CAST Engineering Dashboard - Information - How to compute the metric grade at module level

Purpose

The page describes the computation of the metric grade at module level. Computation need to be done on the central Database. Note that if the grade is shown as 'N/A' in the Dashboard, this means that the diagnostic has not been calculated. This guide does not cover this case. Follow the guide CAST Engineering Dashboard - Quality rule - Missing Quality Rule instead. This computation can be useful when comparing two snapshot results or when the metric grade at system/application/module level is not as expected (0, 4 or any other).

Applicable in CAST Version

Release	Yes/No
8.3.x	✓
8.2.x	✓
8.1.x	✓
8.0.x	✓
7.3.x	✓
7.2.x	✓
7.0.x	✓

Applicable RDBMS

RDBMS	Yes/No
Oracle Server	✓
Microsoft SQL Server	✓
CSS2	✓
CSS1	✓

Details

If you are comparing grades, do the following twice for each snapshot. Otherwise, if you are validating the grade value, you will do the following only once for your snapshot

Factor impacting the metric grade computation at module level

The factors impacting the metric grade computation at module level are

1. The module content
2. Transformation Thresholds
3. The parameters of metric
4. The value of the option multiple values per object
5. Aggregation weight of different sub criteria and diagnostic based metric
6. The Critical criteria

Compute metric grade

If you are comparing grades between the 2 snapshots computation, check, as first step, if the used total and/or detail procedures has been modified between the 2 snapshots computation. This is particularly relevant when the 2 snapshots are not computed with same CAST version: A migration has been performed between the 2 snapshots computation. You can check the section Functional information > Changes in the metrics and diagnostics results.

The diagnostic based metric

1. The module content: The metric grade is computed according to total and successful checks objects identified on local site
    a. Get the number of total objects using following query:

    ```
    SELECT DML.OBJECT_ID, DMR.METRIC_VALUE_INDEX, DMR.SNAPSHOT_ID , sum(DMR.METRIC_NUM_VALUE)
    FROM <CB_NAME>.DSS_METRIC_RESULTS DMR, <CB_NAME>.DSS_MODULE_LINKS DML
    WHERE DML.OBJECT_TYPE_ID = 20000 and DML.OBJECT_ID = <MODULE_ID>
    and DML.MODULE_ID = DMR.OBJECT_ID and DML.SNAPSHOT_ID = DMR.SNAPSHOT_ID
    and DMR.METRIC_ID = <DIAG_ID> and METRIC_VALUE_INDEX = 2
    group by DML.OBJECT_ID, DMR.METRIC_VALUE_INDEX, DMR.SNAPSHOT_ID
    ```

    b. Get the number of failed checks using following query:

    ```
    SELECT DML.OBJECT_ID, DMR.METRIC_VALUE_INDEX, DMR.SNAPSHOT_ID , sum(DMR.METRIC_NUM_VALUE)
    FROM <CB_NAME>.DSS_METRIC_RESULTS DMR, <CB_NAME>.DSS_MODULE_LINKS DML
    WHERE DML.OBJECT_TYPE_ID = 20000 and DML.OBJECT_ID = <MODULE_ID>
    and DML.MODULE_ID = DMR.OBJECT_ID and DML.SNAPSHOT_ID = DMR.SNAPSHOT_ID
    and DMR.METRIC_ID = <DIAG_ID> and METRIC_VALUE_INDEX = 1
    group by DML.OBJECT_ID, DMR.METRIC_VALUE_INDEX, DMR.SNAPSHOT_ID
    ```

    c. Number of successful checks = number of total objects - number of failed checks
    d. Compute the compliance ratio = Number of Successful Checks/Number of total objects
2. Get parameters of the diagnostic using following query

    ```
    SELECT DMHP.SNAPSHOT_ID,
            PARAM_NUM_VALUE,
            PARAM_CHAR_VALUE
    FROM         <CB_NAME>.DSS_METRIC_HISTO_PARAMS DMHP,
            <CB_NAME>.DSS_METRIC_HISTO_TREE DMHT
    WHERE DMHT.METRIC_PARENT_ID = <DIAG_ID>
            AND DMHT.METRIC_ID = DMHP.METRIC_ID
            AND DMHT.SNAPSHOT_ID = DMHP.SNAPSHOT_ID
    ```

3. Get the value of the option multiple values per object using following query

    ```
    SELECT METRIC_AGGREGATE_OPERATOR,
            METRIC_VALUE_INDEX
    FROM <CB_NAME>.DSS_METRIC_VALUE_TYPES
    WHERE METRIC_ID = <DIAG_ID>
    AND METRIC_VALUE_INDEX = 1
    ```

4. Get the value of the transformation Thresholds stored in the central base table using following query:

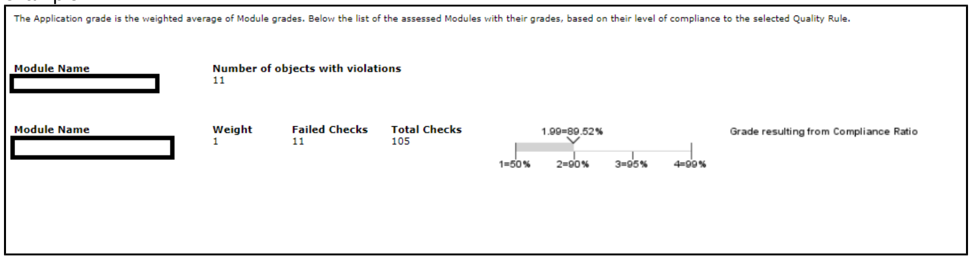
```
Select SNAPSHOT_ID, METRIC_ID, STATUS, THRESHOLD_1, THRESHOLD_2, THRESHOLD_3, THRESHOLD_4
from <CB_NAME>.DSS_METRIC_HISTO_THRESHOLDS
where SNAPSHOT_ID=<ID_Of_the_Snapshot> and METRIC_ID =<METRIC_ID>
```

Using the compliance ratio and the thresholds, compute the metric grade as described below:

The "Grade" of each Metric is the result of a mapping between the "Metric" measure and a decimal value ranging from '1' to '4' using thresholds. Default thresholds being:

    - More than 99% compliance required to attain a grade of '4.00'
    - More than 90% compliance required to attain a grade of at least '3.00'
    - More than 70% compliance required to attain a grade of at least '2.00'
    - More than 30% compliance required to attain a grade of at least '1.00

- example:



The Application grade is the weighted average of Module grades. Below the list of the assessed Modules with their grades, based on their level of compliance to the selected Quality Rule.

Module Name	Number of objects with violations
[redacted]	11

Module Name	Weight	Failed Checks	Total Checks
[redacted]	1	11	105

-

Technical and business criteria

1. Get the aggregation weight of different sub criteria and diagnostic based metric

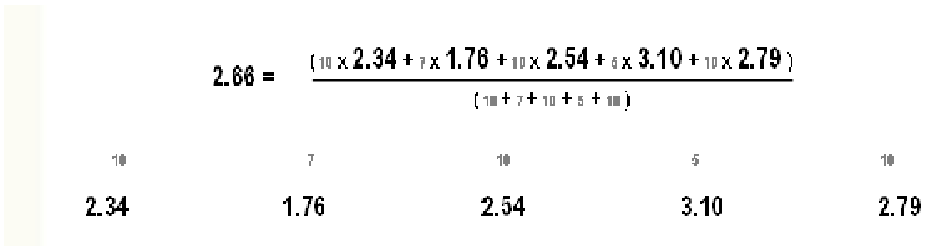
```
Select SNAPSHOT_ID, METRIC_PARENT_ID, METRIC_ID, METRIC_INDEX, METRIC_TYPE, AGGREGATE_WEIGHT ,
METRIC_CRITICAL and
From <CB_NAME>."DSS_METRIC_HISTO_TREE"
where SNAPSHOT_ID=<ID_Of_the_Snapshot> and METRIC_PARENT_ID=<METRIC_ID_OF_SCREENSHOT>
```

2. Get the Critical criteria

```
Select SNAPSHOT_ID, METRIC_PARENT_ID, METRIC_ID, METRIC_INDEX, METRIC_TYPE, AGGREGATE_WEIGHT ,
METRIC_CRITICAL
From <CB_NAME>.DSS_METRIC_HISTO_TREE" where SNAPSHOT_ID=<ID_Of_the_Snapshot> and METRIC_CRITICAL=1 and
METRIC_PARENT_ID=<METRIC_ID_OF_SCREENSHOT>
```

3. Get the metric grades of subcomponents metrics: You can get value by reading them directly in the dashboard Compute the metric using the value input identified above: If no Critical contribution metrics are used, the metric grade is the weighted average of subcomponents metrics' grades. Following an example

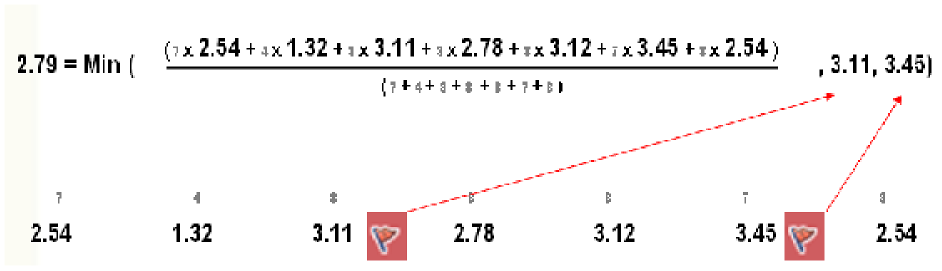
Subcomponent metric grade	weight
2.34	10
1.76	7
2.54	10
3.10	5
2.79	10



$$2.66 = \frac{(10 \times 2.34 + 7 \times 1.76 + 10 \times 2.54 + 5 \times 3.10 + 10 \times 2.79)}{(10 + 7 + 10 + 5 + 10)}$$

The resulting metric grade= 2.50
If critical contribution metrics are used, The metric grade= min (the weighted average of subcomponents metrics' grades, min (metric grade of sub component critical metrics). Follw an example

subcomponent metric grade	weight	Critical contribution
2.54	7	NO
1.32	4	NO
3.11	8	YES
2.78	8	NO
3.12	8	NO
3.45	7	YES
2.54	8	NO



$$2.79 = \text{Min}\left(\frac{(7 \times 2.54 + 4 \times 1.32 + 8 \times 3.11 + 8 \times 2.78 + 8 \times 3.12 + 7 \times 3.45 + 8 \times 2.54)}{(7 + 4 + 8 + 8 + 8 + 7 + 8)}, 3.11, 3.45\right)$$

The resulting grade:

Distribution-based

Each category is processed like a diagnostic-based metric: threshold to turn the percentage that the category represents into a grade. Distribution grade is then the minimum value among the categories' grades. When comparing the grades between 2 snapshots, you have to compare the size of each category. Any difference for one (or more) category leads to different metric grade for distribution metric. Example In snapshot 2, the size of each category of the SQL Complexity distribution is provided in the next table:

Count	%	
Low SQL Complexity Artifacts	92	0
Moderate SQL Complexity Artifacts	0	0
High SQL Complexity Artifacts	0	0
Very High SQL Complexity Artifacts	0	0
Total	92	100

In snapshot 1, the size of each category of the SQL Complexity distribution is provide on the next table:

Count	%	
Low SQL Complexity Artifacts	121	84.03
Moderate SQL Complexity Artifacts	23	15.97
High SQL Complexity Artifacts	0	0
Very High SQL Complexity Artifacts	0	0
Total	144	100

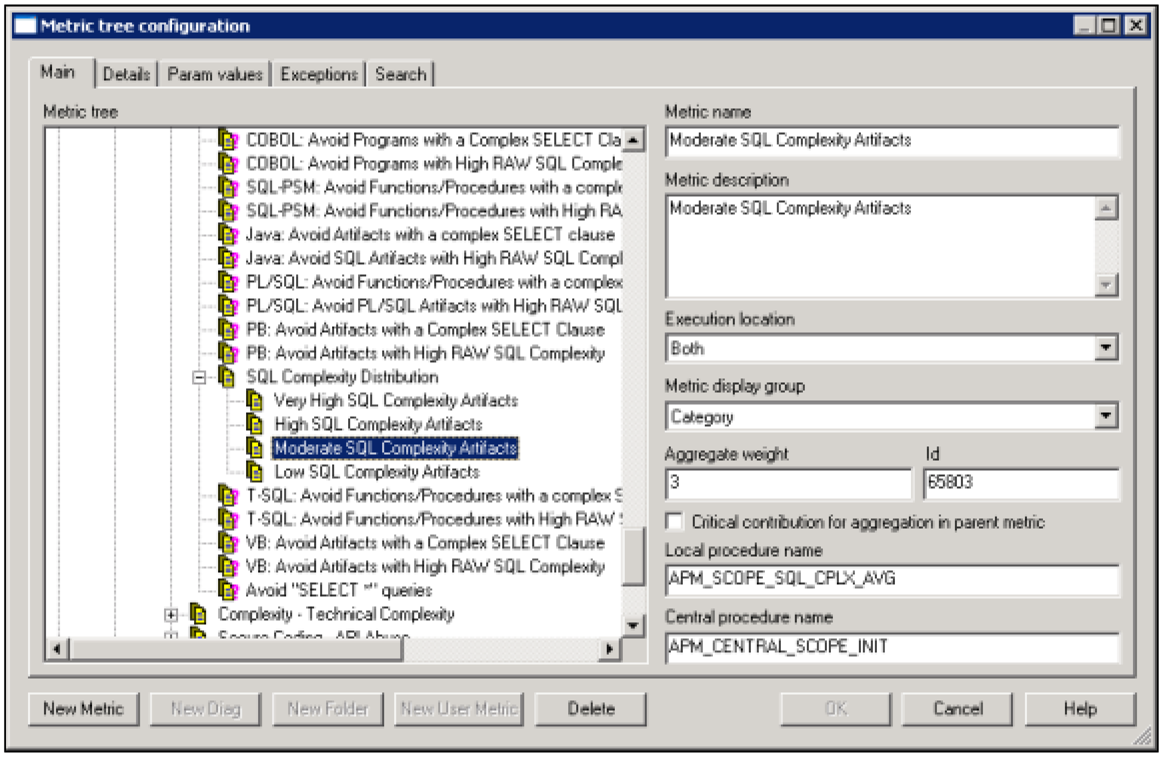
The modification of the size of the 2 categories 'Low SQL Complexity Artifacts' and 'Moderate SQL Complexity Artifacts' between snapshots 1 and 2 leads to different metric grades for 'SQL Complexity distribution' in the 2 snapshots.

The population of each category is computed by the procedure provided in *Local Procedure name *field in the Main tab of the metric tree configuration window of ADGAdmin.

The Factor impacting the size of the populations are the content of the KB and used thresholds. The used thresholds are not stored on central site.

If you have to explain reason leading to have different category size in both snapshots, **You should ask for the knowledge base(s) used for each snapshot computation**. Having the KBS, you should proceed as follow

1. Get list of objects used to compute the population size of the identified category by executing the Local Procedure of the associated category on the KB involved for the snapshot computation
2. Compare the 2 lists and identify the difference
3. Identify the reason leading to this difference (modification the used thresholds, modification of the code source, the modification of the job configuration, fixed bug ....)
4. If you cannot identify the reason, transmit to the foundation team.



Metric tree

- COBOL: Avoid Programs with a Complex SELECT Cla
- COBOL: Avoid Programs with High RAW SQL Comple
- SQL-PSM: Avoid Functions/Procedures with a compl
- SQL-PSM: Avoid Functions/Procedures with High RA
- Java: Avoid Artifacts with a complex SELECT clause
- Java: Avoid SQL Artifacts with High RAW SQL Compl
- PL/SQL: Avoid Functions/Procedures with a complex
- PL/SQL: Avoid PL/SQL Artifacts with High RAW SQL
- PB: Avoid Artifacts with a Complex SELECT Clause
- PB: Avoid Artifacts with High RAW SQL Complexity
- SQL Complexity Distribution
  - Very High SQL Complexity Artifacts
  - High SQL Complexity Artifacts
  - Moderate SQL Complexity Artifacts
  - Low SQL Complexity Artifacts
- T-SQL: Avoid Functions/Procedures with a complex S
- T-SQL: Avoid Functions/Procedures with High RAW
- VB: Avoid Artifacts with a Complex SELECT Clause
- VB: Avoid Artifacts with High RAW SQL Complexity
- Avoid "SELECT *" queries
- Complexity - Technical Complexity

Metric name: Moderate SQL Complexity Artifacts

Metric description: Moderate SQL Complexity Artifacts

Execution location: Both

Metric display group: Category

Aggregate weight: 3

Id: 65803

[ ] Critical contribution for aggregation in parent metric

Local procedure name: APM_SCOPE_SQL_CPLX_AVG

Central procedure name: APM_CENTRAL_SCOPE_INIT

Example, the Moderate SQL Complexity Artifacts, the used procedure is APM_SCOPE_SQL_CPLX_AVG

The used thresholds are provided in the script of the procedure its self Begin Select PARAM_NUM_VALUE Into **L_AVERAGE_THRESHOLD** From DSS_METRIC_PARAM_VALUES Where METRIC_ID = I_METRIC_ID And PARAM_INDEX= 1;
Select PARAM_NUM_VALUE Into **L_HIGH_THRESHOLD** From DSS_METRIC_PARAM_VALUES Where METRIC_ID = I_METRIC_ID And PARAM_INDEX= 2;
ERRORCODE := APM_SCOPE_SQL_CPLX_MIN_MAX(I_SNAPSHOT_ID, I_METRIC_PARENT_ID, I_METRIC_ID, I_METRIC_CHILD_ID, **L_AVERAGE_THRESHOLD**, **L_HIGH_THRESHOLD**);
Return ERRORCODE; End APM_SCOPE_SQL_CPLX_AVG;

The used thresholds are **L_AVERAGE_THRESHOLD** and **L_HIGH_THRESHOLD**

Notes/comments