Software Acquisition Life Cycle Measure Plan based on the revised “IEEE P1633\AIAA R-013A Recommended Practice on Software Reliability”

Dr. Norman F. Schneidewind
Naval Postgraduate School
nschneid@nps.navy.mil
**Software Acquisition Life Cycle Measure Plan based on the revised IEEE P1633AIAA R-013A Recommended Practice on Software Reliability**

**Naval Postgraduate School, Monterey, CA, 93943**

*Approved for public release; distribution unlimited*
Outline

• Background
• Problem Definition
• Potential Solution
• Analysis of Results
• Reliability Risk Model Validation
• Reliability Risk Model Application
• Summary
• References
Background (1)

- Report on the revision of *Recommended Practice, AIAA/ANSI, R-013-1992, Software Reliability* [1]. The revision has the joint sponsorship of the IEEE and the AIAA.

- Emphasis in the original document was on software reliability models, test phase data collection necessary to support the models, and model predictions of software reliability made in the test phase for non-networked software.

- In the ten years since the document was published, there have been notable developments in predicting reliability *much earlier* than the test phase – as early as the requirements phase [2, 3].
Background (2)

- Therefore, the revision will address reliability prediction over all phases of the software life cycle, since identifying errors early reduces the cost of error correction. In addition, there have been advances in modeling and predicting the reliability of networks and distributed systems.
- These developments will be included in the revision.
- The revision will be an important lifecycle software reliability process document to achieve the following objectives:
  - Provide high reliability in DoD and aerospace safety and mission critical systems.
  - Provide a rational basis for specifying software reliability requirements in DoD acquisitions.
  - Improve the management of reliability risk.
Problem Definition (1)

• While software design and code metrics have enjoyed some success as predictors of software quality, the measurement field is stuck at this level of achievement.

• If measurement is to advance to a higher level, we must shift our attention to the front-end of the development process, because it is during requirements analysis that errors are inserted into the process.
• A requirements change may induce **ambiguity** and **uncertainty** in the development process that cause errors in implementing the changes.
• Subsequently, these errors **propagate** through **later phases** of development and maintenance.
• These errors may result in **significant risks** associated with implementing the **requirements**.
• For example, **reliability risk** (i.e., risk of faults and failures induced by changes in requirements) may be incurred by deficiencies in the process (e.g., lack of precision in requirements).
Potential Solution

• Identify the attributes of requirements that cause the software to be unreliable.

• Quantify the relationship between requirements risk and reliability.

• If these attributes can be identified, then policies can be recommended to DoD and NASA for recognizing these risks and avoiding or mitigating them during development.
Analysis of Results (1)

• Identified thresholds of risk factors:
  – Attributes of a requirements change that can induce reliability risk for predicting when the number of failures would become excessive (i.e., rise rapidly with the risk factor) [4].

• Two of the most important requirements risk factors of the Space Shuttle, as measured by their negative affect on software reliability, are \textit{space} and \textit{issues}.

• \textit{Space}: amount of memory space required to implement the requirement change

• \textit{Issues}: number of possible conflicts among requirements.
Analysis of Results (2)

• In [4], it was determined that space and issues had the highest statistically significant relationship with reliability.
  – The greater the cumulative memory space required to implement changes and the greater the number of cumulative conflicting requirements issues caused by the changes, the greater the negative effect on reliability.
Analysis of Results (3)

• An example is shown in Figure 1, where cumulative failures are plotted against cumulative memory space for both actual and predicted data.
  – The figure shows that when memory space reaches **2688 words**, actual cumulative failures reach **three** and climb rapidly thereafter.
Figure 1: Failures vs. Memory Space

\[ CF = 6E-07 \times CS^2 - 0.0003 \times CS + 1.9511 \]

CF: Cumulative Failures
CS: Cumulative Memory Space

Actual
Predicted
Analysis of Results (4)

• In Figure 2, cumulative failures are plotted against cumulative requirements issues, for both actual and predicted cases. When issues reach 272, actual cumulative failures reach three and climb rapidly thereafter.

• In both cases, a cumulative failure count of three has been identified as a critical value.
Figure 2: Failures vs. Issues

CF = 0.2481860 \times \exp(0.0107263 \times CI)

CF: Cumulative Failures
CI: Cumulative Issues
Analysis of Results (5)

- Although the counts of **2688 words** and **272 issues** provide estimates of the **threshold** to use in controlling the reliability of the next version of the software, the next version may not exhibit bends in the curves at the same value of risk factor.
- Therefore, the **prediction equations** and plots **generalize** the relationship between risk factors and reliability, such that they can be used to **predict** cumulative failures for **any** given value of cumulative risk factor.
- This process would be repeated across versions with the prediction equations being updated as more data is gathered.
Analysis of Results (6)

• Additional insight about the relationship between risk factors and reliability can be gained from the first derivative of the prediction equations in Figures 1 and 2 (i.e., rate of change). These are shown in Figures 3 and 4 for space and issues, respectively.

• Because the equation in Figure 1 is a second-degree polynomial, its derivative in Figure 3 is linear;
  – Thus, the prediction is a constant rate of change.
Figure 3. Rate of Change of Failures with Memory Space

\[ \frac{dCF}{dCS} = (-0.0003) + (0.0000012 \times CS) \]

CF: Cumulative Failures
CS: Cumulative Memory Space
Analysis of Results (7)

• In contrast, because the equation in Figure 2 is an exponential, its derivative is also an exponential and is simply the original function multiplied by a constant. This plot is shown in Figure 4.
• In comparing Figures 3 and 4, the implication is that we should have more concern about the negative effect on reliability of issues because of its predicted explosive growth rate.
Figure 4. Rate of Change of Failures with Issues

\[ \frac{dCF}{dCI} = 0.0107263 \times CF \]

CF: Cumulative Failures
CI: Cumulative Issues
• For the next version of this software, we want to predict the cumulative values of risk factors that correspond to given values of cumulative failures, particularly critical values.

• Figures 5 and 6, show the plots corresponding to the equations on the figures. These equations and plots were obtained by solving the equations of Figures 1 and 2 for cumulative risk factor as a function of cumulative failures. For example, if cumulative failures equal to 3 are considered critical, this would correspond to 1596 words of memory (Figure 5) and an issue count of 232 (Figure 6).
Figure 5. Memory Space vs. Failures

$CS = \frac{b + (b^2 - (4 \cdot a \cdot (c - CF)))^{0.5}}{2 \cdot a}$

- $a = 0.0000006$
- $b = 0.0003$
- $c = 1.9511$

CF: Cumulative Failures
CS: Cumulative Memory Space
Figure 6. Issues versus Failures

CI=(LN(CF/a))/b
a= 0.248186
b= 0.0107263
CF: Cumulative Failures
CI: Cumulative Issues
Reliability Risk Model Validation (1)

Release \( n \)

- Collect requirements Risk Factor (RF) data during requirements phase.
- Collect Cumulative Failure (CF) data during test phase.
- Use data to estimate coefficients of reliability risk prediction model.
- Predict CF as a function of RF (e.g. size, complexity) during operations phase.
- Validate model against Actual Cumulative Failures (ACF) data.
- Re-estimate model coefficients using actual cumulative failure data.
- This approach has been demonstrated on the Space Shuttle avionics software \([2, 3]\).
Reliability Risk Model Validation (2)

Model Validation: Release n

Requirements → Collect RF → Collect CF → Test

Estimate \( c_1, \ldots, c_n \)

Predict \( CF = f(RF) \)

\( CF \sim ACF \) → Y (Validated)

Re-estimate \( c_1, \ldots, c_n \)

\( n := n + 1 \)

Discard Model

Operations

- CF: Cumulative Failures (Predicted)
- \( c_1, \ldots, c_n \): Coefficients
- RF: Risk Factor (e.g., memory space, requirements issues)
- CF = f (\( c_1, \ldots, c_n, RF \))
- ACF: Actual Cumulative Failures
Reliability Risk Model Application (1)

Release n+1

- Collect requirements Risk Factor (RF) data during requirements phase.
- Predict reliability CF as a function of RF during requirements phase.
- Determine whether Actual Cumulative Failures (ACF) greater than Goal Cumulative Failures (GCF) during requirements phase.
  - If this is the case, Investigate Process and Product for possible corrective action.
- Collect Cumulative Failure (CF) data during test phase.
Reliability Risk Model Application (2)

Model Application: Release n

- **CF**: Cumulative Failures *(Predicted)*
- **c₁, … ,cₙ**: Coefficients
- **RF**: Risk Factor (e.g., memory space, requirements issues)
- **CF = f (c₁, … ,cₙ,RF)**
- **ACF**: Actual Cumulative Failures
- **GCF**: Goal CF

Diagram:

- A: Collect RF
- B: Investigate Process & Product
- CF > GCF
- CF = f (RF)

Y: Yes
N: No
Summary

• *IEEE P1633 \ AIAA R-013A Recommended Practice for Software Reliability* will be revised for complete life cycle Software reliability Engineering process to achieve the following:
  • Provide high reliability in DoD and aerospace safety and mission critical systems.
  • Provide a rational basis for specifying software reliability requirements in DoD acquisitions.
  • Improve the management of reliability risk.
References


