Rubric lab assignment Software Quality Management

Metrics	*Insufficient (1–4)*	*Poor (5–6)*	*Good (7–8)*	*Excellent (9–10)*
Lines of code	It is completely unclear how many lines of code the system consists of or what has been calculated	The calculation is somewhat close, but many choices have been made without conscious thought	The calculation of the number of lines is accurate and generally well substantiated	The calculation of the number of lines is accurate; the assumptions are made explicit
Unit size	It is completely unclear how many lines the units consist of	Unit sizes have been calculated, but the results are debatable and the precise meaning is unclear	The unit sizes are calculated; it is clear what this means for the risk categories	The calculation has been done accurately, with a correct translation to the given risk categories
Unit complexity	Cyclic complexity is poorly understood; the metric cannot be explained	Cyclic complexity has been calculated, but language constructs are counted incorrectly	The calculated cyclic complexity is almost correct; there is a translation to risk categories	The calculation is accurate, the substantiation is convincing, and the translation to risk categories is correct
Duplication	Code duplicates are not found, or it is unknown how much duplication the system has	The algorithm for finding duplicates works but is complex; therefore its correctness is uncertain	The algorithm for finding duplicates is clear; the percentage of duplication has also been calculated	Duplicates are found in a surprisingly elegant way and converted into a percentage
Test coverage (*)	–	–	A start has been made on calculating test coverage	Test coverage is calculated in a surprising way

Visualisation				
Interaction	The visualization is not interactive	The visualization supports some simple interactive elements	The user can interactively explore the visualized system and adjust the visualization	The interactive elements of the visualization are very well developed
Hierarchy	A hierarchy is missing in the visualization, making it superficial or incoherent	The visualization shows information at different levels, but the detail per level is limited	Information is displayed at different levels, so that overview and details are available	The visualization makes effective use of hierarchical data and treats the levels uniformly
Polymetricity	The data behind the visualization is not very informative and one-dimensional	The data behind the visualization is informative, but it is difficult to see or discover connections	The visualization is rich in multi-dimensional data; the visualization is polymetric	The visualization combines several values per element, thus revealing interesting cross-connections
Insightfulness	The visualization has virtually no added value compared to a textual output	The visualization gives a limited insight into the system	The visualization is effective and leads to insights that would remain hidden without the visualization	The visualization leads to surprising new insights into the system
Originality (*)	–	–	The visualization contains some original or surprising elements.	The visualization is very original and fully developed; the result exceeds expectations

Report	*Insufficient (1–4)*	*Poor (5–6)*	*Good (7–8)*	*Excellent (9–10)*
Reporting	The report is sloppy and inconsistent	The report is fairly neat, but the text requires clarification	Much attention has been paid to the report, the structure is logical and the text is clear	The report is concise, clear and error-free, with ample attention to substantiation
Design choices	There is no description of the design choices or their motivation	The design is described, but the motivation is weak	The description and motivation of the perspective on the architecture is good	The perspective of the visualisation is clearly and precisely formulated; moreover, the choices are convincingly motivated
Interpretation	The report makes no attempt to interpret the results	A total score for maintainability has been derived (according to the SIG model)	The interpretation of the results shows a good understanding of the maintainability	The root causes behind the maintainability scores have been reported and new insights have been found using the visualisation.
Validity	Although values have been calculated, insufficient steps have been taken to check their accuracy	Steps have been taken to validate the results found; the results are largely correct	When checking outcomes, careful consideration was given to the method and to edge cases	Several techniques have been used for a thorough validation
Evaluation	The visualisation is not evaluated	The visualisation is tested to some extent; however, the argumentation is weak	The report contains a good and fair evaluation of the visualisation based on scientific criteria	The visualisation is convincingly tested with criteria from the literature; the evaluation is objective and thorough
HSQL DB (*)	–	–	The metrics have been calculated for HSQL DB.	The maintainability of HSQL DB has been analysed and discussed.

Code				
Code elegance	The code is difficult to read and understand, poorly structured and full of *code smells*	The code is of sufficient quality; some problems could have been solved more easily	The code is neat and understandable, uses Rascal well, and is divided into methods and modules	The code is short and concise, makes excellent use of Rascal features and is suitable as a model answer
Explanation of the code	Code fragments are difficult to explain; Questions about the design are not answered	The written code can be explained; not all choices made can be explained	The code and the design choices made can be explained satisfactorily	All details, as well as the high-level design, are well explained and substantiated
Version control	Version control was not used	Version control was used poorly, with only a few commits, missing files, or with files that do not belong	Version control was used adequately	Version control was used extensively, the repository's layout is clear, and branches were used for developing independent features
Runing time (*)	–	–	It takes an hour or more to initially calculate the metrics for large systems	Systems are analysed efficiently, in $O(n \log n)$

(*) These criteria are optional and hence do not have an insufficient or poor grade