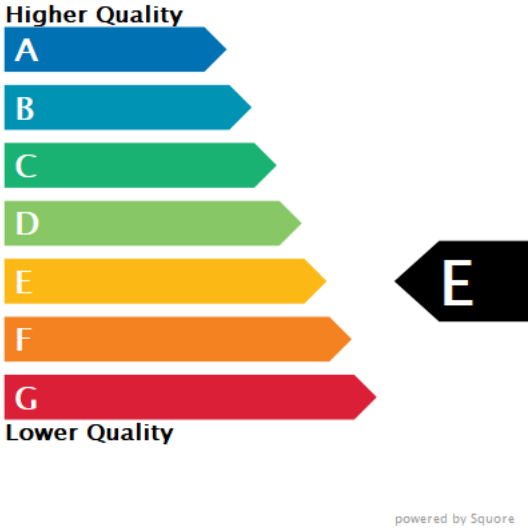
SQUORE

Evaluation Report

Project : Earth



Author : demo

Model : software_analytics

Version : Current (V7)

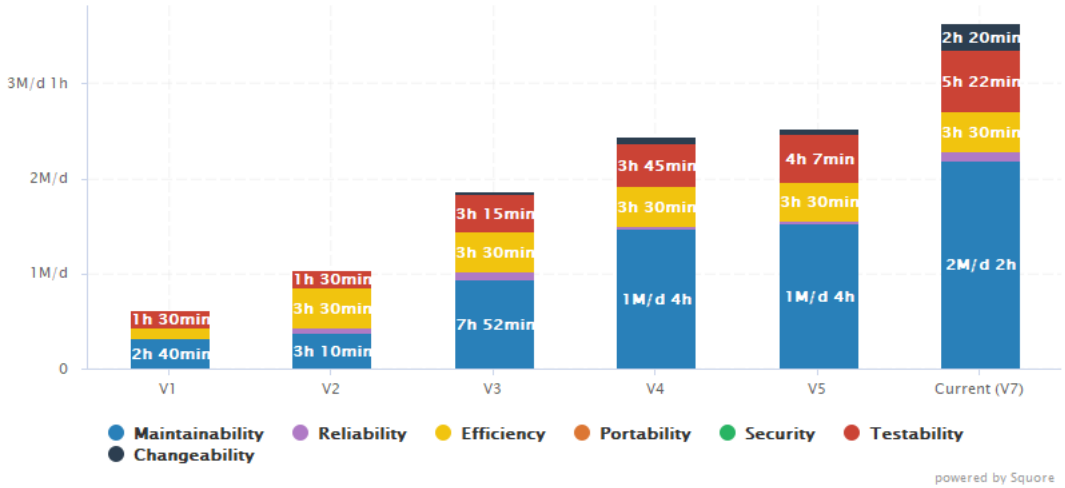
Version date : 2018.12.13 02:00:00 CET

View Squore dashboard

Name	Position	Date of Approval	Signature
General Comment:			

I. Quality Overview

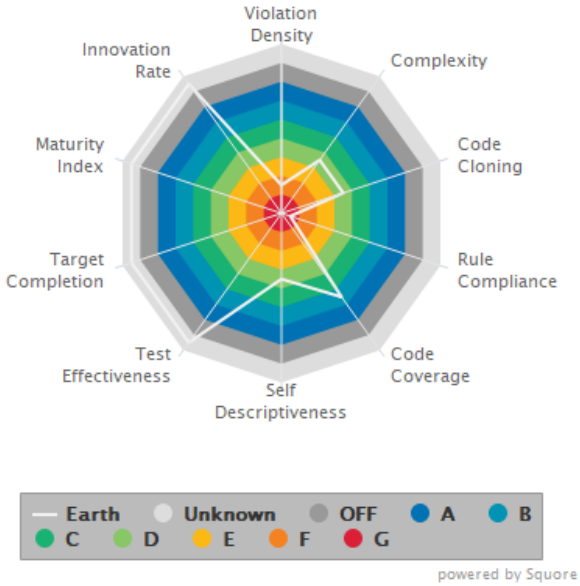
This section presents the Technical Debt trend of the project Earth.



Definition:
Technical Debt represents the human effort that shall be invested for the project in order to fix all deviations from the quality standard.
Technical Debt is computed from Coding Rule, Complexity and Cloning KPI.

Unit: Hours.

Formula:
Maintainability = Sum of remediation cost of all violations of Maintainability Coding Rules
Reliability = Sum of remediation cost of all violations of Reliability Coding Rules
Efficiency = Sum of remediation cost of all violations of Efficiency Coding Rules
Portability = Sum of remediation cost of all violations of Portability Coding Rules
Testability = Sum of effort to reduce module complexity which are not under test
Changeability = Sum of effort to remove cloning/duplication issues



Comment:

II. Complexity Overview

HIS Metrics are extracted from the industry standard.Here are the considered metrics and expected threshold

Function Metrics	Standard Reference
Comment Density COMF	>20%
Number of Paths PATH	<=80
Cyclomatic Complexity VG	<=15
Number of Parameters PARAM	<=5
Statement STMT	<=50
Nesting Level LEVL	<=5
Number of return points RETURN	<=1
Vocab Frequency VOCF	<=4

Classe Metrics	Standard Reference
Number Of Methods (NOM)	<=25
Weighted Metrics per Class (WMC)	<=60
Depth of Inheritance Tree (DIT)	<=2
Number of Children (NOC)	<=2
Multiple Inheritance (MII)	<=1
Number of Attributes (DATA)	<=7
Number of Public Attributes (APBL)	<=0
Number of Statements (STAT)	<=100

Complexity Indicator is based on cumulation of these metrics checking.A Classe or Function is considered as complex if at least half of these metrics do not respect the expected threshold.

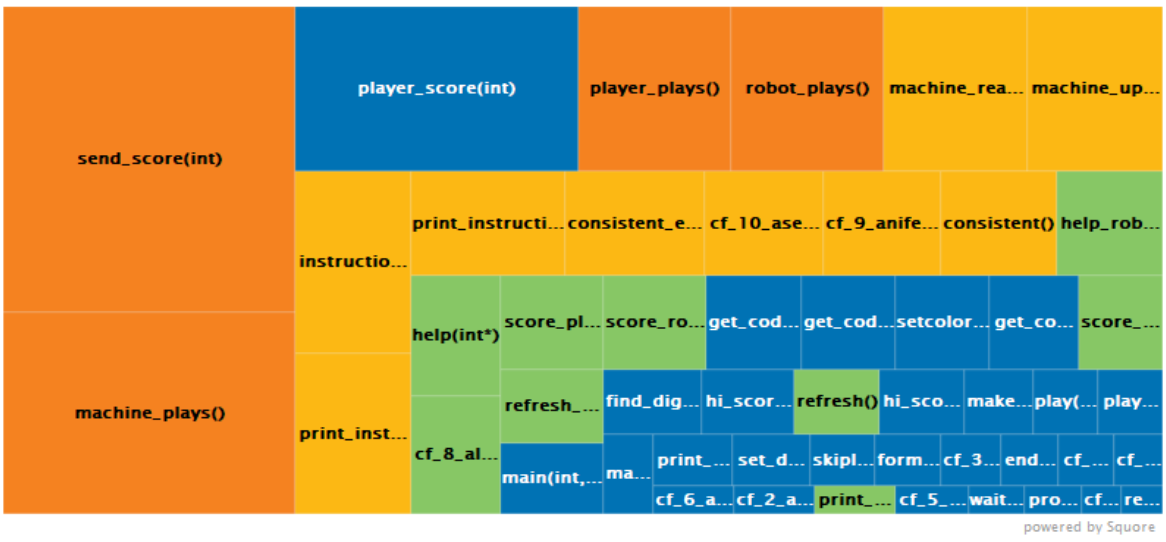
1) Complexity Volume

The complexity volume represent ratio of the code which is computed as very complex at function level (Function which are rated as "F" or "G" in term of

Complexity Volume Statistics

2) Complexity distribution

Distribution at Function level



III. Code Coverage Overview

1) Code Coverage

Code Coverage indicators follow the common Industry standard.

The table below shows the project's thresholds for the different types of coverage, distributed per Critical Factor.

Coverage Type	A	B	C	D
Statement Coverage	80	100	100	100
Branch Coverage	50	80	100	100
MCDC Coverage	0	50	80	100

(Values are percentages)

2) Test Strategy Settings

The Test Strategy consists in focusing on particular functions called: 'TO BETESTED'.
These functions are defined according to the following criteria:

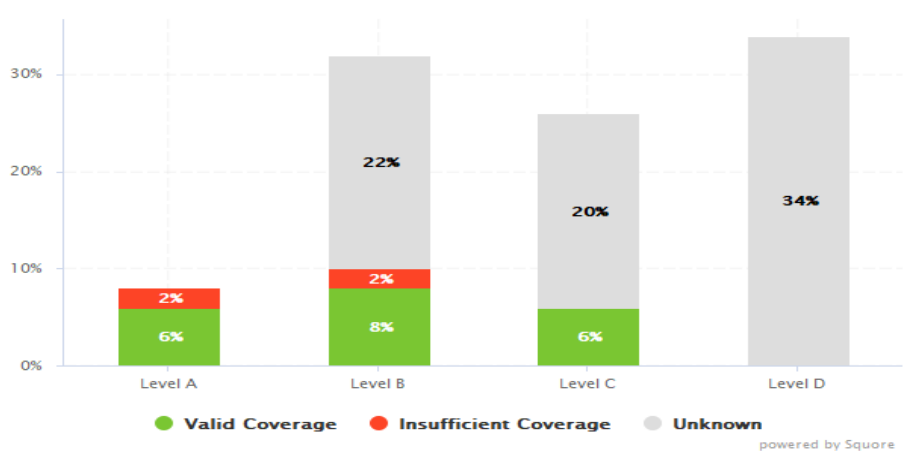
Metric	Threshold
Cyclomatic Complexity	-1
Number of Paths	-1
Nesting Level	-1
Vocab Frequency	-1

Function is considered as "To Be Tested" if all the criteria are reached.

Test Strategy status: 1

3) Test Coverage Compliance

The following graphic represents the Coverage compliance regarding Critical Factor. For each Critical Factor it is possible to assess the percentage of functions that respects the 3 types of coverage (Refer to previous 'Code Coverage' settings).

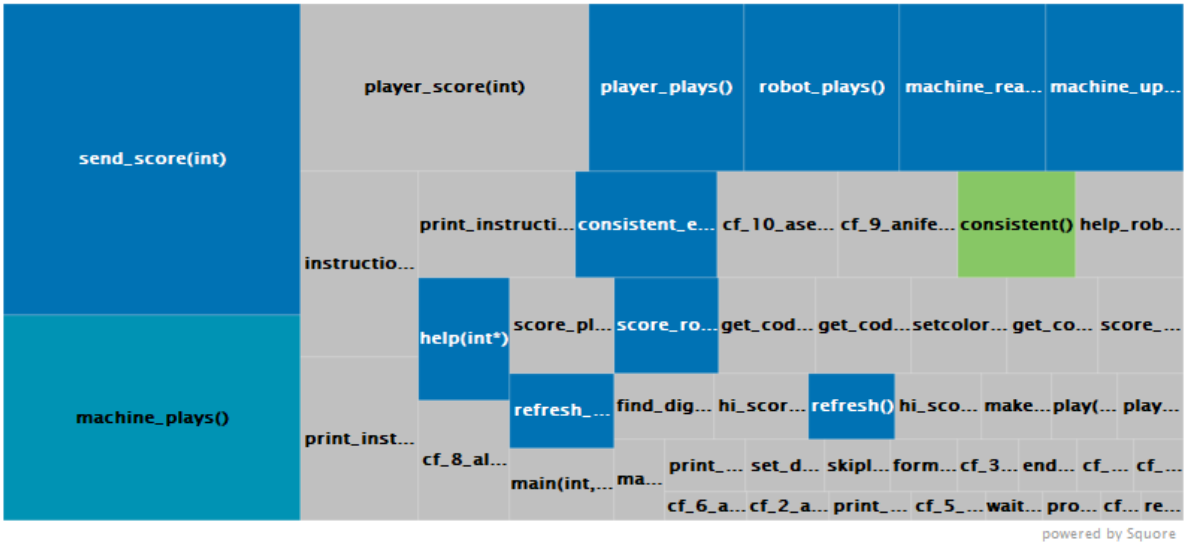


Note: the Test Strategy settings can impact the code coverage compliance.

If the test strategy is activated (status=1), Squore only focuses on 'To Be Tested' functions. This means that a function may not comply with coverage expectations but could be considered as compliant.

4) Coverage deviation Distribution

Distribution at Function level



IV. Coding Rules Overview

1) Coding Rule Compliance

The following table lists the Coding Rules statistics at the project level.

Rule Compliance	47.4%
Non Compliant Standards	10
Non Conformity Count	203
Non Conformity Density	96/KLoc
Coding Standards	19

2) Coding Rule Violations

Practice	Occ.	Delta	Data Provider	Severity	Remediation Cost
Recursion are not allowed	1	+1	Squan Sources	Critical	High
Goto shall not be used	1	+1	Squan Sources	Major	Medium
Missing Break	6	+2	Squan Sources	Critical	Low
Commented-out Source Code is not	1	+1	Squan Sources	Major	Low
Multiple exits are not allowed	39	+1	Squan Sources	Minor	Low
Assignment in Boolean	16	+10	Squan Sources	Minor	Low
Missing compound if	89	+41	Squan Sources	Minor	Tiny
Missing compound statement	47	+19	Squan Sources	Minor	Tiny
Cloned Functions	10	+8	Squan Sources	Major	Medium
Cloned Algorithmic	4	+4	Squan Sources	Major	Medium
Cloned Files	2	+2	Squan Sources	Major	Medium
Avoid Duplicated Blocks in Function	8	+6	Squan Sources	Major	Low
'abort, exit, getenv or system' shall not	3	+1	Squan Sources	Major	Medium
IO Functions shall not be used	3	0	Squan Sources	Major	Medium
'atof, atoi or atol' shall not be used	1	0	Squan Sources	Major	Medium
Time Handling Functions shall not be	1	0	Squan Sources	Major	Medium
Risky Empty Statement	1	+1	Squan Sources	Blocking	Low
Common realloc mistake: 'varname'	5	0	Cppcheck	Blocking	Medium
Dynamic Memory Allocation shall not be	2	0	Squan Sources	Major	Medium
Missing final else	2	+1	Squan Sources	Minor	Low
TODO shall not be commited in sources	1	+1	Squan Sources	Minor	Low
Fallthrough shall be avoided	2	+2	Squan Sources	Blocking	Low

3) Coding Rule Relaxations

Path	Name	Relaxation

Path	apps/master.c		
Name	main(int,char*[])		
Location		Rule	COMPOUND:Missing compound statement
Status	'Derogation' set by demo (2019-04-10T13:58:22)		
Justification	keeping some conciseness by avoiding using compound {}		

Path	apps/master.c		
Name	main(int,char*[])		
Location		Rule	NOASGCOND:Assignment in Boolean
Status	'Derogation' set by demo (2019-04-10T13:58:22)		
Justification	clearing out the buffer by assigning inst to the next char		