NASA/TM-20250009084

Code Quality Risk Assessment: A Novel Process for Quantifying Structural Code Quality at NASA

Cecil H. O'Dell
MPL Corporation, 768 Brushy Fork Rd, Buckhannon, WV, 26201

Vaughn L. Harvey
TMC Technologies, 2050 Winners Dr, Fairmont, WV 26554

Mallory K. Anderson
Science Applications International Corporation, 12010 Sunset Hills, Road Reston, VA 20190

Greyson A. Weimer
Science Applications International Corporation, 12010 Sunset Hills, Road Reston, VA 20190

David W. Carvell
New-Bold Enterprises Inc, 14 Mountain Park Drive, White Hall, WV, 26554

December 2025

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

.

NASA/TM-20250009084

Code Quality Risk Assessment: A Novel Process for Quantifying Structural Code Quality at NASA

Cecil H. O'Dell
MPL Corporation, 768 Brushy Fork Rd, Buckhannon, WV, 26201

Vaughn L. Harvey
TMC Technologies, 2050 Winners Dr, Fairmont, WV 26554

Mallory K. Anderson
Science Applications International Corporation, 12010 Sunset Hills, Road Reston, VA 20190

Greyson A. Weimer
Science Applications International Corporation, 12010 Sunset Hills, Road Reston, VA 20190

David W. Carvell
New-Bold Enterprises Inc, 14 Mountain Park Drive, White Hall, WV, 26554

National Aeronautics and
Space Administration

Katherine Johnson IV&V Facility
Fairmont, WV, 26554

December 2025

NASA STI Program Report Series

Available from:

NASA STI Program / Mail Stop 050
NASA Langley Research Center
Hampton, VA 23681-2199

This technical memorandum is dedicated to the memory of Vaughn L Harvey, a NASA IV&V legend and a core contributor to the CQRA process.

National Aeronautics and
Space Administration

Katherine Johnson IV&V Facility
5000 NASA Blvd., Fairmont, WV 26554

December 2025

Abstract: The Code Quality Risk Assessment (CQRA) is a novel process within NASA to quantify structural code quality of mission software. By balancing source code metrics, static-code analysis tool output, statistical random sampling, and experienced engineering judgment, a robust measure of code quality is produced. This measure is complemented by ample detail on which aspects of the source-code are positive, and which aspects could use improvement. CQRA as a process can be an integral part of improving source quality from the early stages of development, which has been shown to decrease the overall need for code maintenance and the cost of cleaning up issues that will inevitably show up later in the development lifecycle. CQRA as a technique has identified key risks, weaknesses, and strengths with respect to code quality on numerous NASA missions and will continue to be a resource applied to improve mission-critical code quality.

Introduction: Prior to the implementation of the Code Quality Risk Assessment (CQRA) within NASA IV&V's Catalog of Methods (M-152), no known process existed within NASA to directly quantify the level of structural risk for the mission source-code. Numerous processes exist to identify code defects, coding standards violations, metrics threshold violations, vulnerabilities, and mission-impactful risks, but CQRA was the first to holistically assess structural software quality for NASA mission software. Other targeted code analysis approaches consider the impacts of defects to respective missions or established coding standards, which undeniably contributes to quality code, as well as properly validated and verified code. However, CQRA is the first within NASA to measure and assess the "risk landscape" of the code at large.

CQRA consists of a series of larger aspect areas that provide a specific perspective to assess architecture, safety-critical coding standards, maintainability, fault tolerance, testability, complexity, and security. Each of these aspect areas contain further decomposed categories known as focus areas. For each focus area, there are a series of questions that must be answered with either specific code metrics, expert-judgment guided by random sampling, or static-code analysis tool output. The different sources of answers help eliminate the possibility of being skewed by a single avenue of bias or counterproductive debates regarding arbitrary metric thresholds and cut-off values as to what is "concerning" versus what is "ideal." CQRA provides multiple sources of evidence to calculate individual and overall quality scores, and the process is adaptable to the point that it does not depend on individual concepts or arbitrarily defined quality thresholds. Rather, it uses a healthy plethora of inputs with the included capability to tailor or adapt portions of the process to fit the specific needs of the software in question. In other words, truly negative quality indicators have more meaning if they emerge simultaneously from the multiple methods of assessment, and each method plays an equally significant role in the overall scoring.

National Aeronautics and
Space Administration

Katherine Johnson IV&V Facility
5000 NASA Blvd., Fairmont, WV 26554

December 2025

History: CQRA began with the creation of a new approach towards assessing software maintainability on MCC IV&V in early 2020 (M-148). This approach was a precursor to CQRA, and it involved utilizing code metrics like cyclomatic complexity and test code coverage correlated with static-code analysis findings to update the risk posture of MCC IV&V as well as document any systemic concerns with software quality.

This work continued in 2022, when a team of NASA IV&V static-code analysis (SCA) subject-matter experts (SMEs) was formed at the request of the NASA Office of Safety & Mission Assurance (OSMA) Software Assurance Tech Fellow under NASA IV&V's SSO Outreach Team. This team was tasked with investigating and adapting a pre-existing Carnegie Mellon Software Engineering Institute process called Crew Risk Estimation Worksheet (CREW). CREW was adapted and revised by this SME team to better fit the needs of NASA mission software. The adapted CREW was combined with the analysis steps of the previously referenced M-148 and CQRA was created and published as a formal analysis process within the IV&V Catalog of Methods.

Projects Scored: Since the initial creation of the process, numerous NASA missions and software projects have utilized CQRA to identify areas of improvement within mission software. These projects include FPrime, which is an open-source NASA flight software and embedded systems framework, and the Regenerative Fuel Cell (RFC) project led by the Glenn Research Center. IV&V projects that have utilized CQRA as part of their software analysis include Europa Clipper, Roman Space Telescope, Exploration Ground Systems (EGS), xEVAS, and HLS-SpaceX.

Approach: CQRA is a process designed to measure the risk in the code itself, considering only the code files that will be compiled and run on respective NASA missions. The process does not consider design documentation, requirements, or any system artifacts. This process intends to provide evidence for and against structural code quality - the quality of the code itself. Validation of functional code quality, which is how well software correctly meets the specified requirements for the mission, is not in the CQRA's scope, as there are numerous other processes to assess this within NASA and NASA IV&V's domain.

In CQRA, are 31 focus areas within the 6 overall aspect areas mentioned above. All the questions within the assessment are centered around the source code and can be answered either with static analysis tools, code metrics, analyst evaluation, or some combination of all the above.

Each question in CQRA has a score between one, indicating the lowest level of risk, and five, indicating the highest level of risk. The individual question scores are then rolled up into overall scores for each focus area and aspect, as well as for the overall software project. Just as in the individual question scores, the overall scores are on a scale from one to five, with five indicating the highest level of risk. Through this process, software code quality can be quantified in a way that can be easily processed and digested by developers and analysts working on the project. This is normally represented by a final score roll-up containing individual scores for the aspect/focus areas, as well as qualitative comments on what to improve and identified good practices to maintain (Figure 1).

ASPECT RISK	ASPECT /	Focus	SUSTAIN	IMPROVE
1.0	ARCHITECTURE /			
	1.0	Modularity	High system stability, low intercomponent cyclicality, and an easily identifiable architectural pattern	Make delineation between flight and non-flight code more clear and organized
	1.0	Dependency		
	1.1	Patterns		
1.4	STANDARDS /			
	1.4	Criticality	Adherence to large majority of NASA Power of 10	Be wary of functions with extensive length
2.0	MAINTAINABILITY /			
Comments	1.8	Quantity	Commenting style appears consistently easy to maintain, detail included is consistent	Improve comment to code ratio, remove commented out code
	1.6	Quality		
Clarity	2.3	Consistency	Code identation, whitespace, and parentheses used effectively to organize the layout of the code	Utilize consistent naming conventions
	1.5	Constants		
	1.4	Files / Folders		
	2.3	Readability		
	2.8	Conditionals		
Reusability	1.7	Independence	Clear separation between autogenerated code and developed code	Large volume of unused functions/variables, some possible unreachable code, and moderate use of GoTos
	3.8	Utilization		
	1.8	Complication		
	1.0	Duplication		
2.5	FAULT TOLERANCE /			
	2.1	Value Checking	Ample evidence of system/instrument health logging, Ample evidence of a watchdog timer	Add consistency and thoroughness in how exceptions/errors are handled, add consistent detail to logs
	3.1	Error Handling		
	2.5	Error Logging		
	2.9	Instrumentation		
	1.7	Health Monitoring		
1.9	SECURITY /			
General	1.4	Defensiveness	Consistent checking of return values, evidence of regular memory cleans between reads, evidence of hashing	No evidence of modern encryption algorithms
	2.5	Validation		
	1.7	Libraries		
	2.2	Warnings		
	1.5	Traceability		
CERT	1.3	Density	Most CERT rules had no violations detectable by CodeSonar	Recommend utilizing sound static analyzer to ensure CodeSonar diagnostics for buffer overflow/memory leaks are false positives
	1.7	Priority		
1.4	TESTABILITY /			
	1.4	Unit	96.7% of functions have a cyclomatic complexity <= 10, unit testing is generally easy to understand and	There are a significant number of functions with a cyclomatic complexity > 50
	1.5	Complexity		
	1.86		**TOTAL RISK**	

Figure 1: Sample IV&V Mission Overall CQRA Scoring

As mentioned previously, there are three major types of questions in the focus areas that allow for this overall score roll-up: code metrics, expert judgment calls guided by random sampling, and static-code analysis tool output. The code metrics are measured by some of software evaluation tools used by analysts, including Understand, Lattix, CodeSonar, and PMD. Some of the metrics utilized in the process include modified cyclomatic complexity, the average number of code lines per function, and the maximum control structure nesting level within the code (to name a few). A sample score for one of these tool metric questions is shown below in Figure 2.

Measurement / Question	Scale	Percent / Count	Count Limit	Goodness Bias	Weight	Rationale (the reason a score was selected from the evidence)	Tool
Number of goto occurrences within the code, i.e. Hersteller GOTO metric	Count	80	80	-1	MediumHigh	I ran the checker and there were 80 violations across 6056 files that were analyzed. Using Slovin's formula, I determined that I could restrict my sample size for inspection to 45 to achieve a confidence limit of 90%. All of them were valid. I will conclude with 90% Confidence Limit that all 80 are valid.	Understand

Figure 2: Sample IV&V Mission Tool Metric Scoring

Expert judgment also utilizes the metrics collected by software evaluation tools but require the insight of the analyst to determine the final score, rather than previously vetted numerical thresholds. Thus, the final score is determined by both the metrics provided by the analysis tools and the analyst's judgement. This type of assessment requires the analyst make an informed decision on how to score the metric based upon their knowledge and interpretation of the output produced by the software evaluation tool being used. An example of this is shown in Figure 3. This process is shown in Figure 4. For example, a percentage reading from the software tool may be translated by the analyst to a ranking score of either "No," "Minimally," "Somewhat," "Mostly," or "Yes" to answer the question. This is useful for metrics or patterns with no widely agreed upon "negative" threshold, or to include specific context about anything unique in the code itself that could add more to the overall story in the quality assessment.

Measurement / Question	Scale	Score	Goodness Bias	Weight	Rationale (if analyst judgment is involved - why did you score how you did?)	Tool(s)
Does the system use a common architectural pattern (layering, client-server, MVC, etc.)?	Ranking	Yes	1	High	The "Conceptual Architecture Diagram" that is presented appears to show a peer relationship between the 3 top level system components, with a layered architecture within each of those.	Lattix

Figure 3: Sample IV&V Mission Analyst Estimate Scoring

Measurement / Question	Scale	Score	Goodness Bias	Weight	Rationale (if analyst judgment is involved - why did you score how you did?)	Tool(s)
Within components is there a high degree of coupling among modules? (are there many > 5)?	Ranking	No	-1	MediumHigh	I looked at the % value for the files shown in the expanded view. All were 0.0% Thus, this is scored "No".	Lattix

Figure 4: Sample IV&V Mission Analyst Selection Scoring

Each metric in the CQRA has a final score between 1 and 5, and a conversion process must be followed to translate the analyst's scoring into the final score. To fully understand how to translate the scores produced by the analyst into this 1 through 5 score, it is important to understand the attributes that each of the CQRA metrics has. These attributes are shown below in

Figure 5. “Scale” is used to indicate the way the question will be scored, which could be as a “Count,” “Ranking,” “Binary,” or “Percent.” The “Score” column is utilized by “Ranking” and “Binary” questions, while the “Percent / Count” column is used by “Count” and “Percent” questions, and the “Count Limit” column is used for “Count” questions. All metrics utilize the “Goodness Bias” column, which uses a score of 1 to represent that an answer of “Yes” is favorable and a score of –1 to represent that an answer of “No” is favorable, and the “Weight” column, which categorizes the importance of each metric in determining software quality. For example, a metric with a weight of “High” is more critical in determining code quality than a metric with a weight of “Medium.”

Measurement / Question	Scale	Score	Percent / Count	Count Limit	Goodness Bias	Weight
Does each source file #include the header file of the same root name?	Ranking	Mostly			1	MediumHigh
Total number of functions or methods within the code base	Count		44573	50000	-1	High
Average lines of code per file	Count		531	500	-1	Medium
Average lines of code per function or method	Count		72	30	-1	MediumHigh
Percentage of source files that exceed 1000 lines of code in length?	Percent		6.1		-1	MediumHigh

Figure 5: Important CQRA Metric Attributes

The translation from the scoring by the analyst shown above to the final risk score with a value between 1 and 5 is automated within the CQRA spreadsheet and completed in several steps, which are shown in Figure 6 below.

Computed Score			Computed Risk		
Raw Percent Score	Goodness Score	Numeric Weight	Percent Risk	Weighted Percent Risk	Risk
75	75	0.75	25	18.75	1
89.146	10.854	1	89.146	89.146	4.4573
100	0	0.5	100	50	2.5
100	0	0.75	100	75	3.75
6.1	93.9	0.75	6.1	4.575	1

Figure 6: Calculation Steps to Determine Final Metric Risk Score

The “Raw Percent Score” is first determined by converting the various Scales to a purely numeric, 0-100 scale. Ranking is divided such that a “No” answer equals 0, a “Minimally” equals 25, a “Somewhat” equals 50, a “Mostly” equals 75, and a “Yes” equal 100. For a Count scale, the “Raw Percent Score” is determined by dividing the Count by the Count Limit. Percent is a direct translation to the Raw Percent Score. For Binary, a “No” is 0, and a “Yes” is 100.

The “Goodness Score” merely converts each “Raw Percent Score” such that a higher score is preferable. This is accomplished by “inverting” the “Raw Percent Score”, i.e. a “Raw Percent Score” of 25 with a “Goodness Bias” of -1 would result in a “Goodness Score” of 75. A “Raw Percent Score” of 15 with a “Goodness Bias” of 1 would result in a “Goodness Score” of 15.

The “Numeric Weight” is the Weight column translated to Numbers. “High” is 1, “MediumHigh” is 0.75, “Medium” is 0.5, and “Low” is 0.25.

The “Percent Risk” column then takes the “Goodness Score” and converts it to a useable Risk Percentage. Since the “Goodness Score” column has already normalized our “Raw Percent Scores” such that higher is less risky, the “Percent Risk” merely takes 100 minus the “Goodness Score.” This inverts our “Goodness Score” such that a higher score is riskier. This is used as it makes the Risk Calculation much easier.

The “Weighted Percent Risk” is just the “Percent Risk” multiplied by the “Numeric Weight”. This applies our weight from the questions to the risk, aiding in the future rollup of the Focus Areas and Aspects.

The “Risk” column then takes the “Weighted Percent Risk” and converts it to a scale of 1 to 5. A “Weighted Percent Risk” of 100 would equal a 5, and a “Weighted Percent Risk” of 20 or below equals a 1. The Risk calculation therefore is:

$$max\left[1,\ \left(\frac{100}{[WeightedPercentRisk]}\right)*5\right]$$

Random Sampling: Many entities in mission-critical software can have astronomical instance counts. To put the scale into perspective, consider that NASA IV&V routinely receives software builds over 1 million SLOC. Consider how many functions and variables there must be for a given codebase of 1 million SLOC. Assessing quality at large aside from what static-code analysis tools can be difficult, especially considering the possible variability throughout an entire mission flight software (FSW). Different sections of the code may have different developers and varying styles. To address this, particularly for SEI CREW, for questions involving the quality of the functions, comments, and variables within a given code base, the CQRA team turned to the prospect of random sampling. This random sampling was driven by Python and Ada scripts to randomly select files and line numbers in a build, which was then accompanied by established confidence using Slovin's formula [1].

In other words, we had a total population of some "entity" that we wanted to analyze. Examples of an "entity" include files, functions/methods, SCA warnings, requirements, or unit tests.

1) We want to perform an analysis on these "entities" to assess some question(s) about the "entity". Examples are a) Does the file have a header b) Is the name of the function descriptive of what the function is implementing c) is the SCA error message a True Positive
2) We do not know anything about any of the "entities"
3) We want to know the sample size that, by *randomly* analyzing that sample, would allow us to conclude that the results of that analysis are representative of the entire set of "entities", within a confidence limit (CL)

Research of this problem resulted in selecting the following formula, referred to industry as "Slovin's formula". It is as follows: n = N / (1+Ne**2). Where:

N = Total Population.

e (error tolerance) = 1 – CL. (Note: CL can range from 0% to 100%)

n = Sample Size to *randomly* analyze.

Then, for a specific problem: N = 500 functions. We want to know if the function name is descriptive of what the function is implementing. For this example, we will choose the CL = 90% = .9

⇨ **e = 1 - 0.9 = 0.1**

And, applying the formula, we get:

n = 500 / (1 + 500(.01)) = 500 / 6 = 83.3. We need to round up to 84.

Thus, this says we can *randomly* analyze 84 functions, instead of analyzing all 500 functions, and use that result to conclude that the entire population of 500 has the same distribution for the question being asked. i.e., If 80% of the 84 functions analyzed have a name that is descriptive of what the function is implementing, then 80% of the total set of 500 functions do have that as well.

The word “*random*” is critical. You would not want to analyze a set of functions confined to a restricted set of files. That set of *random* “n” to sample can be provided with the implementation of a script.

Risk Integration: NASA IV&V has historically made use of two risk assessment processes applied to prioritize analysis areas and determine a varying degree of rigor to allow focus on the highest risk areas. The first process detailed in S3106 Rev E [2] is called a Portfolio/Project Based Risk Assessment (PBRA) and is designed to evaluate mission capabilities at a high level. Individual capabilities are IV&V generated sentences of functional intent reflecting core components of a mission based on system understanding of the available artifacts. In the PBRA, these capabilities are decomposed and evaluated based upon how much risk they introduce to a mission in terms of both the likelihood that a capability will not have complete success, as well as the impact that a partial or full capability failure. Capabilities are not necessarily linked to individual software entities and may require a “mapping” of software entities within the project to directly identify areas to analyze. These software and system entities are evaluated with the Risk Based Assessment (RBA). This process assesses the level of risk associated with a project’s software entities, which provides IV&V teams with the information needed to determine the best way to allocate resources to a project. The PBRA and RBA assessments both aim to evaluate mission capabilities and software entities in terms of the different ways in which they could introduce risk to a mission. Thus, capabilities and software introducing the most risk can be given the attention needed for targeted IV&V.

The assessments include a measurement for likelihood, which is defined as the potential for the existence of errors within an identified mission capability. This likelihood is composed of a series of attributes: degree of innovation, complexity, testability, and heritage to name a few. However, structural code quality can also be an input for likelihood, as a probability measurement for future errors or hazardous conditions to occur in within a mission software codebase. Just like the earlier point of the benefit to having variety in quality assessment methods, the same is true for likelihood attributes. Evidence based attributes such as code quality can be added to the assessment for the PBRA/RBA or for specific assurance objectives. Additionally, by completing this calculation for each individual mission capability, differences in capability quality can be considered for risk assessment scoring, which can inform appropriate levels of rigor.

A variety of information is needed to calculate the quality of capabilities for NASA missions. These values will serve as inputs to the quality calculation. Figure 7 details some of these necessary inputs, which includes the score from the CQRA. Note that a QVscribe score (QVscribe is an NLP tool to assist in requirements analysis by quantifying requirement quality and identifying potential issues within requirement text) will also serve as an input value.

Capability	Release Number	QVscribe Score	CQRA Score
Enter each PBRA capability as an individual row on the sheet.	Enter an integer representing the number of the project's release that is being analyzed. For example, if the fourth release version of the software is being analyzed, "4" should be entered as the score. Note that this number should be the same for all capabilities.	Enter the value of the QVscribe analysis of the capability.	Enter the value of the Code Quality Risk Assessment (CQRA) for the project. This score should be the same for all capabilities, as all of the source code is analyzed at once.

Figure 7: Quality Calculation Overall Input Values

The rest of the input values required to calculate a quality score are detailed below in Figure 8. These input values pertain to Technical Issue Memorandums (TIMs), which are formal records of issues in a mission's development artifacts determined to have an adverse impact to the mission if they are not resolved. TIMs are categorized into different levels of severity, with Severity 1 TIMs having a catastrophic impact to a mission if not resolved. Severity 4 TIMs are not included in the quality calculation, as they represent only an impact to the maintainability of the mission artifacts if not resolved.

Number of Requirements TIMS	Total Number of Requirements TIMS	Number of SCA TIMs	Total Number of SCA TIMs
Enter the number of TIMs that use either the M-2 or the M-3 methods for severity 1, severity 2, and severity 3 issues, respectively, for *this release of the project only*.	Enter the total number of TIMs that use either the M-2 or the M-3 methods for severity 1, severity 2, and severity 3 issues, respectively, for the project's entire analysis history.	Enter the number of TIMs that use the M-9 method for severity 1, severity 2, and severity 3 issues, respectively, for *this release of the project only*.	Enter the total number of TIMs that use the M-9 method for severity 1, severity 2, and severity 3 issues, respectively, for the project's entire analysis history.

Figure 8: Quality Calculation TIMs Input Values

The overall quality score that can be used in conjunction with existing PBRA scoring methods is comprised of several calculations that utilize the input values described in the figures above. Scores relating to the rate of both requirements TIMs and SCA TIMs are calculated first. These scores are known as the Requirements TIMs Rate Score and the SCA TIMs Rate Score and are described in detail in Figure 9 below. Overall scores for both Requirements TIMs and SCA TIMs can then be calculated by finding the weighted average of the TIMs Rate Scores for severity levels 1, 2, and 3. These scores, known as the Requirements TIM Score and the SCA TIM Score, are also shown below in Figure 9. Note that severity 1 TIMs are given a higher weight than severity 2 TIMs, which are given a higher weight than severity 3 TIMs. This is because severity

1 represents a catastrophic impact to the mission if not resolved, while severity 2 represents a critical impact, and severity 3 represents a moderate impact.

Requirements TIMs Rate Score	SCA TIMs Rate Score	Requirements TIM Score	SCA TIM Score
The number of requirements TIMs in the current release version (C) are compared with the number of requirements TIMs for the total project divided by the release version number for the project that is being analyzed (R). Scores are assigned as follows: 5: C > R 4: C = R 3: 0.7 <= C/R < 1 2: 0.4 <= C/R < 0.7 1: C/R < 0.4 Repeat for severities 1, 2, and 3.	The number of SCA TIMs in the current release version (C) are compared with the number of SCA TIMs for the total project divided by the release version number for the project that is being analyzed (R). Scores are assigned as follows: 5: C > R 4: C = R 3: 0.7 <= C/R < 1 2: 0.4 <= C/R < 0.7 1: C/R < 0.4 Repeat for severities 1, 2, and 3.	This score is found by taking the average of the Requirements TIMs Rate Score for the Severity 1, Severity 2, and Severity 3 Requirements TIMs.	This score is found by taking the average of the SCA TIMs Rate Score for the Severity 1, Severity 2, and Severity 3 SCA TIMs.

Figure 9: Calculations Relating to TIMs

Once the calculations described above are completed, the overall Requirements Quality Risk Score and Code Quality Risk Score can be calculated. The Requirements Quality Risk Score is simply the average of 6 – the QVscribe Score and the Requirements TIM Score. Note that the QVscribe score must be subtracted from 6 because it is measured on a different scale than all of the other input values for the quality calculations. A score of 5 in QVscribe indicates strong quality, while a score of 5 for the rest of the inputs and calculations indicates poor quality. Subtracting the QVscribe score from 6 puts all the input values on the same scale. The Requirements Quality Risk Score is measured on a scale of 1 to 5, where 1 represents high requirements quality and 5 represents low requirements quality. The Code Quality Risk Score is calculated as the average of the CQRA Score and the SCA TIM Score and is measured on a scale of 1 to 5, where 1 represents high code quality and 5 represents low code quality.

Requirements Quality Risk Score = ((6 – QVscribe Score) + Requirements TIM Score) / 2

Code Quality Risk Score = (CQRA Score + SCA TIM Score) / 2

The final calculation is for the overall Quality Score. This score takes all the previously discussed input values into account and is equally based upon requirements quality and code quality. The overall quality score is calculated as the average of the Requirements Quality Risk

Score and the Code Quality Risk Score and is measured on a scale of 1 to 5, where 1 represents high quality and 5 represents low quality.

Quality Score = (Requirements Quality Risk Score + Code Quality Risk Score) / 2

This overall quality score serves as an important metric to consider when looking at a project's risk. Project requirements and software both play an immense role in determining the risk that a project has, and identifying areas of poor quality can help to significantly mitigate that risk. The current IV&V risk assessment processes do not take quality into account, so including the Quality Score calculation would ensure that requirements and code quality are factored into the risk calculations for future missions. This increases the overall effectiveness of the risk assessment and efficiency in the use of resources and engineering rigor for an IV&V project.

To ensure that this Quality Score is meaningfully incorporated to the PBRA likelihood measurement, the different versions of the S3106 document, which record how the PBRA and RBA processes are to be completed, must be considered. Missions currently undergoing IV&V analysis use either Revision D, Revision E, or Revision F of the S3106 document to score the potential risk of the mission's capabilities and software entities. Because there are different revisions of the document that are in use for this process, the incorporation of the Quality Score must be tailored for each revision.

Revision D of the S3106 document uses each capability's Complexity, Testability, Degree of Innovation, and Development Characteristics as criteria for calculating the PBRA likelihood score. Each criterion is scored on a scale of 1 to 5, with 5 representing the highest risk, placing them on the same scale as the final likelihood score. To integrate the Quality Score here, it can simply be introduced as additional criteria. Because a Quality Score of 1 represents high quality, while a score of 5 represents low quality, the Quality Score integrates easily with the rest of the Revision D likelihood criteria.

The PBRA likelihood score for Revision E of the S3106 document is calculated in a similar way as Revision D, with the calculation criteria differing slightly to more closely align with the "Capability Based Assurance approach to performing IV&V." The criteria for calculating the likelihood score for this revision are Complexity, Robustness, Innovation, Information Assurance, Heritage, and Mission Specific. As in Revision D, each of these criteria is scored on a scale of 1 to 5, with 5 representing the highest risk, placing them on the same scale as the final likelihood score. Because of this, the Quality Score can simply be introduced as additional criteria to use in calculating the final likelihood score.

Revision F is currently undergoing development at NASA IV&V, but the likelihood components will be complemented by the evidence-based quality criteria in a similar fashion to Revisions D&E.

All above implementations serve as a blueprint for enhancing and further developing follow-the-risk strategies traditionally employed by NASA IV&V to identify the areas of mission software with the most risk. Currently, CQRA's risk integration is being used on multiple NASA IV&V projects to include the story of the risk within the structural code composition in addition to the mission aspects. Together, this forms a more complete and effective landscape of risk articulation.

Conclusion: The Code Quality Risk Assessment was prototyped and streamlined at NASA's Independent Validation and Verification facility and to this day is an analysis technique applied to mission-critical software. CQRA helps fill a controversial gap in the software assurance industry – how to measure code quality. Rather than debating over specific metrics, acceptable thresholds, tool capabilities or performances, CQRA uses all possible resources to avoid biases to produce a robust picture of structural code quality. Moving forward, NASA IV&V is making use of it to help NASA missions identify code quality risks and strengthen their development practices, which leads to more robust and reliable code for NASA missions. Paired together with static-code analysis approaches, independent test analysis, and requirements implementation analysis, a complete picture of the risk landscape of mission-critical code is now within reach. For more information on CQRA, please reach out to NASA IV&V.

References:

[1] SHAADAN, NORSHAHIDA BT. "Sample size determination in survey research." (2009).

[2] NASA IV&V. *S3106: PBRA and RBA Process Rev E*, https://www.nasa.gov/wp-content/uploads/2016/04/s3106_pbra_and_rba_process_ver_e_04-26-2018.pdf

National Aeronautics and
Space Administration

Katherine Johnson IV&V Facility
5000 NASA Blvd., Fairmont, WV 26554