1 class="title"><a name="d0e1"></a>Simulink Design Verifier Report</h1></
↪div>
    <div><h2 class="subtitle">Model03</h2></div>
    <div><div class="author"><h3 class="author"><span class="firstname">Author</span>
↪</h3></div></div>
    <div><p class="pubdate">27-Oct-2020 12:03:54</p></div>
  </div>
</div>
<div class="chapter" title="Chapter 1. Summary">
  <div class="titlepage">
    <div><h2 class="title"><a name="d0e13"></a>Chapter 1. Summary</h2></div>
  </div>
  <p title="Analysis Information"><b>Analysis Information </b><a name="d0e22"></
↪a></p>
  <div class="table">
    <div class="table-contents">
      <table summary="" border="0" cellspacing="0" fastRender="1">
        <tbody>
          <tr><td align="left">Model:</td><td align="left">Model03</td></tr>
          <tr><td align="left">Mode:</td><td align="left">Design error detection</td>
↪</tr>
          <tr><td align="left">Status:</td><td align="left">Completed normally</td></
↪tr>
          <tr><td align="left">Analysis Time:</td><td align="left">66s</td></tr>
        </tbody>
      </table>
    </div>
  </div>
  <p title="Objectives Status"><b>Objectives Status </b><a name="d0e51"></a></p>
  <div class="table">
    <div class="table-contents">
      <table summary="" border="0" cellspacing="0" fastRender="1">
        <thead>
          <tr><th align="left">Number of Objectives:</th><th align="left">5</th></tr>
        </thead>
        <tbody>
          <tr><td align="left">Objectives Proven Valid: </td><td align="left">4</td></
↪tr>
          <tr><td align="left">Objectives Falsified with Test Cases: </td><td align=
↪"left">1</td></tr>
        </tbody>
      </table>
    </div>
  </div>
```

```
</div>
```

MQC extracts the following information, stores and transforms it to the MQC data structure:

- the `ArtifactName` from the `<h2 class="subtitle">` tag

- the `ReportDateTime` from the `<p class="pubdate">` tag

- the `MeasurementName` from the `Analysis Information` section of the "Summary" chapter

- from the `Objectives Status` section of the "Summary" chapter:

  - `Number of Objectives` as `Objectives.Total`

  - `Objectives Proven Valid` as `Objectives.Passed`

  - `Objectives Falsified with Test Cases` as `Objectives.Failed`

Perforce Helix QAC

MQC supports two types of QA-System QAC report formats:

- *XML*

- *HTML*

XML

The QAC XML adapter reads rules as `Guidelines` and rule violations as `Findings`. In this way the static analysis measures are named similarly for the different static analysis tools (e.g. MXAM, SL Check).

MQC will read from the XML file:

```
<AnalysisData timestamp="20200423T162554" projectpath="C:/Users/public/AppData/Local/
↪samples/Examples"
            reportpath="C:/Users/public/AppData/Local/samples/Examples/configs/
↪reports">
  <dataroot type="project">
    <tree type = "rules" >
      <RuleGroup name="xxx" total="5212" active="1112" >
        <Rule id = "" total="5" active="2" text="" >
          <Rule id = "" total="2" active="2" text="" >
            <Message guid = "" total="2" active="2" text="" />
          </Rule>
        </Rule>
        <Rule id = "" total="3" active="0" text="" >
          <Rule id = "" total="3" active="0" text="" >
            <Message guid = "" total="1" active="0" text="" />
            <Message guid = "" total="2" active="0" text="" />
          </Rule>
```

```
        </Rule>
      </RuleGroup>
    </tree>
  </dataroot>
  <dataroot type="per-file">
    <File path=".../example.h">
      <tree type="rules">
        <RuleGroup name = "xxx" total="84" active="53" >
          <Rule id="xxx" total="29" active="23"
                text="There shall be no occurrence of undefined or critical␣
→unspecified behaviour" >
            <Message guid = "qac-9.3.1-0602" total="6" active="0" text="" />
            <Message guid = "qac-9.3.1-0603" total="12" active="12" text="" />
            <Message guid = "qac-9.3.1-0836" total="1" active="1" text="" />
            <Message guid = "qac-9.3.1-0848" total="5" active="5" text="" />
            <Message guid = "qac-9.3.1-0854" total="5" active="5" text="" />
          </Rule>
        </RuleGroup>
      </tree>
      <tree type="levels">
        <Level guid = "QA_WARNING" total="55" active="30" name="Warnings" ></Level>
        <Level guid = "QA_ERROR" total="2" active="2" name="Errors" ></Level>
      </tree>
    </File>
  </dataroot>
</AnalysisData>
```

MQC extracts the following information, stores and transforms it to the MQC data structure:

- from the `<AnalysisData>` header:

    - `ReportDateTime`: read from attribute `timestamp`

Each `<File>` element inside `<dataroot type="per-file">` is treated as separate artifact. For each artifact `Guidelines` and `Findings` are read:

MQC treats different rule sets `<RuleGroup name = "M3CM" ...` as measurements (the name of the rule group `M3CM` is used as measurement name in MQC). All guidelines and findings are counted separately per measurement.

- from each `<File>` header:

    - `ArtifactPath`: read from attribute `path`

- per measurement (rule group):

    - `Findings.Failed`: the number of active rule violations (`active` > 0)

    - `Findings.Suppressed`: value of attribute `total` - `active` (number of suppressed rule violations)

- – `Guidelines.Suppressed`: the number of `Rule` elements with `Message` elements (lowest level) with attribute `active = "0"`

- – `Guidelines.Failed`: the number of all `Rule` elements (per artifact) with `Message` elements substracted by the number of suppressed guidelines (`Guidelines.Suppressed`).

- • from the `<dataroot type="project">` `<tree type = "rules" >` element

  - – `Guidelines.Total`: the number of `Rule` elements with `Message` elements (lowest level)

  - – `Guidelines.Passed`: `Guidelines.Total` (for all artifacts) - `Guidelines.Failed` (per artifact)

The number of passed guidelines includes the number of suppressed guidelines (all rules without and with suppressed violations)!

HTML

MQC will read from the HTML file:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>
  <body>
    <div id = "head" >
      <div class="stitle">
       Project   : C:/Users/public/AppData/Local/Examples<br/>
       Status at: 23 Apr, 2019 at 16:25:13
      </div>
    </div>
    <div class="head">
      <div class="summary">
        <table border="0">
          <col border="50"/>
          <tr><td>Number of Files</td><td>185</td></tr>
          <tr><td>Lines of Code (source files only)</td><td>16431</td></tr>
          <tr><td>Total preprocessed code line</td><td>3036</td></tr>
          <tr><td>Diagnostic Count</td><td>948</td></tr>
          <tr><td>Rule Violation Count</td><td>1212</td></tr>
          <tr><td>Violated Rules</td><td>183</td></tr>
          <tr><td>Compliant Rules</td><td>7</td></tr>
          <tr><td>File Compliance Index</td><td>97.41%</td></tr>
          <tr><td>Project Compliance Index</td><td>6.68%</td></tr>
        </table>
      </div>
    </div>
    <div id="content">
      <div class="dpp">
```

```html
        <div class="subsec"><h5>M3CM</h5></div>
        <div class="rgtable">
          <table border="1" >
            <tr><th>Files</th><th>Rule 0</th><th>Rule 1</th><td><b>Total Violations</
→b></td></tr>
            <tr>
             <td><a href="example.h" title="example.h">example.h</a></td>
             <th>2</th><th>3</th><td><b>5</b></td>
            </tr>
          </table>
        </div>
        <div class="subsec"><h5>QA-C</h5></div>
        <div class="rgtable">
          <table border="1" >
            <tr><th>Files</th><th>Rule 0</th><th>Rule 1</th><td><b>Total Violations</
→b></td></tr>
            <tr>
             <td><a href="example.h" title="example.h">example.h</a></td>
             <th>1</th><th>5</th><td><b>6</b></td>
            </tr>
          </table>
        </div>
    </div>
     <div class="worstrules">
        <div class="rgtable">
          <table border="1" >
            <tr><th>Files</th><th>Rule 0</th><th>Rule 1</th><th>Rule n</th></tr>
            <tr>
             <td><a href="example.h" title="example.h">example.h</a></td>
             <th>0</th><th>65</th><td>6</td>
            </tr>
          </table>
        </div>
        <div class="rgtable">
          <table border="1" >
            <tr><th>Files</th><th>Rule 0</th><th>Rule 1</th><th>Rule n</th></tr>
            <tr>
             <td><a href="example.h" title="example.h">example.h</a></td>
             <th>13</th><th>105</th><td>0</td>
            </tr>
          </table>
        </div>
      </div>
    </div>
  </body>
</html>
```

MQC extracts the following information, stores and transforms it to the MQC data structure:

- from `<div class="stitle">` that contains *'Status at:'* read date and time and stored in MQC as `ReportDateTime`

- from `<div id="content"><div class="dpp">`:

  – from `<div class="subsec">` read `<h5>` as `MeasurementName`.

  MQC treats different rule sets as measurments (the name of the table is stored as measurement name in MQC). All guidelines and findings are counted separately per measurement.

  – from `<div clas="rgtable"><table>` for each `<tr>`:

    * `ArtifactName`: read from first `<td>` of `<tr>`

    * `Findings.Failed`: from last `<td>` of `<tr>`.

- from `<div id="content"><div class="worstrules">` header:

  – from `<div class="subsec">` read `<h5>` as `MeasurementName`

  – from `<div class="rgtable"><table>` for each `<tr>`:

    * `Guidelines.Failed`: count each `<td>` of `<tr>` with violation (value > 0)

    * `Guidelines.Passed`: number of all guidelines for current rule - `Guidelines.Failed`

  – from `<table>` inside `<div class="summary">`:

    * `Guidelines.Ovarall Failed`: read second `<td>` of `<tr>` that first is equal to *'Violated Rules'*

    * `Guidelines.Overall Passed`: read second `<td>` of `<tr>` that first is equal to *'Compliant Rules'*

    * `Guidelines.Overall Total`: sum of `Project Rules.Violated` and `Project Rules.Compliant`

TargetLink Code Coverage

The TargetLink Code Coverage adapter reads the base coverage measures (total/reached/unreached branches) and the already calculated code coverage measure for statement and branch/decision coverage from the HTML report file.

The adapter expects report files, which end with `ccdoc_Main.html`.

```
<body>
  <table>
    <tr><td>TL Code Coverage Report for Application</td><td> : </td><td>
→Model_04</td></tr>
    <tr><td>Generated by User</td><td> : </td><td>UserName</td></tr>
    <tr><td>Date and time of report generation</td><td> : </td><td>2021-02-
→13 04:37:08</td></tr>
    <tr><td>Code Coverage level</td><td> : </td><td>Decision Coverage (C1)</
→td></tr>
```

(continues on next page)

```
  </table>
  <table>
    <tr>
      <th> </th>
      <th>Code Coverage</th>
      <th>Total Branches</th>
      <th>Reached Branches</th>
      <th>Unreached Branches</th>
    </tr>
    <tr>
      <td>Model_04</td>
      <td>96.60 %</td>
      <td>676</td>
      <td>653</td>
      <td>23</td>
    </tr>
    ...
  </table>
</body>
```

From the first table MQC extracts the following data:

- read third `<td>` of `<tr>` where the first is equal to *'Date and time of report generation'* (stored in MQC as `ReportDateTime`)

- read third `<td>` of `<tr>` where the first is equal to *'Code Coverage level'* (used in MQC as base measure name, e.g. when reading `Decision Coverage (C1)` the base measure name will be `Code Coverage\Decision`)

From the second table MQC reads the header row and the first data row to extract the following data:

- read first `<td>` of second `<tr>` as `ArtifactName`

- read each `<th>` of first `<tr>` as variable name (ignoring the first - empty - one) and the corresponding `<td>` of the second `<tr>` as measure value

  - `Total Branches` stored with variable name `Total`

  - `Reached Branches` stored with variable name `Covered`

  - `Unreached Branches` stored with variable name `Uncovered`

Rational Test RealTime (RTRT)

MQC supports Rational Test RealTime reports in html format.

MQC will read from index.htm file `Generated on` date as `ReportDateTime`.

```
<table WIDTH="100%" BORDER="2" >
  <tr><td>Name </td ><td>Status</td ><td>Failed</td ><td>Passed</td ><td>Total</td >
↪</tr>
```

```
   <tr><td>ArtifactPath.xrd</td><td>Passed</td><td>0 </td ><td>28</td ><td>28</td ></
→tr>
</table>
```

MQC extracts the `Testcase\Result` measure information from *index.html* file in *Reporter* directory, stores and transforms it to the MQC data structure:

- `Name` column stored in MQC as `ArtifactPath`

- `Failed` column stored in MQC as `Testcase\Result.Failed` variable

- `Passed` column stored in MQC as `Testcase\Result.Passed` variable

- `Total` column stored in MQC as `Testcase\Result.Total` variable

Any extension (including '_' and 'numbers') at the end of the artifact path will be removed, if there is at least one other row with the same base name, e.g. 'Artifact' and 'Artifact1' or 'Artifact_1' will all be reduced to 'Artifact'. In that case the read measure values will be summed up per variable name.

```
<table>
 <tr>
  <td><b>Item</b></td>
  <td><b>Functions</b></td>
  <td><b>Functions and exits</b></td>
  <td><b>Statement blocks</b></td>
  <td><b>Decisions</b></td>
  <td><b>Basic conditions</b></td>
  <td><b>Modified conditions</b></td>
  <td><b>Multiple conditions</b></td>
 </tr>
 <tr>
  <td><B>ArtifactPath.C</B></td>
  <td> 10 / 10 </td>
  <td> 35 / 35 </td>
  <td> 35 / 36 </td>
  <td> 40 / 46 </td>
  <td> N/A </td>
  <td> N/A </td>
 </tr>
</table>
```

MQC extracts the coverage information from the last table of *RateDoc.html* file in *Cvi* directory, stores and transforms it to the MQC data structure:

- `Item` column stored in MQC as `ArtifactPath`

- all other columns are measures with the column header as base measure name (by capitalizing the words, making the last word singular and prefixing it with `Code Coverage\`):

  - read number before / stored in MQC as `Covered` variable.

  - read number after / stored in MQC as `Total` variable.

– read the difference between these two numbers stored in MQC as `Uncovered` variable.

For the sample data extracted data is:

- Code Coverage\Function.Covered=10 , Code Coverage\Function.Total=10 , Code Coverage\Function.Uncovered=0

- Code Coverage\Functions And Exit.Covered=35 , Code Coverage\Functions And Exit.Total=35 , Code Coverage\Functions And Exit.Uncovered=0

- Code Coverage\Statement Block.Covered=35 , Code Coverage\Statement Block.Total=36 , Code Coverage\Statement Block.Uncovered=1

- Code Coverage\Decision.Covered=40 , Code Coverage\Decision.Total=46 , Code Coverage\Decision.Uncovered=6

MQC keeps the extension of the read artifact path ('.h' / '.c'), but removes the following '#' and 'numbers' from end of the name, e.g. 'Artifact.c' and 'Artifact.c #1' will all be reduced to 'Artifact.c'. In that case the read measure values will be summed up per variable name.

MathWorks Simulink Requirements

MQC supports MathWorks Simulink Requirements reports in html format.

```
<span class="SLReqReportTitleAttribute">Published on</span>
<span>:</span>
<span class="SLReqReportTitleAttribute">23-Feb-2022</span>
<p><span class="SLReqReqSetImplementationTitle">Implementation Status</span></p>
<table class="SLReqReqImplementationTable">
   <tr>
       <th class="SLReqReqImpTableHeader">
        <span><span class="SLReqReqSetImpTotalName">Total</span></span>
       </th>
       <th class="SLReqReqImpTableHeader">
        <span><span class="SLReqReqSetImpImplementedName">Implemented</span></span>
       </th>
       <th class="SLReqReqImpTableHeader">
        <span><span class="SLReqReqSetImpJustifiedName">Justified</span></span>
       </th>
       <th class="SLReqReqImpTableHeader">
        <span><span class="SLReqReqSetImpNoneName">None</span></span>
       </th>
   </tr>
   <tr>
       <td class="SLReqReqImpTableBody"><span><span class="SLReqReqSetImpTotalValue">
↪11</span></span></td>
       <td class="SLReqReqImpTableBody"><span><span class=
↪"SLReqReqSetImpImplementedValue">3</span></span></td>
       <td class="SLReqReqImpTableBody"><span><span class=
↪"SLReqReqSetImpJustifiedValue">0</span></span></td>
```

(continues on next page)

```
      <td class="SLReqReqImpTableBody"><span><span class="SLReqReqSetImpNoneValue">8
→</span></span></td>
   </tr>
</table>
```

MQC extracts the following information, stores and transforms it to the MQC data structure:

- the `ReportDateTime` from the `<span class="SLReqReportTitleAttribute">` tag, after the same tag contains `Published on`

- from the `<table class="SLReqReqImplementationTable">` table under the `<span class="SLReqReqSetImplementationTitle">` tag, read first row as variable names and second row as values:

    - `ImplementationStatus.Total`

    - `ImplementationStatus.Implemented`

    - `ImplementationStatus.Justified`

    - `ImplementationStatus.None`

- read the report file name as `ArtifactName`

Generic data sheet

MQC provides the possibility to read any CSV or Excel data sheet as a data source. For that, the user has to define adapter options. Adapter options for the Generic Data Sheet adapter are rules to map for instance the columns of an Excel table to MQC data dimensions like revisions, artifacts, measures etc. Additionally, another option may specify how to filter rows.

The following code block shows an example of an adapter option to read MXAM data from an Excel sheet.

The "ImportDefinitions" define how to fetch the relevant data

- FileExpression: name pattern of files that could be interpreted by this adapter option

- ArtifactName: fetched from table column [SubComponents_Path]

- Value: fetched from table column [Count]

An Excel sheet sample that could be read with this adapter option is shown in Figure 4.53

```
$schema: http://quality-commander.de/userguide/v71/schema/AdapterOptionSource.schema.
→json
$version: 1.0
Name: 'GenericDataSheet: MXAM Adapter Option'
AdapterOptions:
- $type: MES.MQC.DataSourceLibrary.Adapters.Others.
→GenericDataSheetAdapter+AdapterOptions
  ImportDefinitions:
  - FileExpression: MXAMReport.*_[0-9]+\.xlsx
```

```
ArtifactPath: '[SubComponents_Path]'
ArtifactName: '[SubComponents_Path]'
DataSource: MXAM
MeasurementName: '[Chapters]'
MeasureName: '[MeasureName]'
VariableName: '[ResultType]'
Value: '[Count]'
ReportDateTime: MXAMReport.*_(\d{8}).xlsx
ReportDateTimeFormat: yyyyMMdd
```

Documents_Id	Authors	SubComponents_Path	Chapters	MeasureName	ResultType	Count
d122ef22-b8e7-4720-9908-c6be14e530e3	Model Engineering Solutions GmbH	EV3Control	GuidelineAnalysis	FindingCount	Failed	5,00
d122ef22-b8e7-4720-9908-c6be14e530e3	Model Engineering Solutions GmbH	GlobalPosition	GuidelineAnalysis	FindingCount	Passed	20,00

Figure 4.53: Sample Excel sheet for reading MXAM data by Generic data sheet adapter

Manual data import (Excel Template)

MQC provides the possibility to import data from any other data source using the manual import option.

All measures imported into MQC have to be configured in the Quality Model (see *Base Measures*). Imported but not configured measures will be ignored in all visualizations as well as for the quality calculation.

The manual import has to be done using Excel. From the **Data Sources** dialog choose the `Data Template` button to create an Excel import file that can be used to load data into MQC.

Figure 4.54: Export the Data template for using manual data import functionality

Select one of the following options to customize the data import template according to your needs:

- `All`: The Excel file already contains a column for each measure configured in the quality model as well as one row for each artifact imported respectively configured in the project structure.

- `Filtered`: The Excel file only contains those measures and artifacts selected via the filter panel on the right-hand side of the pages.

- `Marked`: The Excel file only contains those measures and artifacts that were marked by the user, e.g. if the user have selected an artifact KPI and a specific data source KPI at the Data Status page, the file only contains a row for the marked artifact as well as only columns for measures belonging to the marked data source.

Artifact	ReportDate	TPT.ModelTest.DecisionCoverageC1.Reached	TPT.ModelTest.DecisionCoverageC1.Total
EV3Control_de	13.08.2019	80	90
VehicleManage	13.08.2019	70	85

Figure 4.55: Sample file for manual import of two Base Measures for two Artifacts for the same revision

The column `Artifact` consists of entries representing the objects for which data shall be collected and for which quality shall be computed (e.g. Simulink models, requirements documents, software components).

The entries of the column `ReportDateTime` are considered by MQC as the days when raising the data, which shall be collected. Those configured report dates are used to assign the imported data to MQC revisions.

Finally, all other columns are representing measures for which data shall be imported into MQC. As shown in Figure 4.55, measures (i.e. the column names) have to follow the syntax `DataSource.Measurement.BaseMeasure.Variable`.

- `DataSource`: Specify where your data is coming from (e.g. the name of the tool which produced the data).

- `Measurement`: Provide more structuring regarding the data (e.g. the reason why data is collected, the test environment etc.). This is optional and may be left empty. In this case the measure name syntax is `DataSource.BaseMeasure.Variable`. Nevertheless, it is recommended to fill out `Measurement` to achieve a high compliance to ISO 250xx.

- `BaseMeasure`: Specifies a group of measures.

- `Variable`: The name of the specific measure belonging to the base measure group.

The value for a specific measure for an artifact at a certain report date then has to be assigned to the corresponding cell within the Excel template.

In case you might want to assign default values for your imported base measures, you have to define them in compliance with this syntax in the quality model (see *Base Measures*).

4.6.2 API Adapters

4.6.3 Custom Adapters

Custom Adapters, provided by MES, available as Open Source or developed yourself, can be added in this Dialog. On import a C# class file gets compiled while an IronPython script is executed and validated. If an error occurs, the import fails and the error messages are shown. A custom adapter source file (.cs or .py) can contain multiple custom adapters.

Figure 4.56: Compiling error messages on a failed import of a Custom Adapter

After a custom adapter file is imported it will be saved together with the current MQC Project as a library item or a dxp file. If the source file is changed it has to be manually reloaded in this dialog so that it is compiled again and the adapter is updated in the MQC Project.

Additionally to the functionality of managing the adapters, the Adapters Dialog provides a button to view the Execution Order of the Adapters, which is defined in the Adapters itself.

With the Button "Test importing a Report File" a single data source file can be tested for import. If there are any errors while checking the Adapters and executing the correct Adapter, the error messages are shown to the user, else the data is shown in a result table.

Developing a Custom Adapter

An MQC Data Source Adapter has to inherit the **MES.MQC.DataSourceLibrary.Adapters.Adapter** class.

A Custom Adapter can be either a .net C# class or a IronPython class.

The File Adapters, API Adapters and Example Adapters can be downloaded from the Adapters Dialog in MQC.

When developing a new custom adapter, the following properties and methods have to be implemented:

Required Properties

- DataSource: string
- FileExtensions: List<string>

Figure 4.57: Execution Order of all enabled Adapters. The Adapters are executed in this order, by first checking if the file extension(s) match the file that is to be imported and then let the Adapter validate itself, if the provided file is to be handled by it.

Figure 4.58: Adapters Dialog in MQC: Download-Button of Adapter Examples on the right side.

Optional Properties

• Name: string (defaults to class name)

• Description: string (defaults to empty)

• Priority: integer (defaults to 100)

• FindingsProvidedForDataSources: List<string> (default to empty list)

Required Methods

• IsValid: bool

• Read: AdapterReadResult

Optional Methods - GetHumanReadableFilePath: string - GetHumanReadableFilePaths: List<string>

Adapter Class

```
public abstract class Adapter
{
    /// <summary>
    ///    Unique Name of the Adapter
    ///    Defaults to the ClassName, can be overridden with a
    ///    user defined Name
    /// </summary>
    public virtual string Name

    /// <summary>
    ///    Description of the Adapter that is visible in the
    ///    Adapter-Dialog as a Popover
    ///    Absolute links get transformed into HTML Link-Tags,
    ///    Linebreaks (\n) get transformed into HTML linebreaks (<br>)
    ///    HTML Tags are not allowed
    /// </summary>
    public virtual string Description

    /// <summary>
    ///    File Extensions of the Adapter
    ///    This Property has to be defined and has to have at least one
    ///    file extension
    ///    The Adapter is only used for FilePaths with the defined
    ///    file extensions. The IsValid method is not called unless
    ///    the file extension matches.
    /// </summary>
    public abstract List<string> FileExtensions

    /// <summary>
    ///    Priority of the Adapter
    ///    The execution order of all adapters depend on the defined
```

```csharp
///     priorities. The higher the priority the earlier the adapter
///     is validated and executed.
///     The default priority for Tool Adapters is between 10-110,
///     Custom Adapters should define a priority of 200 or higher if
///     they should be executed before the tool adapters.
///     If two adapters have the same priority, the order is depending
///     on the Name of the adapter.
/// </summary>
public virtual int Priority


/// <summary>
///     The default Data Source of the Adapter
///     This Property has to be defined and is taken as data source
///     name for all data imported using this adapter.
///     If a report file contains data from multiple data sources, set
///     this Property to a default name (e.g. "Unknown") and
///     use the DataSource Property of each AdapterData object to define
///     a dedicated Data Source per metric.
/// </summary>
public abstract string DataSource


/// <summary>
///     List of DataSources for which the adapter can provide findings.
/// </summary>
public virtual List<string> FindingsProvidedForDataSources => new List<string>();


/// <summary>
///     IsValid has to be implemented by the Adapter class
///     The method is called when the file extensions match
///     If the file extension is unique this method can just return
///     true, else it should check if the file should be imported by
///     the adapter
///     If true is returned, the current adapter is executed and the
///     Read method is called. No other adapter is checked afterwards.
/// </summary>
protected abstract bool IsValid(AdapterContext context);


/// <summary>
///     Read has to be implemented by the Adapter class
///     This method is called when the file extensions match and
///     isValid returns true, no other adapter is called
///     The data of the file should be read and returned as a List
///     of AdapterData
/// </summary>
protected abstract AdapterReadResult Read(AdapterContext context);


/// <summary>
```

```
///     Number of Human Readable Files
/// </summary>
public virtual int HumanReadableFileCount


/// <summary>
///     Human Readable File Extensions of the Adapter
/// </summary>
public virtual List<string> HumanReadableFileExtensions


/// <summary>
///     Human Readable File Names of the Adapter
/// </summary>
public virtual List<string> HumanReadableFileNames


/// <summary>
///     Start of Human Readable File Names of the Adapter
/// </summary>
public virtual List<string> HumanReadableStartFileNames


/// <summary>
///     Get the human readable file path
///     Can be overridden with a user defined function
/// </summary>
protected virtual string GetHumanReadableFilePath(string filePath)


/// <summary>
///     Get the human readable file paths
///     Can be overridden with a user defined function
/// </summary>
protected virtual List<string> GetHumanReadableFilePaths(string filePath)


/// <summary>
///     TransformStringToValue is a Utility method
///     Parse a string value to a double value in a
///     culture-independent (invariant) way.
/// </summary>
protected double? TransformStringToValue(string value)
```

AdapterContext Class

```
public class AdapterContext
{
    /// <summary>
    ///     Should the adapter import findings?
    /// </summary>
```

```csharp
    public bool ImportFindings { get; }


    /// <summary>
    ///    File name of the report file
    /// </summary>
    public string Name { get; }


    /// <summary>
    ///    File name without extension of the report file
    /// </summary>
    public string NameWithoutExtension { get; }


    /// <summary>
    ///    File extension of the report file
    /// </summary>
    public string Extension { get; }


    /// <summary>
    ///    File path of the report file
    /// </summary>
    public string Path { get; }


    /// <summary>
    ///    Directory path of the report file
    /// </summary>
    public string DirectoryPath { get; }


    /// <summary>
    ///    File content of the report file
    /// </summary>
    public string Content { get; }


    /// <summary>
    ///    Modified date of the report file
    /// </summary>
    public DateTime CreationDate { get; }


    /// <summary>
    ///    Message bag, where errors and warnings can be thrown
    /// </summary>
    public AdapterMessageBag MessageBag { get; }


    /// <summary>
    ///    It loads the file path as an XDocument, which can be traversed via XPath
    ///    See https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/
    →concepts/linq/linq-to-xml-overview for documentation.
    ///    The document is stored in a property and returned. The reading of the file␣
    →is therefore only done once.
```

```
    /// </summary>
    public XDocument AsXDocument()


    /// <summary>
    ///     It loads the file path as an HtmlDocument (HtmlAgilityPack)
    ///     See https://html-agility-pack.net/ for documentation.
    ///     The document is stored in a property and returned. The reading of the file
→is therefore only done once.
    /// </summary>
    public HtmlDocument AsHtmlDocument(string subContent = null)


    /// <summary>
    ///     It loads the file path as an DataSet
    ///     The DataSet is stored in a property and returned. The reading of the file
→is therefore only done once.
    /// </summary>
    public DataSet AsDataSet()
}
```

AdapterMessageBag Class

```
public class AdapterMessageBag
{
    /// <summary>
    ///     ThrowError is a Utility method
    ///     Throws an Error message to be either ignored, displayed at
    ///     a notification (on data source import or refresh)
    ///     or shown in a validation dialog (if a report file is
    ///     imported as a test).
    ///     The import of the current filePath is aborted with an
    ///     internal exception
    /// </summary>
    public void ThrowError(string title, string description)


    /// <summary>
    ///     ThrowWarning is a Utility method
    ///     Throws an Warning message to be either ignored, displayed
    ///     at a notification (on data source import or refresh)
    ///     or shown in a validation dialog (if a report file is
    ///     imported as a test).
    ///     The import of the current filePath continues unimpeded by
    ///     the warning.
    /// </summary>
    public void ThrowWarning(string title, string description)
```

AdapterReadResult Class

```
public class AdapterReadResult
{
    /// <summary>
    ///     The imported data (aggregated numbers)
    /// </summary>
    public List<AdapterData> Data { get; }


    /// <summary>
    ///     The imported measures (which data should be defaulted if not imported)
    /// </summary>
    public List<AdapterMeasure> Measures { get; }


    /// <summary>
    ///     The imported findings (data details)
    /// </summary>
    public List<AdapterFinding> Findings { get; }
}
```

AdapterData Class

```
public class AdapterData
{
    /// <summary>
    ///     Date of the report
    /// </summary>
    public DateTime DateTime { get; set; }


    /// <summary>
    ///     Name of the Data Source
    ///     This property is optional and can remain null, in this case
    ///     the DataSource property of the CustomAdapter is used
    /// </summary>
    public string DataSource { get; set; } = null;


    /// <summary>
    ///     Path of the Artifact
    /// </summary>
    public string ArtifactPath { get; set; }


    /// <summary>
    ///     Name of the Artifact
    ///     This property is optional, the ArtifactName will be taken
    ///     from the Project Structure if available or derived from the
    ///     artifact path if not.
```

(continues on next page)

```csharp
    /// </summary>
    public string ArtifactName { get; set; }


    /// <summary>
    ///     Name of the Measure
    /// </summary>
    public string MeasureName { get; set; }


    /// <summary>
    ///     Name of the Measurement
    /// </summary>
    public string MeasurementName { get; set; }


    /// <summary>
    ///     Name of the Variable
    /// </summary>
    public string VariableName { get; set; }


    /// <summary>
    ///     The Value
    ///     Has to be of the double type
    /// </summary>
    public double? Value { get; set; }
}
```

AdapterMeasure Class

```csharp
public class AdapterMeasure
{
    /// <summary>
    ///     Date of the report
    /// </summary>
    public DateTime DateTime { get; set; }


    /// <summary>
    ///     Name of the Data Source
    ///     This property is optional and can remain null, in this case
    ///   the DataSource property of the CustomAdapter is used
    /// </summary>
    public string DataSource { get; set; } = null;


    /// <summary>
    ///     Path of the Artifact
    /// </summary>
    public string ArtifactPath { get; set; }
```

```csharp
    /// <summary>
    ///     Name of the Measure
    /// </summary>
    public string MeasureName { get; set; }


    /// <summary>
    ///     Name of the Measurement
    /// </summary>
    public string MeasurementName { get; set; }


    /// <summary>
    ///     Name of the Variable
    /// </summary>
    public string VariableName { get; set; }
}
```

AdapterFinding Class

```csharp
public class AdapterFinding
{
    /// <summary>
    ///     Date of the report
    /// </summary>
    public DateTime DateTime { get; set; }


    /// <summary>
    ///     Name of the Data Source
    ///     This property is optional and can remain null, in this case
    ///    the DataSource property of the CustomAdapter is used
    /// </summary>
    public string DataSource { get; set; } = null;


    /// <summary>
    ///     Path of the Artifact
    /// </summary>
    public string ArtifactPath { get; set; }


    /// <summary>
    ///     SubStructurePath of the Artifact
    /// </summary>
    public string ArtifactSubStructurePath { get; set; }

    /// <summary>
    ///     Structure path where the method/result/etc of the Finding is connected to
```

```csharp
    /// </summary>
    public string FindingStructurePath { get; set; }


    /// <summary>
    ///     Result value of the Finding (e.g. Warning, Failed)
    /// </summary>
    public FindingResult Result { get; set; }


    /// <summary>
    ///     Display name of the structure of the Finding
    /// </summary>
    public string DisplayName { get; set; }


    /// <summary>
    ///     Description of the Finding (contains detailed issue information)
    /// </summary>
    public string Description { get; set; }


    /// <summary>
    ///     Anchor of the related human readable html file
    /// </summary>
    public string HumanReadableAnchor { get; set; }


    /// <summary>
    ///    Add multiple adapterData entries to create relations
    /// </summary>
    public void AddData(IEnumerable<AdapterData> data)


    /// <summary>
    ///    Add an adapterData entry to create a relation
    /// </summary>
    public void AddData(AdapterData data)
}
```

4.6.4 General Adapter Options

File adapters support the configuration of FilePath based Adapter Options for the following data fields:

- ArtifactPath

- DataSourceName

- MeasurementName

- ReportDateTime

The values for these fields can be extracted from the report file path with regex expressions.

In the adapter options, the data field has to be appended with a "FromFilePath". (e.g. "ArtifactPathFrom-

FilePath") You can specify multiple definitions per data field. The definitions are checked in order and the first matching definition is used.

Each definition contains:

- **Regex**

  A regular expression to match the file path.

- **Result**

  The resulting value from the regex. Static values or dynamic regex results like $1 can be used.

- **IsFallback** (optional, default is false)

  Should the definition be only used if there was no value read from the adapter itself.

- **Format**

  Only relevant for ReportDateTime. DateTime-Format (e.g. "yyMMdd")

Listing 4.21: Example of general adapter options defined for TptXmlAdapter for reading 'Artifact-Path', 'MeasurementName' and 'ReportDateTime' from file path

```
AdapterOptions:
- $type: MES.MQC.DataSourceLibrary.Adapters.Tools.TptXmlAdapter+AdapterOptions
  ArtifactPathFromFilePath:
    - Regex: ^.+\\(.+)_TPT[^\/]*\.xml$
      Result: $1
    - Regex: ^.+\\Model_([^\\]+).*$
      Result: $1
  MeasurementNameFromFilePath:
    - Regex: ^.+_(Report|Data)_([^\\]+).*$
      Result: $2
  ReportDateTimeFromFilePath:
    - Regex: ^.+\\([0-9]+)_(Report|Data)_.+$
      Format: yyMMdd
      Result: $1
```

4.7 PAGES

MQC structures all the important information of your project and shows it on different pages. Mainly, there are three kinds of pages, showing either quality, the imported data or data availability (see *Interactive Pages*).

On all pages the main visualization always shows one revision with the possibility to switch revisions from the dropdown menu above (see *Revision Selection*). By default, **Everything** means that all revisions with imported data will be shown. For example, if revision granularity is set to *Days*, days without imported data are not shown in the visualizations.

All pages contain interactive visualizations. If you click into any of the visualizations, the other visualizations will react and show information related to what you have marked (see *Marking*).

Based on the Project Structure you have imported, the filter panel on the right (see *Filter Panel*) will display options for filtering the data.

All MQC pages can be easily enabled or disabled by selecting the corresponding page in the **Pages** dialog of the left-hand side panel .

In the **Pages** dialog the Layout Sources, Dashboard Sources and Custom Pages can be configured, saved and exported similar to all other configuration sources (see *Configuration Sources*).

4.7.1 Pages Layouts

MQC allows you to create custom page layout configurations for the Interactive pages, so you can define personalized pages that will fit your needs and only display the relevant data.

MQC is shipped with one default Page Layout Template source containing different Page Layout definitions. The active page layouts are stored in the Page Layout sources: **QualityPageLayout.yml**, **DataPageLayout.yml** and **AvailabilityPageLayout.yml**. The Template source contains the different page `Templates` for one `Page` with given `Name` (e.g., for `Quality` named `Status` as shown in Listing 4.22). You can define a custom `Layout` where visualizations for the three main areas of the page: `Top`, `Kpi` and `Main` can be specified. For each area `Visuals` can be defined with the corresponding `Identifier`. Which visualization should be active is specified in `ActiveVisualization` by giving the `Identifier` of the visualization. The source can be defined as shown in Listing 4.22.

Listing 4.22: Code snipped from the default MQC Page Layout Template source in YAML

```
$schema: http://quality-commander.de/userguide/v71/schema/PageLayoutSource.schema.json
$version: 1.0
Templates:
  - Page: Quality
    Name: Status
    Layout:
      Top:
        - Visuals:
            - Identifier: QualityBinTrend # Visualization tab 1
            - Identifier: QualityTrend # Visualization tab 2
          ActiveVisual:
            Identifier: QualityBinTrend # Visualization tab 2
      Kpi:
        - Visuals:
            - Identifier: QualityKpiArtifact # Visualization tab 3
          ActiveVisual:
            Identifier: QualityKpiArtifact # Visualization tab 3
        - Visuals:
            - Identifier: QualityKpiQualityProperty # Visualization tab 4
          ActiveVisual:
            Identifier: QualityKpiQualityProperty # Visualization tab 4
      Main:
        - Visuals:
```

Figure 4.59: Pages dialog for page configurations

```
                    - Identifier: QualityHeatmap # Visualization tab 5
                    - Identifier: QualitySunburst # Visualization tab 6
                    - Identifier: QualityDataOrigin # Visualization tab 7
                ActiveVisual:
                    Identifier: QualityHeatmap # Visualization tab 5
    - Page: Quality
      Name: Trend
      Layout:
          Top:
            - Visuals:
                - Identifier: QualityBinTrend
                - Identifier: QualityTrend
              ActiveVisual:
                Identifier: QualityBinTrend
          Kpi:
            - Visuals:
                - Identifier: QualityKpiArtifact
              ActiveVisual:
                Identifier: QualityKpiArtifact
            - Visuals:
                - Identifier: QualityKpiQualityProperty
              ActiveVisual:
                Identifier: QualityKpiQualityProperty
          Main:
            - Stacked:
                - Visuals:
                    - Identifier: QualityTrendByArtifact
                  ActiveVisual:
                      Identifier: QualityTrendByArtifact
                - Visuals:
                    - Identifier: QualityTrendByQualityProperty
                  ActiveVisual:
                      Identifier: QualityTrendByQualityProperty
```

If you want some visualizations to be shown on top of each other in one area you can use the keyword `Stacked` and specify the `Visuals` as shown in Listing 4.22.

The current Layout for a Page can be changed from the Page Layout Switcher in the left hand side (see *Templates*).

4.7.2 Dashboards

MQC provides the possibility to create custom dashboard page configurations. This allows you to define personalized dashboard pages that fit your needs.

To configure a Dashboard source, you need to create a YAML file in which you specify the `Authors` and `DashboardPages`. For each dashboard page `Name` and `Items` should be specified. You can define a vi-

Figure 4.60: Default Quality Page

sualization by adding the `Visual` in `Items`. The `Visual` should contain the category and the name of the visual (e.g., Project.InformationCard). In addition, the `Position` (`X` and `Y`) and the `Size` (`Width` and `Height`) of the element should be defined. It is also possible to hide the title of the visualization by setting the property `TitleBarHidden` to true.

The configuration can be done in YAML following the schema:

Listing 4.23: Defining dashboard source in YAML

```
$schema: http://quality-commander.de/userguide/v71/schema/DashboardSource.schema.json
$version: 1.0
Authors:
    - Model Engineering Solutions GmbH
DashboardPages:
    - Name: 'Dashboard'
    Items:
    - Visual: 'Project.InformationCard'
        Position:
            X: 0
            Y: 0
        Size:
            Width: 6
            Height: 8
    - Visual: 'Quality.TrendLineChart'
        Position:
```

(continues on next page)

```
        X: 6
        Y: 0
    Size:
        Width: 18
        Height: 8
 - Visual: 'Data.WorstArtifactsKpiChart'
    Position:
        X: 16
        Y: 11
    Size:
        Width: 8
        Height: 5
    TitleBarHidden: true
```

In this way a dashboard file may be used for multiple projects as well.

4.7.3 Custom Pages

Custom pages and charts are configured using Excel.

Page Title	Chart Title	Chart Type	Measure Name	Chart Data Reference
Code Condition Coverage	Condition Coverage Quality Trend	Trend	Code Condition Coverage	Quality
Code Condition Coverage	Condition Coverage Measure Trend	Trend	MTest.ModelTest.Code Condition Coverage.Reached	Data
Code Condition Coverage	Condition Coverage Measure Trend	Trend	MTest.ModelTest.Code Condition Coverage.Total	Data
Guideline Compliance	Guideline Compliance Status	Status	Guideline Compliance (GuidelineAnalysis)	Quality
Guideline Compliance	Guidelines Status	Status	MXAM.GuidelineAnalysis.GuidelineCount.Failed	Data
Guideline Compliance	Guidelines Status	Status	MXAM.GuidelineAnalysis.GuidelineCount.Passed	Data
Guideline Compliance	Findings Status	Status	MXAM.GuidelineAnalysis.FindingCount.Failed	Data
Guideline Compliance	Findings Status	Status	MXAM.GuidelineAnalysis.FindingCount.Passed	Data

Figure 4.61: Configuration of two additional pages, each page with two charts

The configuration has the following structure:

- **Page Title:** a user defined title for the current custom page

- **Chart Title:** a user defined title for the current chart to be shown within the current custom page

- **Chart Type:** the type of the current chart

  – use **Status** for status bar charts

  – use **Trend** for trend line charts

- **Measure Name:** the full qualified name of the measure to be shown in the current chart; measure names are expected in the following notation:

  – Base Measures:

  DataSourceName.MeasurementName.BaseMeasureName.VariableName

  – Derived Measures:

  DataSourceName.MeasurementName.DerivedMeasureName.VariableName

– Quality Properties:

QualityPropertyName

- **Chart Data Reference:** the type of the current measure to be shown in the current chart, which must be either "Data" (for base measures and derived measures) or "Quality" (for quality properties)

It is not possible to combine different types of measures within the same chart! A chart must either contain data measures, i.e. base measures and/or derived measures, or quality values, i.e. quality properties.

It is possible to combine different chart types (status and trend) on the same page, whereas it is not recommended. This is because of the specific marking behavior, which is not the same for status and trend charts. Selecting a specific artifact and/or a specific revision affects all visualizations of a certain page.

4.8 ADVANCED

4.8.1 Annotations

Use annotations sparingly to describe or justify the quality of certain quality properties and artifacts. Modify the quality value or bin only if necessary. For more information see *Annotations*.

Create and Edit Annotations

Annotations can be configured in the **Annotations** dialog.

To create a new annotation click `Create` at the top right of the dialog.

Figure 4.62: Create a new annotation by clicking on the `Create` button.

This opens a form that has to be filled out by the user.

In case of marked certain artifacts or quality properties from the KPIs, or an element from the main visualization, the marked elements are pre-selected respectively in the `Annotations` dialog and in the `Create Annotation` dialog. If you want to remove this pre-selection, simply uncheck the `Show marked only` check box at the top of the Annotations dialog.

If an element was marked in the main visualization, the `Valid From`, `Valid To` and `Bin` fields will also be filled automatically with the corresponding information.

Figure 4.63: Prefilled form to create an annotation after selecting a quality tile in the Quality Status matrix.

When no quality properties or artifacts were selected, the form is empty and has to be filled out completely.

The following fields are available to define an annotation.

- **Artifacts:** (mandatory)

  Select one or more artifacts this annotation should be applied to.

- **Quality Properties:** (mandatory)

  Select one or more quality properties this annotation should be applied to.

- **Title:** (mandatory)

  Add a caption or a title to identify the annotation easily or to provide a quick overview of the annotation.

- **Description:** (optional)

  Add a more detailed justification or description of the annotation.

- **Valid From:** (optional)

  Select the start date from which the annotation is active. If left empty MQC treats the annotation as active from the start date of the project.

- **Valid To:** (optional)

  Select the end date until which the annotation is active. If left empty MQC treats the annotation as active until the end date of the project.

- **Condition:**

  Define the qualifying condition based on which the underlying quality value or bin would be changed.

- **Target:**

  Define the target quality value or bin you would like to annotate when the condition defined above is true.

- **Disabled:**

  Enable or disable an annotation. Disabled annotations are not applied but remain in the project and can be enabled at any point in time.

- **Annotation Source:** (mandatory)

  Assign the annotation to annotation source. By default a *New Annotation Source* with the name *NewAnnotationSource.yml* will be added and the annotation will be stored in this source. Later *NewAnnotationSource.yml* can be saved (see *Managing Configurations*).

You can use different combinations of qualifying conditions and desired target values for an annotation. For example:

- If you want to change a quality bin from Bad to Acceptable, set the Condition type to Bin and select the quality bin `Bad`. Then set the Target type to Bin and select the quality bin `Acceptable`.

- If a quality bin is too broad you could use the Quality condition instead. Set the Condition type to Quality and select a range between `10` to `15`. Then set the Target type to Bin and select the quality bin `Acceptable`.

- If the target bin makes the quality too high and you would like to specify it directly, then set the Target type to Quality and the quality value to `25`.

- Finally, you can also define the condition as bin and target as value. Set the Condition type to Bin and select the quality bin `Bad`. Then set the Target type to Quality and the quality value to `25`.

If the target is a bin then the underlying quality value is also changed and vice versa. The respective values are chosen from the bin configuration (see *Quality Bins*).

Once annotations are created, they can be seen in the Annotation dialog. If any field is invalid, it is marked in red, the annotation is grayed out, and the annotation cannot be applied to the project, for example, if the `Valid From` or `Valid To` dates are outside the timeline of the project.

Figure 4.64: Annotation dialog after annotation is created.

Each annotation can be edited in the Edit Annotation dialog which is the same as the Add Annotation (Figure 4.63). Click `Update` to apply the changes.

Figure 4.65: Quality status page with annotations applied to your project. `Non-Annotated Quality` stands for the quality value before annotation was applied.

4.8.2 Artifact Mapping Patterns

Artifact Mapping allows you to adapt artifact names as they are used by data sources to your needs. Especially if different data sources use different denominations for the same artifact, **Artifact Mapping** may be used to define a common artifact name. All imported measure values collected by the data sources are automatically assigned to the new artifact name.

Listing 4.24: Different artifact paths are mapped to a common artifact name in YAML

```
Artifacts:
- Name: ObstacleDetection
  Paths:
  - ObstacleDetection
  - EV3Control_demo_ec/VehicleManager/ObstacleDetection
  - ObstacleDetection_demo_ec
  - Obstacledetection_demo_ec
  - ObstacleDetection_demo_ec/ObstacleDetection
```


	A	B	C	D	E	F	G
1	ArtifactName	ArtifactWeight	ContextCategories	StructureSoftware	StructureTeams	StructureComponents	ArtifactPaths
2	EV3Control_main	1		SDOAC SW	Team F	Integration	EV3Control_main EV3Control_demo_ec EV3Control_demo_ec/EV3Control
3	ObstacleDetection	1	MLC_CC	SDOAC SW	Team F	Detection	ObstacleDetection EV3Control_demo_ec/VehicleManager/ObstacleDetection ObstacleDetection_demo_ec Obstacledetection_demo_ec ObstacleDetection_demo_ec/ObstacleDetection
4	GlobalPosition	1	Global_CC	SDOAC SW	Team P	Control	GlobalPosition EV3Control_demo_ec/VehicleManager/VehicleControl/GlobalPosition GlobalPosition_demo_ec GlobalPosition_demo_ec/GlobalPosition
5	ManageVehicleStates	1	MLC_CC	SDOAC SW	Team P	Control	ManageVehicleStates ManageVehicleStates_demo_ec ManageVehicleStates_demo_ec/ManageVehicleStates
6							

ProjectArtifactStructure   ProjectMilestoneStructure   **ArtifactStructure**   ⊕

Figure 4.66: Different artifact paths are mapped to a common artifact name in Excel using the "Artifact Structure" sheet

In Excel, as shown in Figure 4.66, in the `ArtifactPaths` column of the **Artifact Structure** sheet, all artifact paths with same artifact name are listed separated by line break or comma characters.

In case the same base measures were provided for different artifact paths for the same revision and are now mapped to a common artifact name, MQC only uses the most recent of these base measures to calculate quality for the artifact.

Artifact Mapping configuration is part of the **Project Structure source** (see *Project Structure*).

Additionally, it is possible to map artifacts in case you do not have a project structure configuration yet. To map artifact paths to artifact names, you need to define regular expression patterns.

In **Project Structures** menu you can define **Artifact Mapping Pattern** from the dialog **Create Artifact Mapping Pattern**. Here, you can define the `Search RegEx` and `Replace` values and the effective result can also be directly seen on the pageable list.

For example in Figure 4.67, the pattern is defined as: `Search RegEx:  ^(.*)_demo_.*$` , `Replace: $1`, which means the artifact name is the substring of artifact path before *_demo_*.

Figure 4.67: Define new pattern for artifact mapping

It is possible to define more than one pattern can be defined. However, the first matching pattern from the list will be applied to the artifact path. The order of the patterns can be changed in the list. The result of these patterns are also present in the export of artifact structure.

4.8.3 Quality Bins

MQC allows to adapt the Quality Bin Configuration via the **Quality Model** dialog. Bins are used to group quality aspects of a project, i.e. a quality property per artifact for a certain point in time (revision), into categories (see *Bins*). As per default, MQC uses the calculated quality value to assign a quality property to a quality bin.

You can define project specific quality bins by assigning a `Color`, `Name`, and an `Upper quality boundary` (see Figure 4.68). After creating/editing a quality bin it will be added to the list of already existing quality bins. There is no limit of quality bins you can configure for your project, but be aware there can be only one quality bin for the same upper quality boundary value.

Figure 4.68: Dialog to add or edit Quality Bin

With the given upper quality boundary values MQC is able to define the value ranges for the configured quality bins. Quality values are between 0 % and 100 %. Therefore it is required to define an upper quality boundary of 100 % for the highest quality bin. Then MQC automatically takes the upper quality boundary of the next quality bin as the lower boundary (where the quality bins will be automatically sorted according to the given quality boundary values).

For the default configuration the resulting value ranges are as follows:

- Good (green):]80%, 100%]

- Acceptable (yellow):]20%, 80%]

- Bad (red): [0%, 20%]

4.9 QUALITY CALCULATION

In MQC, functions are used to calculate

- quality properties (see *Quality Computation*)

- derived measures (see *Derived Measures*)

4.9.1 Functions in MQC

There are two types of functions in MQC:

- *Functions to calculate numerical values*, e.g. quality measurement functions where the result is a quality value between 0 and 1

- *Functions to evaluate conditional expressions*, where the result is a certain quality bin (see *Conditions*)

In both cases *Mathematical functions* as well as *Logical functions and operators* can be used to define a function.

In all functions, base and derived measures always must be defined in square brackets `[` and `]`!

Functions to calculate numerical values

These functions are used to calculate a number, e.g. a value between 0 and 1 as quality, or to combine the values of multiple similar data measures (via sum, avg, etc.) to compute derived measures.

The following examples depict how to define numerical functions, starting from the simplest case to more complex expressions.

```
[Model Decision Coverage.Ratio]
```

In this case the imported data measure is already a value between 0 and 1. It may be directly used to derive the corresponding quality property value.

```
[TestCount.Passed] / [TestCount.Total]
```

Such expressions are mainly used to compare a value against an expected value. The closer the value of `TestCount.Passed` is to the `TestCount.Total`, the better the quality.

```
(1.0 * [TestCount.Succeeded] +
 0.2 * [TestCount.Failed] +
 0.0 * [TestCount.Errors] +
```

```
 0.0 * [TestCount.Inconclusive])
/ [TestCount.Total]
```

Combining similar measures like the amount of different test results, may lead to a more detailed representation of the reality. Factors may be used to increase or reduce the impact of a certain measure. In the above example a factor of `0.2` is used for the number of failed test cases to differentiate between these tests and the not executed ones.

```
[TestCount.Total] == 0 ? 0 : [TestCount.Passed] / [TestCount.Total]

[GuidelineCount.Passed] + log(1 + [FindingCount.Passed], 2)
```

The above examples show more complex expressions using `inline if` statements respectively mathematical functions like `log(x, y)`.

Functions to evaluate conditional expressions

The result of a conditional function is either `true` or `false`. This type of functions is used to define *Conditions*.

The simplest case of such a conditional function is shown by the following example.

```
True
```

A more complex example uses different base measures (possibly from different data sources) to evaluate if the result of the condition is `true` or `false`.

```
((([Statement Coverage.Ratio] >= 0.8 and
  [TestCount.Failed] = 0 and
  [TestCount.Error] = 0) or
 ([MC/DC Coverage.Ratio] >= 0.8 and
  [TestCount.Failed] = 0 and
  [TestCount.Error] = 0))
```

4.9.2 Mathematical functions

Apart from the four basic arithmetic operations (+, −, * and /), the following functions can be used:

- Abs(x):

  Returns the absolute (positive) value of x

- Cbrt(x):

  Returns the cube root of x

- Ceil(x):

  Returns x rounded up to its nearest integer

- Cos(x):

  Returns the cosine (a value between -1 and 1) of the angle x (given in radians)

- Exp(x):

  Returns the value of e to the power of x

- Floor(x):

  Returns x rounded down to its nearest integer

- Log(x, y):

  Returns the logarithm of x with base y (if no y is given, the base is e)

- Max():

  Returns the highest value in a list of arguments

- Min():

  Returns the lowest value in a list of arguments

- Pow(x, y):

  Returns the value of x to the power of y

- Round(x):

  Returns x rounded to its nearest integer

- Sign(x):

  Returns if x is negative, null or positive

- Sin(x):

  Returns the sine (a value between -1 and 1) of the angle x (given in radians)

- Sqrt(x):

  Returns the square root of x

- Tan(x):

  Returns the tangent of x

- Trunc(x):

  Returns the integer part of x

4.9.3 Logical functions and operators

The following statements resp. operators can be used:

- Inline if

  To define conditional expressions to be used e.g. for step functions.

```
[OpenIssues.Total] = 0
    ? 1
    : [OpenIssues.Total] <= 10
          ? 0.5
          : [OpenIssues.Total] <= 100
                ? 0.2
                : 0
```

- And

  Connects two or more expressions and returns only `true` if all expressions return `true`. Can be used in `inline if` statements as well as to define bin conditions.

- Or

  Connects two or more expressions and returns `true` if at least one of the expressions returns `true`. Can be used in `inline if` statements as well as to define bin conditions.

5 SERVER

5.1 REQUIREMENTS

5.1.1 Spotfire Server 11.4 LTS

Hardware

Processor

- Minimum: 2 Cores (Intel Core i3 or equivalent), 2 GHz, 64-bit

- Recommended: 4 Cores or more (Intel Xeon E5 or equivalent), 2 GHz or higher, 64-bit

RAM

- Minimum: 8 GB

- Recommended: 16 GB or greater

Hard disk space

- Minimum: 10 GB for a small test system.

- Recommended: At least 30 GB to fit multiple deployment areas, information link and library content caching, library exports, and log files.

Software

Operating system (only 64-bit)

- Microsoft Windows Server 2019

- Microsoft Windows Server 2016

- Microsoft Windows Server 2012 R2

- Microsoft Windows Server 2012

- Red Hat Enterprise Linux 8

- Red Hat Enterprise Linux 7

- CentOS 8

- SUSE Linux Enterprise Server 15

- SUSE Linux Enterprise Server 12

Spotfire Server database

- Oracle 19c

- Oracle 18c

- Oracle 12c Release 2 (12.2.0.x)

- Oracle 12c Release 1 (12.1.0.x)

- Microsoft SQL Server 2019

- Microsoft SQL Server 2017

- Microsoft SQL Server 2016

- Microsoft SQL Server 2014

- PostgreSQL 12

- PostgreSQL 11

Authentication method

- LDAP - Microsoft Active Directory

- LDAP - The Directory Server product family (Netscape Directory Server, iPlanet Directory Server, Sun ONE Directory Server (v5), Sun Java System Directory Server (v6), Oracle Directory Server (v7)

- Kerberos v5 for Active Directory

- NTLMv2

- Microsoft Windows NT Domain

- Database table (Spotfire Server is used to both manage and store user names and passwords)

- Custom JAAS modules using username/password credentials

- X.509 client certificates

- OpenID Connect

- Custom web authentication component

Installation privileges on Windows

- Spotfire Server 11.4.x must be installed under a Windows user account with full administrative privileges.

Locale

- The server should run in an English (United States) language setting.

Web browser (for server administration/configuration pages)

- Google Chrome

- Microsoft Edge

- Mozilla Firefox

Network

- IPv4

- IPv6

Java Development Kit

- Oracle Java SE Development Kit 11 (included in installation)

- OpenJDK 11

5.1.2 Spotfire Server Node Manager 11.4 LTS

Hardware

Processor

- Minimum: 2 cores (Intel Core i3 or equivalent), 2 GHz, 64-bit

- Recommended: 6 cores or more (Intel Xeon E5 or equivalent), 2 GHz or higher, 64-bit

RAM

- Minimum: 16 GB per instance

- Recommended: 16 to ~50 GB per instance. If more than 50 GB RAM is available per instance, it is generally more beneficial to add an additional instance.

Hard disk space

- Minimum: 10 GB

- Recommended: At least two times the RAM size as available disk space.

Software

Operating system (only 64-bit)

- Microsoft Windows Server 2019

- Microsoft Windows Server 2016

- Microsoft Windows Server 2012 R2

- Microsoft Windows Server 2012

Microsoft .NET Framework

- Microsoft .NET Framework version 4.5.x (4.5.2 and higher), 4.6.x, 4.7.x, 4.8.x

Installation permissions

- Spotfire Web Player service 11.4.x must be installed under a Windows user account with full administrative privileges.

- Spotfire Automation Services Service 11.4.x must be installed under a Windows user account with full administrative privileges.

5.2 INSTALLATION

This documentation will only describe the installation on a windows server. For the installation of the server, not the node manager, on linux please read the Spotfire installation documentation. Contact support@quality-commander.com for the full documentation.

5.2.1 Prerequisites

Make sure that your system fulfills the requirements listed on the Server Requirements page.

Database server

A database server must be up and running before you can install Spotfire Server, preferably on a separate server. Setting up a database server is not part of this installation description.

The following database systems are supported by the Spotfire environment:

- MSSQL,
- Oracle or
- PostgreSQL.

Ports

Decide on the ports for the spotfire server and the node manager.

Spotfire Server

- Front-end port (default: 80, 443 for https)
- Back-end registration port (default 9080)
- Back-end communication port (default 9443)

Node Manager

- Back-end registration port (default 9080)
- Back-end communication port (default 9443)
- Services communication port (default 9501)

The ports have to be open in the firewall:

- The front-end ports have to be accessable to the public to open the web ui.
- The back-end and services ports have to be accessable by the other server (node manager <-> server)

If the node manager is running on the same host as the server - not recommended - the default ports have to be changed for the node manager registration (e.g. 10080 and 10443).

SSL Certificate

To use an encrypted HTTPs connection for the front-end an SSL certificate is required either as p12 or a jks keystore.

Additionally, the corresponding keystore password is needed.

Authentification

Knowledge about your organization's IT infrastructure is required to set up any login method other than user name and password authentication against the Spotfire database.

Installation Location

The default installation location on the server is `C:\MQC` and highly recommended.

If you want to use a different path for the installation, a script and 2 configuration files have to be edited prior to the installation.

- `configurations\AutomationServices\root\Spotfire.Dxp.Worker.Host.exe.config`
- `configurations\WebPlayer\root\Spotfire.Dxp.Worker.Host.exe.config`
- `configurations\ConfigureServer.bat`

Open the files in Notepad and replace all occurances of `C:\MQC` into the preferred alternative path.

5.2.2 Download of Installation files

Download the server installation zip file provided by MES and extract it on the server. For access to the installation files please contact mqc@model-engineers.

The following files are included in the zip archive:

- `setup-win64.exe` (Server Installation)
- `nm-setup.exe` (Node Manager Installation, see node manager chapter)
- `mqc.sdn` (Client package)
- `scripts\*` (Database Scripts)
- `configurations\*` (Configuration files for the server, services and user groups)
- `tomcat\*` (Replacement files for the tomcat webapp)

5.2.3 Database Setup

The database must be prepared for Spotfire before the server installation is run.

Please, contact your local DB administrator to create and prepare the Spotfire database and a corresponding database user.

The following permissions have to be applied to the database user: Connect, Control.

The default language of the database user has to be set to English.

Collect the following information needed for the Spotfire server configuration. This should be provided by your DB administrator:

  • Database server hostname

  • Database server port

  • Spotfire database name

  • Spotfire database user

  • Spotfire database password

The database server port depends on the chosen database system. The default ports are:

  • 1433 - MSSQL

  • 1521 - Oracle

  • 5432 - PostgreSQL

Depending on the chosen database server, the administrator may use one of the following files to create the necessary Spotfire tables:

  • `\scripts\mssql_database.sql`

  • `\scripts\oracle_database.sql`

  • `\scripts\postgres_database.sql`

5.2.4 Spotfire Server Setup

You can install the Spotfire Server files interactively on Windows, using the installation wizard.

For security and product performance reasons, it is recommended that you install Spotfire Server on a different computer than the database.

1. Double-click `setup-win64.exe` to start the installation.

2. In the installation wizard Welcome dialog, click Next.

3. In the License dialog, read the agreement, accept the terms, and then click Next.

4. In the Destination Folder dialog change the location to `C:\MQC\Server11.4`, and then click Next.

5. In the Windows Service dialog, select "Create Windows Service" and then click Next.

6. In the Spotfire Server Port dialog specify the front-end port, and then click Next.

7. In the Backend Communication Ports dialog specify the back-end ports, and then click Next.

8. In the Ready to Install dialog, click Install.

9. After the installation is complete, select "Launch the configuration tool" and then click Finish.

Verify that the ports specified in the installation are open in the windows firewall. Open the Windows Firewall Advanced Security Settings and create a new Inbound Port Rule which allows to connect via the server ports (e.g. 80, 443, 9080 and 9443).

5.2.5 Spotfire Server Configuration

Bootstrap including database connection settings

If the Configuration Tool was not opened in the last step of the installation process, double click on `C:\MQC\Server11.4\tomcat\spotfire-bin\uiconfig.bat`.

Click on "Create a new bootstrap file…" to open the dialog to set up the database connection. Select the Driver template related to your database and provide the connection information: hostname, port, identifier, username and password. Define a password for the configuration tool in the Other setting section and keep the rest of the settings as they are by default.

After that save the new bootstrap. The Connect to Database status should now be marked as green on the system status page.

Default configuration

Double-click on `configurations\ConfigureServer.bat` and input the tool password. This script imports a default configuration for server and the node services (web player and automation).

To see the configuration in the Configuration Tool, you have to close and reopen the tool, double click on `C:\MQC\Server11.4\tomcat\spotfire-bin\uiconfig.bat`.

After changes to the configuration are made, the service "TIBCO Spotfire Server 11.4.3 LTS" has to be started. Open the "Services" app to trigger the start.

Create Admin user

For the deployment of the MQC client and installation of node services, please create an admin user in the Configuration Tool.

5.2.6 Tomcat Configuration

Tomcat Deployed Applications

Copy the `tomcat\` directory of the installation zip archive to the spotfire installation `C:\MQC\Server11.4\tomcat` and replace all files.

Tomcat SSL Certificate

It is highly recommended to add an SSL certificate and use the https protocol for the webserver. After a hostname and domain is assigned to the server and an SSL certificate has been bought, the certificate can be added to the tomcat configuration.

If the server is accessable from the internet, a free letsencrypt certificate is an alternative.

https://letsencrypt.org/

The ssl certificate has to be in the pkcs12 (.p12) or java keystore (.jks) format. A typical PEM certificate (.pem, .cer, .cert) can be converted to .p12 by using openssl:

```
"C:\Program Files\OpenSSL\bin\openssl" pkcs12 -passout pass:PASSWORD -export
-in certificate.pem -inkey certificate.key -out certificate.p12 -name tomcat
-CAfile certificate-chain.pem -caname root
```

Save the .p12 certificate file in `C:\MQC\Server11.4\tomcat\certs`

Open `C:\MQC\Server11.4\tomcat\conf\server.xml` in a text editor. Add a new 443 connector configuration. Change the keystore file name to the name of the .p12 certificate. Change the keystore password to the password of the .p12 certificate.

Optionally the Connector for the default port 80 can be removed.

```
<Connector port="443"
          maxHttpHeaderSize="65536"
          connectionTimeout="30000"
          enableLookups="false"
          URIEncoding="UTF-8"
          disableUploadTimeout="true"
          server="MES Quality Commander"
          compression="on"
          compressableMimeType="text/html,text/xml,text/plain,text/css,application/
↪json,application/javascript,image/svg+xml,application/xml"
          acceptorThreadCount="2"
          keepAliveTimeout="30000"
          maxKeepAliveRequests="-1"
          maxThreads="2000"
          SSLEnabled="true"
          scheme="https"
          secure="true"
          keystoreType="PKCS12"
          keystoreFile="./certs/certificate.p12"
          keystorePass="PASSWORD"
          clientAuth="false"
          sslProtocol="TLS" />
```

Http Redirect

To use always a secured connection, an http redirect should be configured.

Open `C:\MQC\Server11.4\tomcat\conf\server.xml` in a text editor. Extend the valve configuration.

```
<Valve className="org.apache.catalina.valves.rewrite.RewriteValve" />
<Valve className="org.apache.catalina.valves.ErrorReportValve"
       showReport="false"
       showServerInfo="false" />
```

Then navigate to `C:\MQC\Server11.4\tomcat\conf\Spotfire\localhost\` and create a file `rewrite.config`. Open the newly created file in a text editor and add the following content.

```
RewriteCond %{HTTP_HOST} !^hostname\.domain$ [NC]
RewriteCond %{SERVER_PORT} ^80$ [OR]
RewriteCond %{SERVER_PORT} ^443$
RewriteRule (.*) https://hostname.domain$1 [R=301,L]


RewriteCond  %{SERVER_PORT} ^80$
RewriteRule (.*) https://hostname.domain$1 [R=301,L]
```

Take care that in the HTTP_HOST condition all special characters used in the `hostname.domain` name are escaped using a single backslash `\`.

```
Example: mqc-server.com

RewriteCond %{HTTP_HOST} !^mqc\-server\.com$ [NC]
```

5.2.7 Spotfire Server Deployment

Login to the web ui by using the created admin user.

http://hostname.domain or http://localhost

To deploy the MQC packages you need to access the `mqc.sdn`, which comes with the server installation zip file provided by MES. In case you have only extracted the archive on the server, you should login to http://localhost using a browser on the server node.

Navigate to "Deployments & Packages" and add a new area named "MQC_5-3".

On the right side, click on "Add packages" and select `mqc.sdn`. After the packages have been uploaded, validate and save the area.

Afterwards make the newly created deployment are the default area. The area created during the installation can now be removed.

5.2.8 Node Manager Setup

To deploy Spotfire services (like Spotfire Web Player, Spotfire Automation Services, the TERR service, and Spotfire Service for Python) on a node, you must first install the node manager software to manage the service.

You can install a node manager interactively on Windows, using the installation wizard.

It is recommended to install each Spotfire node on a separate host.

1. Double-click `nm-setup.exe`. You might be prompted to install Microsoft .NET Framework at this point.

2. In the installation wizard Welcome page, click Next.

3. In the License page, read the agreement, select I accept, and then click Next.

4. In the Destination Folder dialog change the location to `C:\MQC\NodeManager11.4`, and then click Next.

5. In the Node Manager Ports page specify the back-end ports. If the node manager is running on the same host as the server - not recommended - the default ports have to be changed for the node manager registration (e.g. 10080 and 10443).

6. Click Next. The Spotfire Server page opens.

7. In the Spotfire Server page, specify the details of the spotfire server: - Server name (ip or host-name.domain) - Server backend registration port - Server backend communication port (TLS)

8. In the Network Names page, select the computer names that can be used by backend trust. In general you can leave all the listed names as they are.

9. In the Ready to Install page, click Install.

10. Click Finish when done.

Verify that the ports specified in the installation are open in the windows firewall. Open the Windows Firewall Advanced Security Settings and create a new Inbound Port Rule which allows to connect via the node manager ports (e.g. 9080, 9443 and 9501).

Now the service "TIBCO Spotfire Node Manager 11.4.3 LTS" has to be started. Open the "Services" app to trigger the start.

5.2.9 Node Manager registration and service creation

Login to the web ui by using the created admin user. Navigate to "Nodes & Services" and click on the "Untrusted Nodes" tab.

The installed node manager will be listed, if installed and started correctly. Select the check box next to the new node manager and then click "Trust nodes".

Click on the "Your network" tab and the node manager is now available.

Automation Services

Select the node manager on the left side and click on "Install new service"

• Deployment area: Select the "MQC_5-3" deployment area.

• Capability: Select "Automation Services"

- Configuration: Select "AutomationServicesConfiguration"

Leave the other values on their default and click on "Install and start".

Web Player

Select the node manager on the left side and click on "Install new service"

- Deployment area: Select the "MQC_5-3" deployment area.

- Capability: Select "Web Player"

- Configuration: Select "WebPlayerConfiguration"

- Resource pool: Select "Create new" and specify a name (e.g. Web Player Resource Pool)

Leave the other values on their default and click on "Install and start".

5.3 AUTHENTICATION

5.3.1 LDAP Authentication

The MQC authentication can be integrated with an existing LDAP directory and delegates the actual authentication responsibility to configured LDAP servers.

For more information on this topic please check: https://docs.tibco.com/pub/spotfire_server/10.4.0/doc/html/TIB_sfire_server_tsas_admin_help/GUID-88F57426-44C3-4FA5-BD49-ED8A849200E3.html

The best practice is to create two groups in LDAP directory, one as "MQC Editor" and other as "MQC User", and add related users to them. The result is, that only users with valid accounts in the LDAP directory can log in to MQC.

Configuring LDAP

If the LDAP directory server has been set up to communicate over TLS, it is strongly recommended to establish a secure connection between the MQC server and the LDAP directory. If you have a self-signed certificate, set MQC Server to trust this:

- Export the certificate to a file and copy it to the MQC server in the path: *<installation dir>\jdk\lib\security*.

- Run the following command as Administrator mode in that path: *....\bin\keytool -import -file certificate.crt -keystore cacerts -alias spotfire_ldaps*

- When prompted, enter **changeit** as password.

- Verify that the certificate has been successfully added by the following command: *....\bin\keytool -list -keystore cacerts -alias spotfire_ldaps*.

- When prompted, enter **changeit** as password.

Open the configuration tool on spotfire server and follow steps to set up LDAP:

1. On the Configuration tab, select **BASIC LDAP** for **Authentication** field. The **User directory** field also switches to **LDAP**.

Figure 5.1: Set up LDAP in configuration tool.

2. In the left panel of the page, click **Authentication: LDAP**, then click New and set a name for configuration.

Figure 5.2: Create new LDAP configuration.

3. Fill in the fields of the configuration as follows:

   • Next to **Enable for**, check both the **Authentication** and **User directory**. Through this setting, MQC server stores the user account in the database, and at each login, MQC verifies the password through the LDAP directory.

   • Fill **LDAP username** and **LDAP password** fields with the user name and password of an LDAP service account with read access to Active Directory.

- Click the **LDAP Server URL** text box and enter the URL in the form ldap://server/port:, for example, ldap://myserver.example.com:389.

  If the TLS protocol is used to connect to the LDAP server, enter the URL in the form: ldaps://server/:port, for example, ldaps://myserver.example.com:636.

- Click the **Context names** text box and enter the contexts you want to synchronize.

- The **Synchronization schedule** field has default values that cause a daily synchronization between the MQC database and the LDAP directory and at each restart of the MQC server. by pressing the `Add` button you can add a new synchronization option.

Figure 5.3: Fill Basic Settings.

- This step should be performed when you create a specific group(s) in the LDAP directory to access

MQC:

– Under **Group Synchronization**, select **Yes** for **Group Synchronization** and **Filter Users by Groups** fields.

– Click the **Groups** text box and enter the groups that should have access to MQC, for example CN=MQC-Show-Access,OU=groups,DC=mesroot,DC=com.

Figure 5.4: Fill Group Synchronization.

4. Click Test connection to verify your entries.

5. Click Save configuration.

6. Restart spotfire service.

7. Log in to the MQC server as an admin user, select the **Users & Groups** menu and add users/groups from the LDAP directory to the *MQC editor* or *MQC user* group.

   To reset to the default authentication type, open the configuration tool and select **BASIC Database** in the **Authentication** field. The **User Directory** field also changes to **Database**, save the changes and restart the Spotfire service.

5.3.2 OpenID Connect Authentication

OpenID Connect (OIDC) allows a Single Sign On end user authentication to the MQC server without the necessity to manually add users (and their credentials) to the server.

Enable OAuth provider

Use a third party application like GitLab as OpenID Connect identity provider. Please, follow the instructions from the documentation of your selected application, to enable the OpenID Connect.

As Redirect URL use the Return Endpoint URL that can be copied from the OpenID Connect section in the server configuration tool (see *Configure OpenID Connect at the MQC server*). Typically, this URL starts with the

configured public address URL of your MQC server:

`http[s]://<mqc server>[:<port>]/spotfire/auth/oidc/authenticate`

Additionally, add the following scopes to your provider application:

- openid

- profile

- email.

Configure OpenID Connect at the MQC server

Login to the MQC server and open the configuration tool by double click on `C:\MQC\Server11.4\tomcat\spotfire-bin\uiconfig.bat`. You need the tool password, which was defined during the intial installation/configuration (see *Bootstrap including database connection settings*).

Figure 5.5: Add a new OpenID Connect provider via the Configuration Tool.

As shown in Figure 5.5, go to the `Configuration` tab and there select the **OpenID Connect** section.

Here you can also find the **Return Endpoint URL**, which has to be added as Redirect URL in your OpenID Connect provider (see *Enable OAuth provider*).

Select `Yes` to enable OpenID Connect authentication. Then, click on `Add new provider` and choose a name for your OpenID Connect provider.

Specify the Discovery document URL, the Client ID and the Client secret, as received when registering a client at the provider.

The Discovery document URL returns a JSON listing of for example the OpenID/OAuth endpoints, supported scopes, public keys used to sign the tokens, and other details.

If you use GitLab as OpenID Connect provider, the correct settings can be fetched from `https://gitlab.com/.well-known/openid-configuration`.

Afterwards, save the configuration and restart the server.

5.4 ADMINISTRATION

5.4.1 User Groups

Each MQC user has to be assigned to one of the following user groups, which are available at the MQC Server:

- *MQC Editors*
- *MQC Users*

MQC Editors

Users belonging to the "MQC Editors" group can create new projects (see Figure 5.6), adapt existing projects, e.g. changing the configuration, save projects or browse already stored projects from the library.

Figure 5.6: MQC Editors are able to create new projects and to load existing projects

A user belonging to the group "MQC Editors" has access to the configuration menu. For more information see *Managing Configurations*.

MQC Users

Users belonging to the user group "MQC Users" can only browse projects stored in the MQC Server library or on a network drive connected to the server (see Figure 5.7).

Figure 5.7: MQC Users are only able to load existing projects from the library or from a network drive

Once an existing project has been loaded, a viewer has no access to the `Configuration` menu and is not able to change the configuration (e.g. by adding a quality model or a project structure) or the settings of the project.

Figure 5.8: Panel functionalities available to viewer users

5.4.2 Server Library

The MQC Server library is a virtual storage of created projects (.dxp files locally) and Automation Jobs.

Every user has access to the same library, but the user permissions can be restricted to grant the access to specific folders only.

While a user belonging to the group MQC Users can only open existing projects, an MQC Editors user can also save a new project in the library.

Figure 5.9: Server library with accessible folders and saved analyses

User Permissions for Library folders

Open the MQC desktop client and connect to the server as MQC admin.

From the menu bar select **Tools > Library administration...** as shown in Figure 5.10.

This opens the library administration dialog (see Figure 5.11). Select an existing folder or click on the `New Folder` button to add a directory folder to the server library.

Click on `Edit` to set the permissions for the selected folder. In the folder permissions dialog (Figure 5.12) uncheck the **Inherit permissions from parent folder** option to be able to specify different permissions.

Use the drop down to search for specific users or user groups. Select a user respectively user group from the list and use the arrow buttons on the right-hand side to set the permissions.

If you would like to allow the access to a specific folder for selected users only, you have to remove the **Everyone** user group from the **Browse + Access** section, which has been assigned per default to newly added folders.

Users belonging to the MQC Users group (see *User Groups*) need **Browse + Access** permissions for a folder.

Users belonging to the MQC Editors group have to be assigned to **Browse + Access + Modify** to be able to store MQC projects in the selected folder.

The user group "Automation Services Users" has to be assigned to **Browse + Access + Modify** for all folders in the library. This access is required for *Update Projects on Server* to work.

5.4.3 Update Projects on Server

The MQC Server Automation Services is a web service, which is responsible to automatically execute jobs.

Figure 5.10: Open the Library Administration via the MQC toolbar.

Figure 5.11: Library administration dialog to add/delete folders and to set user permissions.

Figure 5.12: Set detailed permissions for user or user groups via the folder permissions dialog.

To schedule such a job with the purpose to periodically update MQC projects stored in the server library, the job itself must be saved in the library.

Add Update Job to Library

Open the MQC desktop client and connect to the server as MQC admin.

From the menu bar select **Tools > Automation Service job builder** as shown in Figure 5.13.

Figure 5.13: Open the Automation Service job builder via the MQC toolbar.

This opens the job builder dialog. Click on the `Add` button and choose the job **MQC: Update Projects in Library** (see Figure 5.14).

Now, the selected job must be added to the server library via **File > Save as Library Item...** chosen from the menu bar on top of the dialog. In the **Save** dialog navigate to the **Automation Jobs** folder, provide a name for your job, e.g. "UpdateProjects" and click on `Save` (see Figure 5.15).

The update job is now available on the MQC server and can be scheduled there.

Ensure Library Access for the Automation Service

Configure the *User Permissions for Library folders* to allow the "Automation Services Users" access to modify projects in all folders, including the root, of the library.

Figure 5.14: Select the job MQC: Update Projects in Library from the job builder dialog.

Figure 5.15: Add the selected job to the MQC Server Library.

Schedule Update Job

Open a browser and login to the MQC Server as MQC Admin. Then select **Automation Services** from the dropdown at the menu bar (see Figure 5.16) to open the "Automation Services" page.

Figure 5.16: Go to Automation Services page to schedule a job.

Click on the `Schedule a job` button to add a new job to the automation services. Choose the `Browse` button and navigate to the **Automation Jobs** folder in the server library. The folder should contain the update job, e.g. "UpdateProjects", which was added to the library before (see Figure 5.17).

After adding the job file, use the `Add schedule` button to define one or multiple schedules to update the projects stored in the library (see Figure 5.18).

Figure 5.19 shows a job that is periodically executed each day at 0:00 AM and at 1:30 AM. This job can be disabled to stop the automatic update of all projects stored in the server library.

To control the automatic server update per stored project, please, refer to *Keep the project up to date*.

Figure 5.17: Select the "UpdateProjects" job file from the server library.

Figure 5.18: Add one or more time schedules to the "UpdateProjects" job.

Figure 5.19: Enable or disable the configured job to update MQC projects stored in the MQC Server Library.

5.4.4 Internal Settings

MQC Working Directory

For some use cases, e.g. cloning a Git repository to be used as data source, MQC requires some space on a local drive. This also applies, when working with the Web Player. In this case, a temp directory on the server itself is used.

Per default, MQC uses the following path and automatically creates the directories, if not existing:

```
C:\MQC\Temp
```

If on your system the access to the C drive can not be granted, e.g. for security reasons, it is possible to configure another path to be used by MQC.

Open the MQC desktop client and connect to the server as MQC admin.

From the menu bar select **Tools > Administration manager...** as shown in Figure 5.20.

Figure 5.20: Open the Administration Manager via the MQC toolbar.

This opens the administration manager dialog. Choose the **Preferences** tab and then select the corresponding user group from the list on the left-hand side.

As shown in Figure 5.21, expand the **MQC** section and select **AppDataPath**.

Figure 5.21: Change the MQC working directory via the AppDataPath in the preferences of a specific user group, e.g. the MQC Editors group.

Then, click on the `Edit` button to add the directory path to be used by MQC and submit your change.

The MQC working directory path has to be changed for the user groups MQC Editors MQC Users, and Automation Services Users.

6 CONCEPT

6.1 DIMENSIONS AND STRUCTURES

The aim of MQC is to compute quality for a project based on provided data and to visualize this data in a structured way. In order to do this, MQC structures the provided data systematically. This is done with the help of artifacts, datasources, measurements, measures, variables, revisions and milestones.

Based on this structuring, MQC can offer several views on the provided data as well as the derived quality. MQC follows the terminology of ISO-25010, ISO-25012, and ISO-25022 when structuring the provided data and the computed quality. This chapter gives an overview of the terms used in MQC and their relation to ISO-25010, ISO-25012, and ISO-25022.

6.1.1 Artifacts and Artifact Structure

When data is collected within MQC, it is associated with artifacts for which that data was measured and which should be used for visualization and quality computation. Artifacts in MQC can be any object for which quantitative data is collected. Examples of artifacts are Simulink models, generated code or documents like review protocols. Figure 6.1 shows examples of artifacts. The Simulink model, the requirements document, the respective TargetLink model, and the generated code all represent artifacts.

Figure 6.1: Example of artifacts in MQC

However, artifacts can also be defined in a more general sense. They do not always need to represent real objects as work products. Artifacts can also represent virtual objects like process steps, for example.

MQC supports to adapt artifact names as they are used by data sources to the needs of the user (see *Artifacts*). Especially if different data sources use different denominations for the same artifact, a proper artifact

mapping may be used to define a common artifact name. All imported measure values collected by the data sources are automatically assigned to the new artifact name.

In case the same base measures were provided for different artifact paths for the same revision and are now mapped to a common artifact name, MQC only uses the most recent of these base measures to calculate quality for the artifact.

Artifacts can be grouped hierarchically. This structure can represent a state of the artifact itself, but it can also represent a logical structure. For instance, Simulink models can be grouped based on their functionality. However, a Simulink model can also be grouped with respect to different responsibilities (e.g. model 1 and 2 are grouped by role 1). As an example, consider several Simulink and TargetLink models that are grouped according to their implementation. See Figure 6.2 for an example of a general artifact grouping in MQC.

Figure 6.2: Artifact grouping in MQC

For details about how artifacts can be grouped in MQC, please refer to *Artifact Structures*.

Defining relations between your artifacts and certain data sources, measurements, measures or even variables can be helpful to configure your project properly. You can assign data sources, measurements or measures to certain artifacts, defining the exact data that is expected or excluded for these artifacts within your project. You can define that association between Data Sources and Artifacts via Context Categories (including measurements and measures).

For more information concerning the usage of Context Categories for your project, please refer to *Context Categories*.

6.1.2 Revisions

In real-world projects, data is collected for artifacts at different points in time. When data is imported into MQC, it is assigned to revisions. Assigning the data to revisions depends on the time stamp that is provided by the data source. If there are multiple instances of data available for the same revision, usually the latest
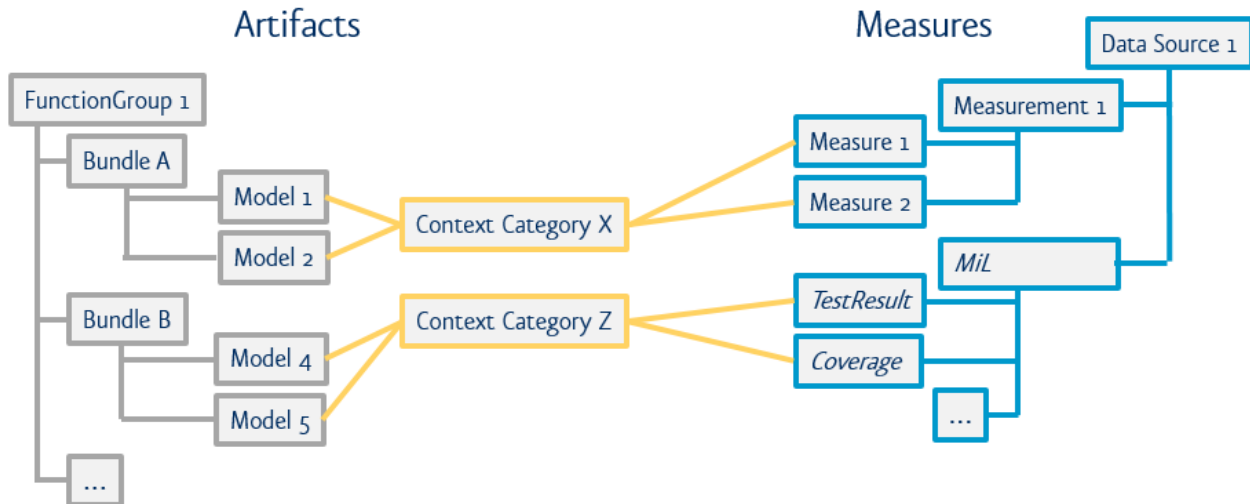
Figure 6.3: Relationship between artifacts and data sources (including measurements and measures)

data instance is used and visualized in MQC. As an example, consider a weekly revision granularity and several reports with data created on different days of a week. Then, the one with the latest time stamp is used for the weekly revision, see Figure 6.4

MQC supports multiple types of revision granularity:

- Months (revision name e.g. "2021-10"),
- CalendarWeeks (revision name e.g. "2021-W41"), and
- Days (revision name e.g. "2021-08-31").

6.1.3 Milestones

Each project can define its own milestones, where a milestone represents an important development step, i.e. the milestone time defines the end of a respective period, which starts at the end of the previous milestone and lasts until the current milestone time.

In MQC, this is realized by clustering revisions and linking the clustered revisions to a milestone defined for a certain project according to a common timeline. This means all revisions with a start time after the previous milestone and before the current milestone are linked to the current milestone. With that, data assigned to a revision is also linked to the milestone, to which the revision belongs.

The relationship between revisions and milestones is depicted in Figure 6.5.

For details on how to define milestones in MQC see *Milestones*.

6.1.4 Measures and Measurements

Data is collected from a data source via measurements. A measurement yields one or multiple measures. This is done for an artifact at a specific point in time (by a revision). Figure 6.6 shows this workflow, i.e. the measurement yields data represented by the three dimensions Measure, Artifact and Revision.
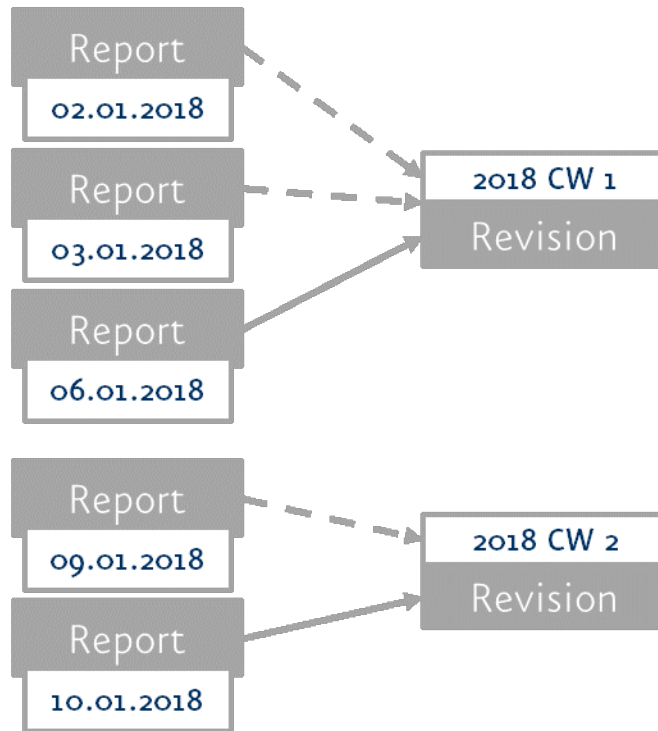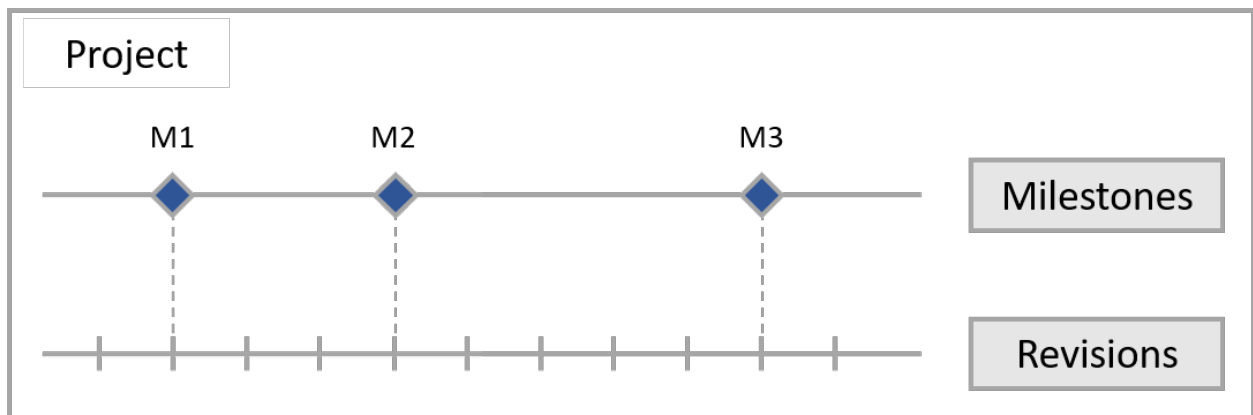
Figure 6.4: Assigning data to revisions

Figure 6.5: Relationship between project, milestones and revisions
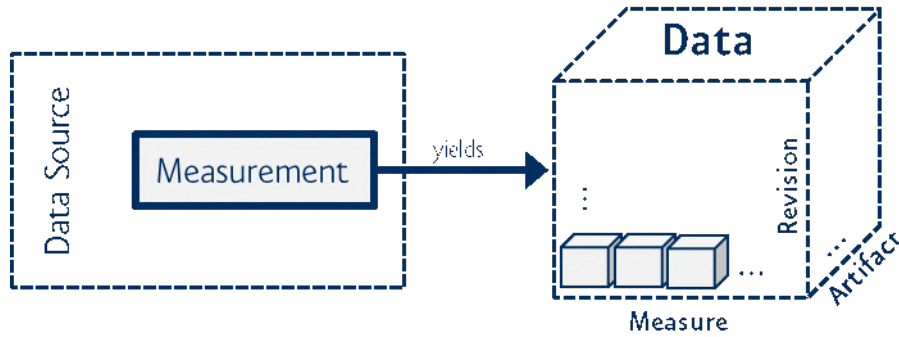
Figure 6.6: From measurement to data

In the following, we concentrate on one artifact for a fixed revision so that the dimensions of the data cube are reduced, see Figure 6.7.
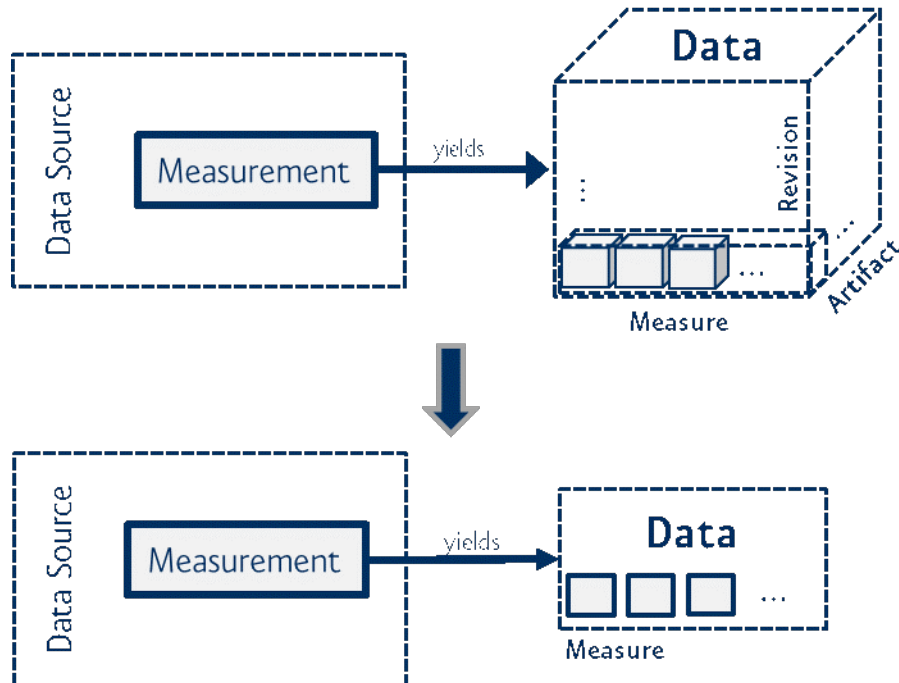

Figure 6.7: Freeze artifact and revision dimension

The measurement is the process, by which data is collected and by which the values of one or multiple measures are determined. Hence, by the execution of a measurement at a certain point in time, a value is assigned to a particular measure (see Figure 6.8).

For example, the measurement "Measuring MISRA compliance" for model A and revision X results in a guideline report containing a measure `Passed` and the assigned measure value 10.

MQC often imports data from tools directly. The source of the data is named data source. For example, the tools MXAM or MTest can be data sources.

In MQC, measures are grouped. Such a group of measures directly read from a data source is called base
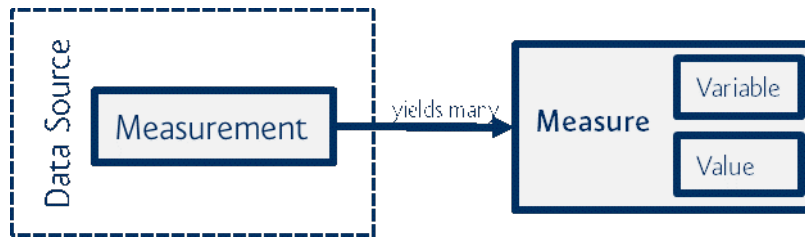
Figure 6.8: Measurement yields measures

measure and may be for example the `Local Complexity` (from the data source MXRAY). A base measure consists of at least one variable (i.e. `Good`, `Acceptable` and `Bad`), each variable has a measure value.

Please, note that there is not necessarily a one-to-one relation between measurement and data source. Rather it could be that a data source provides the same measures for different measurements. For example, test results may be related to either MiL tests or SiL tests, but nevertheless may have the same measure names (e.g. `TestCount.Failed`). In that case, measure values are assigned to the variables of a base measure group by either executing the measurement "MiL" or the measurement "SiL". See Figure 6.9 for the relationship between data source, measurements and base measures.
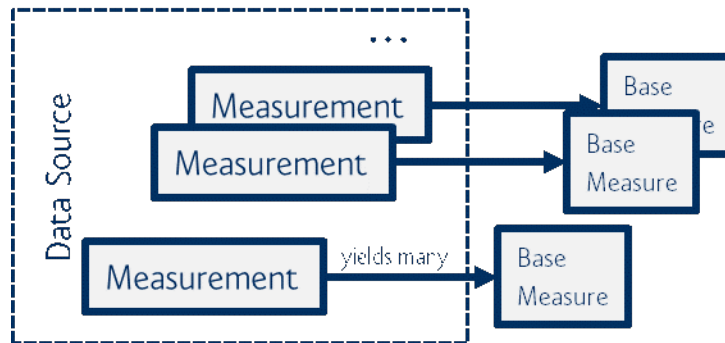

Figure 6.9: Data Source, measurements and base measures

6.1.5 Derived Measures

MQC differentiates between base measures (see *Measures and Measurements*) and derived measures. Derived measures are computed either from base measures or from other derived measures (see Figure 6.10). They are used to visualize their trend and/or to simplify the quality computation.

As an example, consider the measurement "Measuring MISRA compliance" with the base measure variables `Passed`, `Failed` and `Warning`. Here, a suitable derived measure variable `Total` could be added, which stands for the total number of all `Passed`, `Failed` and `Warning` guidelines (see Table 6.1):

Total = Passed + Failed + Warning
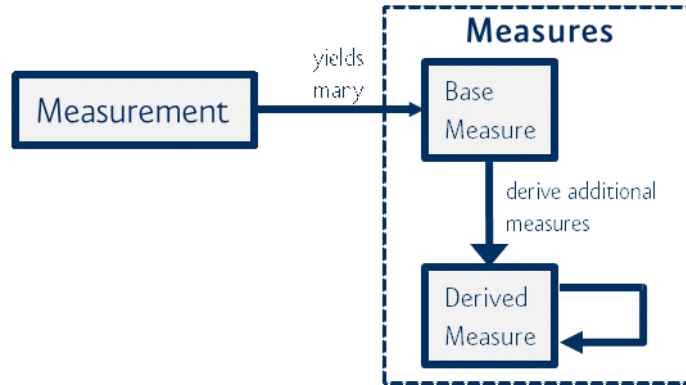
Table 6.1: Derived measure for measuring MISRA compliance

Figure 6.10: Relationship between measurements, base measures and derived measures

Measurement	Base Measure			Derived Measure
	Passed	Failed	Warn-ing	Total = Passed + Failed + Warning
Measuring MISRA Compliance	10	5	5	20

A measure function to calculate a derived measure is executed per artifact, per revision and per measurement (see *Quality Computation* for more details).

For details about how to configure derived measures in MQC, please refer to *Derived Measures*.

For details about how to define functions and available operators, please refer to *Quality Calculation*.

6.2 QUALITY

This chapter explains how MQC computes quality based on the imported data, derived measures (both are introduced in *Dimensions and Structures*) and a Quality Model. Additionally, it describes how quality is structured in MQC.

6.2.1 Quality Computation

When data is collected and imported in MQC, it is handled in terms of measures.

A measurement – that is the set of operations executed to determine values for measures – is applied and it yields a collection of measures including their measure values (variables). Either these measures can be base measures coming directly from the data source or derived measures, i.e. computed from base measures or other derived measures (see Figure 6.10).

One, two or multiple of these base and derived measures are used to calculate a quality value. This quality value is associated to a so-called quality property. Quality properties define the lowest level of computed quality. Examples of quality properties are "Guideline Compliance" or "Test Sequence Compliance".

For each quality property, a measurement function defines how to calculate the quality value by using base and derived measures.

Quality values always need to be between 0 and 1 (means, between 0% and 100%). This is ensured by MQC even if the measurement function yields a value outside these boundaries.

Alternatively, it is possible to define *Conditions*. Then, the result of a measurement function is not a numerical value. Instead, data measures are used to define conditional expressions. As a result, a specific quality bin (see *Bins*) is assigned if a condition is `true`.

The workflow from measuring the data to calculating the quality value for a quality property is shown in Figure 6.11.
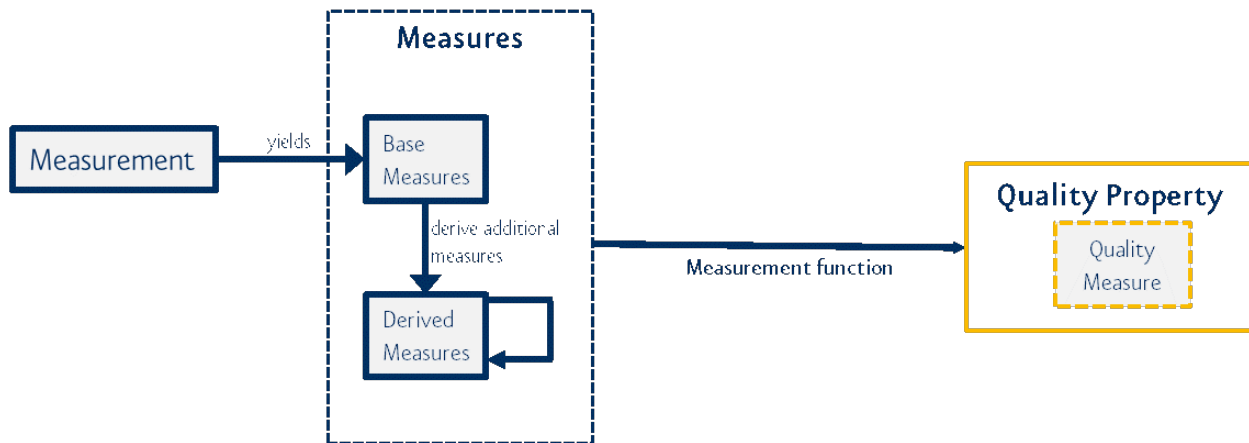

Figure 6.11: Workflow from collecting data with a measurement to calculating a quality value by a measurement function

Each measurement function is executed:

- **for each artifact**

  If a project consists of three artifacts, a measurement function to calculate e.g. the Test Sequence Compliance is executed three times based on the corresponding measures imported for each of the three artifacts.

- **for each revision**

  Per artifact, quality is re-calculated at fix points in time (revisions) during project runtime, each time using the actual measures imported for that artifact. If a project lasts e.g. for 3 weeks and the revision granularity is weekly, a quality measurement function is executed three times, once per week, for each artifact.

- **for each measurement**

  A data source may provide the same data measures from different measurements, e.g. test results may be imported for MiL and for SiL tests. If a measurement function is not explicitly defined for a specific measurement, it is executed for each measurement separately, see Figure 6.12.

Please refer to *Quality Properties* for details on how to configure quality properties and measurement functions in MQC.

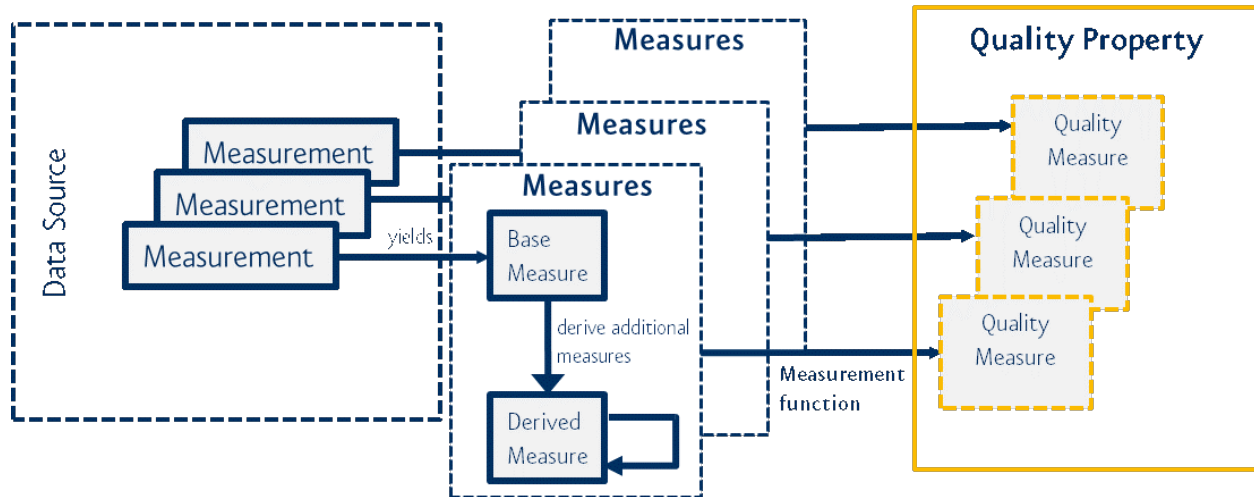For details about how to define functions and available operators, please refer to *Quality Calculation*.

Figure 6.12: Measurements from the same data source use the same measurement function for computing the quality values

6.2.2 Bins

Quality values can be mapped to quality bins. A quality bin is a kind of category that contains a set of quality values of a certain value range. Per default, MQC uses a quality bin definition as follows:

- Bad, for quality values in [0%, 20%]

- Acceptable, for quality values in]20%, 80%]

- Good, for quality values in]80%, 100%].

MQC allows to customize these categories. It is possible to configure less or more bins than MQC provides as default, to adapt the value ranges of the bins and even to choose an alternative color scheme. For more details see *Quality Bins*.

Next to the quality value, the quality bin adds an additional attribute to each quality property (see Figure 6.13).

By counting the number of elements in the bin categories, e.g. in the categories "Good", "Acceptable" and "Bad", MQC is able to provide a quality value distribution.

Conditions

Per default, quality values calculated by a measurement function are mapped to *Bins*. For example, all values in a range from 80% to 100% may be assigned to the category "Good", which is indicated by a green color in all quality visualizations.

Alternatively, it is possible to skip the calculation of quality values and directly assign a quality bin if the imported measures fulfill a specific condition.

Such a bin condition could be for example:

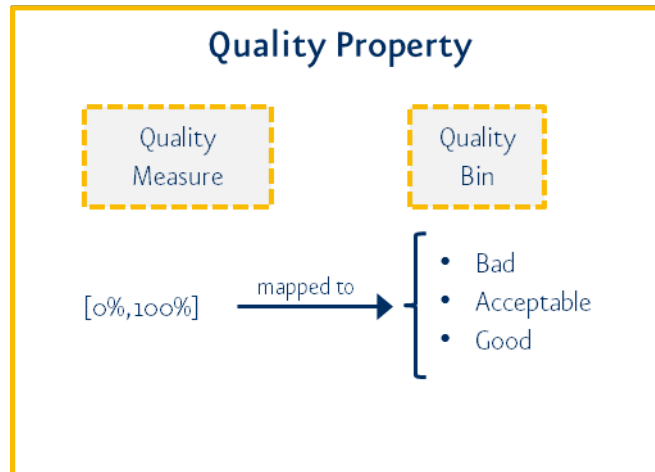- in case there are no issues at all -> assign quality bin "Good"

Figure 6.13: Quality bin as additional attribute in quality property

- in case the number of issues is higher than 0 but less than 50 -> assign quality bin "Acceptable"

- in case the number of issues is higher then 50 -> assign quality bin "Bad"

Please refer to *Bin Conditions* for details on how to configure quality bin conditions in MQC.

6.2.3 Structure

Quality properties are the computable atomic elements of the quality model with values between 0 and 1. By an adjustable aggregation method, an overall quality value is calculated.

MQC allows the user to implement a general quality model that complies with ISO-25010. The user can define quality properties and a quality structure according to the project and process needs. Figure 6.14 shows an example structure with two levels, "Characteristics" and "Subcharacteristics", of a quality model.
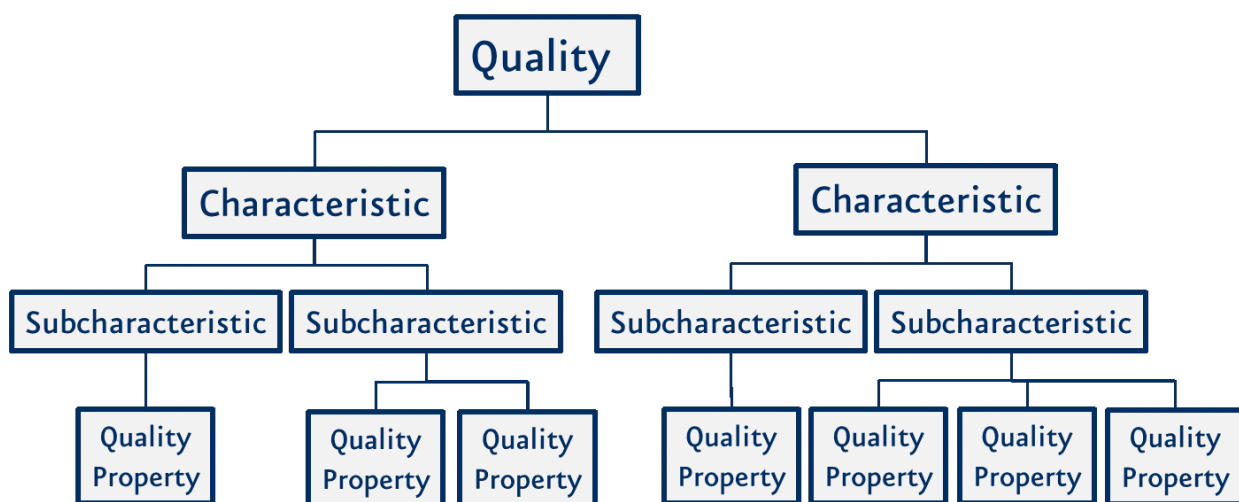

Figure 6.14: Quality Model – Concept (ISO 25010, p. 2)

6.2.4 Aggregation

Looking at the three dimensions 'Measure', 'Artifact' and 'Revision' all the quality values build a quality cube, see Figure 6.15. Each quality property represents a slice.
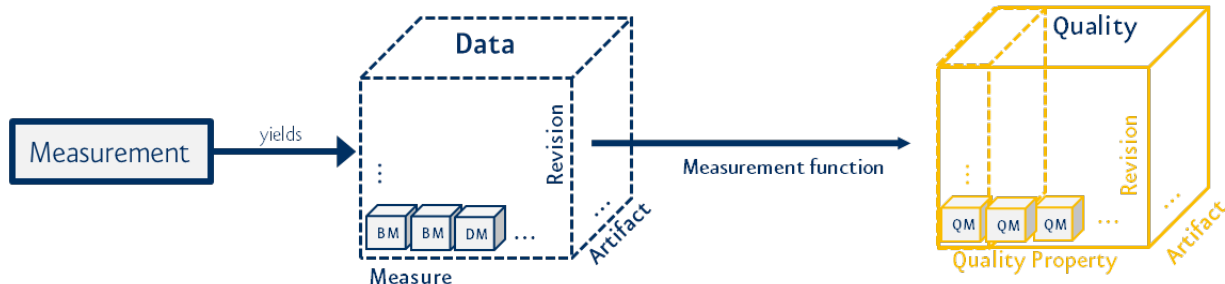

Figure 6.15: From data cube to quality cube

Based on the quality value for each quality property per revision and artifact, a first aggregation step is performed. For every single quality property, the corresponding quality values for all artifacts are aggregated by calculating the average (see Figure 6.16).
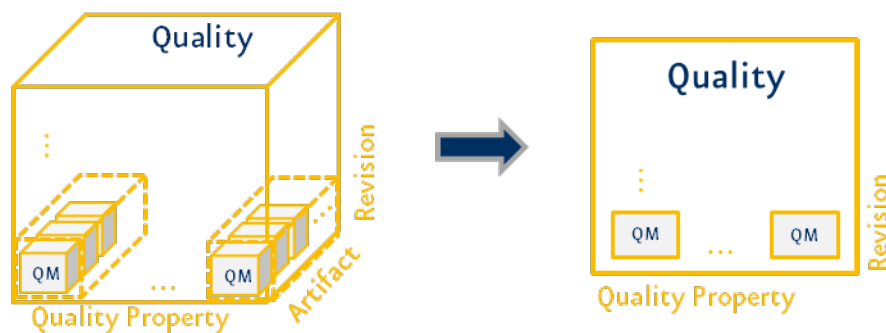

Figure 6.16: First aggregation step with respect to all artifacts

This is the basis for further aggregation in the quality structure.

According to the Quality Model defined by the user, the quality properties are used for the quality aggregation into their parent structure elements and the overall quality. Each of these aggregations is done using the associated quality properties directly as a base. There is no technical hierarchy in the quality aggregation calculation.

All aggregations currently use the (weighted) average calculation as aggregation function, e.g. the average of all quality property measures for a certain structure-element is calculated to gain the quality value for this particular structure-element and so on.

The structure does not influence the overall quality. It can define different structure levels and elements to group and better visualize quality properties, especially for large projects.

6.3 DATA PROPAGATION

Data imported into MQC is assigned to a revision depending on the point in time the data was collected, i.e. the report has been created (see *Revisions*).

Missing data in terms of not yet available respectively not yet imported into MQC, leads to missing quality, which has an impact on the overall quality of the project, because per default missing quality is treated as 0% (see *Aggregation*).

Therefore, data that was available in previous revisions can be used in later revisions as well until it is replaced with data from a newly created report, e.g. after a test re-execution.

Using data propagation, MQC is able to calculate and visualize certain quality metrics even for those revisions, where the data used to calculate the particular quality metric has not been re-collected respectively re-imported.

Data may become invalid, if for instance the artifact changes (new model version), but the data is not re-collected.

Therefore the user must decide carefully if and which data to propagate.

In any case, data is replaced if more up-to-date data - new data collected for the same object - is loaded into MQC.

Details about how to switch on or off data propagation in MQC can be found in *Propagation of data*.

6.4 CONTEXT CATEGORIES

By default, MQC expects each measure configured in the quality model (see *Base Measures*) to be provided for each artifact defined in the project structure configuration (see *Artifact Structures*) respectively for each imported artifact if no project structure has been applied so far.

Expected data that is not imported for a certain artifact for a certain revision is stated as "missing" to inform the user about e.g. not yet completed tasks.

However, this may produce lots of missing data, especially for the case that some measures may never be provided for a particular artifact.

By using context categories you can define artifact-relevant data, thus configure, which measures (data sources, measurements, and base measures) are expected for an artifact. In that way, data previously stated as "missing" - and probably resulting in bad quality - is treated as "excluded" data. This means it is ignored in availability and quality calculation.

For details about how to define context categories see *Context Categories*. How to assign context categories to artifacts is described in *Artifacts*.

Context categories will *NOT* automatically be applied to an MQC project after loading proper project structure and quality model. Per default the usage of context categories is disabled, hence, all data is expected and shown for all artifacts.

Data and quality excluded by context categories are shown as white areas in visualizations.

For details about how to switch on or off context categories see *Context Categories*.

6.5 TARGET VALUES

Targets in MQC can be used to compare the current progress of a project against an expected respectively planned progress. If for instance for a particular milestone in a very early phase of a project the structural coverage is not yet expected to have a quality of 100%, this can be reflected by defining a proper target.

By choosing then *Relative* scope of quality assessment (see *Quality Assessment Scope*), the so-called relative quality, which is based on the configured targets, is shown in the visualizations.

Additionally, configured targets can be shown in trend and status visualizations on custom pages (see *Custom Pages*). Details about how to switch on or off targets on custom pages can be found in *Target values in visualizations (Custom pages)*.

6.5.1 Calculation of Target Values per Revision

Target values are defined for milestones, which means it is intended to reach a specific target value at the end of the corresponding milestone.

Because of the fact that data is assigned to and quality is computed for revisions, MQC calculates all the target values for all revisions belonging to a specific milestone based on its due date. This is done using a linear interpolation between the previous and the next configured milestone target value (see Figure 6.17).

Targets always start with a value of 0 (project start date).

After the last milestone with configured target value, MQC keeps the last target value.

For a comprehensive target value calculation, a proper milestone configuration is needed. Each milestone must contain a valid milestone start and due date. For details see *Milestones*.

6.6 ANNOTATIONS

MQC allows you to annotate the quality of the quality properties and artifacts in your project. This functionality will enable you to add a description to justify the observed quality for the duration of your project. This description/comment lets other users know why a specific quality property is performing poorly. In this way, the justification is documented more transparently.

MQC objectively calculates the quality of your project. This can be configured in Quality Model, Project Structure, Target Values, and Context Categories, etc.

You could use annotations to document deviations from expected values for special, real-world cases. Additionally, you can use annotations to change the quality value or bin if they need to be treated differently than the one calculated.

Annotations should be used to justify very specific and local deviations, especially if such a deviation applies to a specific point in time only. Particularly, annotations should not be used to cover one of the following cases.
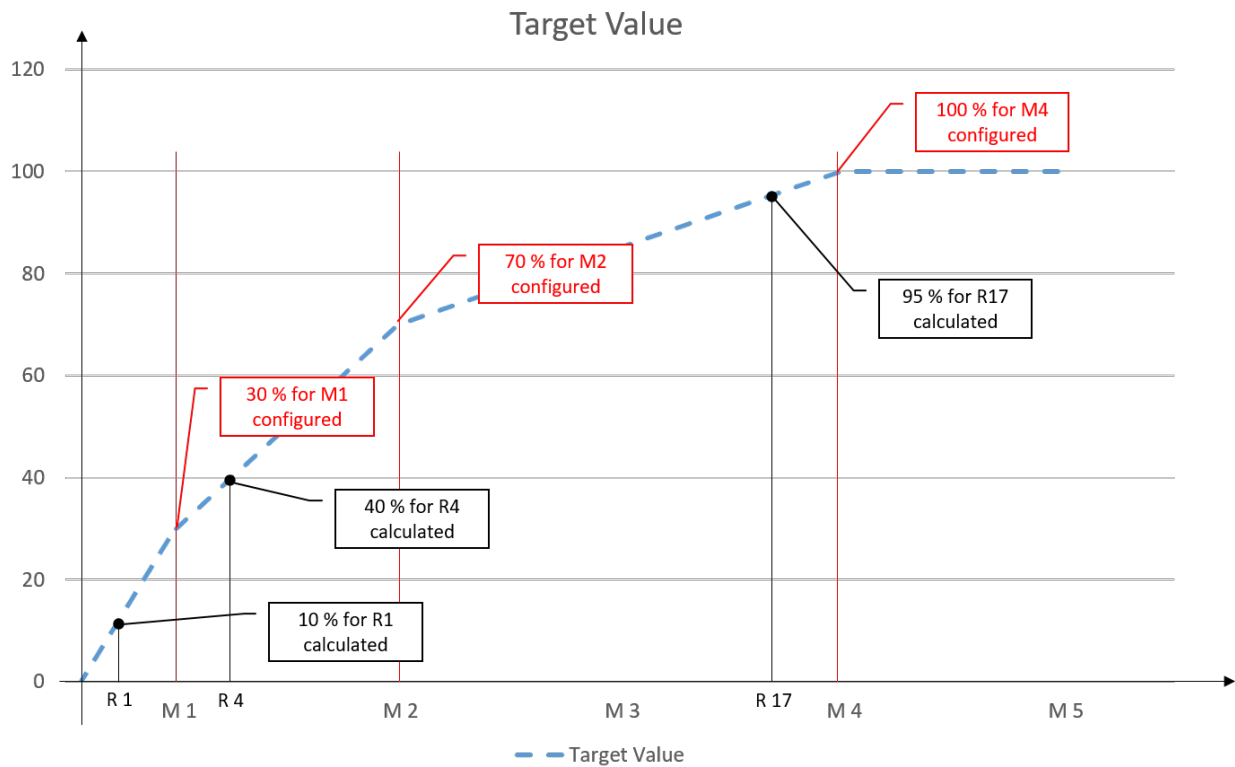
Figure 6.17: Target values configured for milestones M1, M2 and M4, linearly calculated target values for revisions assigned to milestones

- If you don't expect any data measure and don't want MQC to calculate quality for a particular artifact respectively model, this artifact should be excluded from the project using a proper Project Structure configuration (see *Artifact Structures*).

- If you don't want to track a certain quality for a particular artifact, you should use context categories to exclude the underlying data for that particular artifact (see *Context Categories*). If data is excluded for an artifact, also the quality resulting out of this data will not be calculated for that artifact.

- If the measured data values, that are loaded into MQC, objectively imply a certain quality, but the resulting quality value shown in MQC does not reflect that, the corresponding quality measurement function should be adapted accordingly (see *Quality Properties*).

- Define targets for quality values (see *Target Values*) rather than using annotations as justification. By using targets, you may specify that the relative quality is 100% if the configured target was reached or even exceeded, even if the calculated absolute quality does not reflect that at all, but meets the expectations for that point in time.

MQC displays applied annotations:

- as an "A"-Indicator inside the quality matrix (cells with annotated quality show an "A" in addition)

- as an "A"-Indicator inside the Heatmap (cells with annotated quality show an "A" in addition)

- within the Tooltip for an annotated quality point (title, description and the change of quality, if defined, will be displayed)

- in the KPIs (the KPI value shows the annotated quality, the comparative value shows the "Non-Annotated Quality").

The possibility to add annotation source(s) allows you to import annotations that are not fitting to your project. These annotations are considered unmatched and are displayed.

After loading, you can edit these unmatched annotations to make them applicable to your current project (see *Annotations*).

6.7 ACTIONS

MQC helps the user to identify the next steps to resolve quality deficits by recommending actions, where an action is a specific task which can improve one or more quality properties for a specific artifact.

Each action gets a priority according to the corresponding quality property measure value (see section *Priority Groups*). These priorities are used to create a sorted list of actions per revision.

6.7.1 Definition of Actions

Actions to improve the quality properties may be defined in the quality model (see *Quality Model*). Here, actions by default have a linear relationship to the related quality properties. For example, if a quality property has 80% quality then the respective action is assigned a priority of 20% and further assigned a priority group.

Besides the possibility to define a one-to-one relation between actions and quality properties, which means, one action directly improves a specific quality property, the following is possible, too:

- **one action** assigned to **multiple quality properties**

  For example, an action "Derive further test sequences from uncovered requirements" may improve a quality property "Testable Requirements with Test Sequences" as well as the "Model Condition Coverage".

- **multiple different actions** assigned to **single quality property**

  For example, a quality property "Testable Requirements with Test Sequences" may be improved by deriving further test sequences from the requirements (see above), but also by linking already existing but not yet linked test sequences to requirements (action "Link test sequences to (covered) requirements").

In case the quality model contains a quality property without any assigned action, MQC assigns a default action "Improve ..." followed by the name of the quality property.

Additionally, if the quality property does not have a calculated measure value for a revision then the respective action is replaced by another default action, "Start evaluation ..." followed by the name of a quality property. This means that, the data to calculate that quality property has not yet been loaded into MQC.

6.7.2 Priority Groups

To create a sorted list of actions for multiple artifacts, each action gets a priority based on the corresponding quality property measure value.

MQC uses the following categories:

- Very High
- High
- Moderate
- Low
- Very Low

An action with a priority "Very High" has be to executed in any case, whereas the one with priority "Very Low" can be postponed until all other more urgent actions have been performed.

For example, if a quality property has a value of 5 % for a specific artifact, the quality must be improved, therefore the corresponding action gets a "Very High" priority. If the calculated quality property measure value is already 90 %, it still can be improved, but there may be other tasks (worse quality) to be executed before, and the corresponding action gets a priority "Very Low".

These priorities can be influenced by defining specific artifact weights and quality property weights. By that, an action for a certain artifact will get a higher priority, if the artifact is more important than other artifacts.

Assigned priorities only define, if an action should be executed before another action based on the fact that one quality property has a relatively bad quality value than another.

A given priority does not say anything about the improvement impact of an action on the overall quality!

Contact

Mail:

> Model Engineering Solutions GmbH

> Waldenserstraße 2-4

> 10551 Berlin

> Germany

Email: info@model-engineers.com

Tel.: +49 (0) 30 2091 6463 0

Fax: +49 (0) 30 2091 6463 33

Web: https://model-engineers.com/

TECHNICAL support

Web: https://model-engineers.com/mqc.html

Email: mqc@model-engineers.com

Tel.: +49 (0) 30 2091 6463 50

Copyright Note

THIRD-PARTY PRODUCTS

MATLAB®, Simulink®, Stateflow® and Embedded Coder® are registered trademarks of The MathWorks, Inc. TargetLink® is a registered trademark of dSPACE GmbH. Windows®, Excel® and Visual Studio® are registered trademarks of the Microsoft Corporation. Testwell CTC++ is a trademark of Verifysoft Technology GmbH.