Requirements Volatility Impact: A Measure for All Seasons

Riley Rice
Booz Allen Hamilton
8283 Greensboro Dr.
McLean, VA 22102
USA

Booz | Allen | Hamilton

Contents

- The Problem
- An Elegant Solution
- Adaptations To Different Environments And Needs
- Managing Requirements With This Function

The Problem

- Changes To Requirements
- Insufficient Solution: Requirements Volatility Function
- Industry Experience
- Cascading Rework Factor

The Problem: Changes To Requirements

- One of the most common, high-impact problems in software development is ***requirements volatility after requirements baseline***
- Changes to requirements increase costs and schedule and lower final quality
- If ***V*** = Volatility, ***C*** = Costs and Schedule, and ***Q*** = Quality,

$$V \approx C / Q$$

- Rework adds to cost and schedule
- Retrofitting adaptations produces sub-optimized solutions, decreasing quality

Insufficient Solution

The Requirements Volatility Function

- The standard Requirements Volatility function is a ratio of requirements modifications to the total number of requirements

- But there is a problem here: This does not take into account the increasing effect of such changes as development progresses

$$V = \frac{a + c + d}{T}$$

Where:

a = ***added*** requirement count

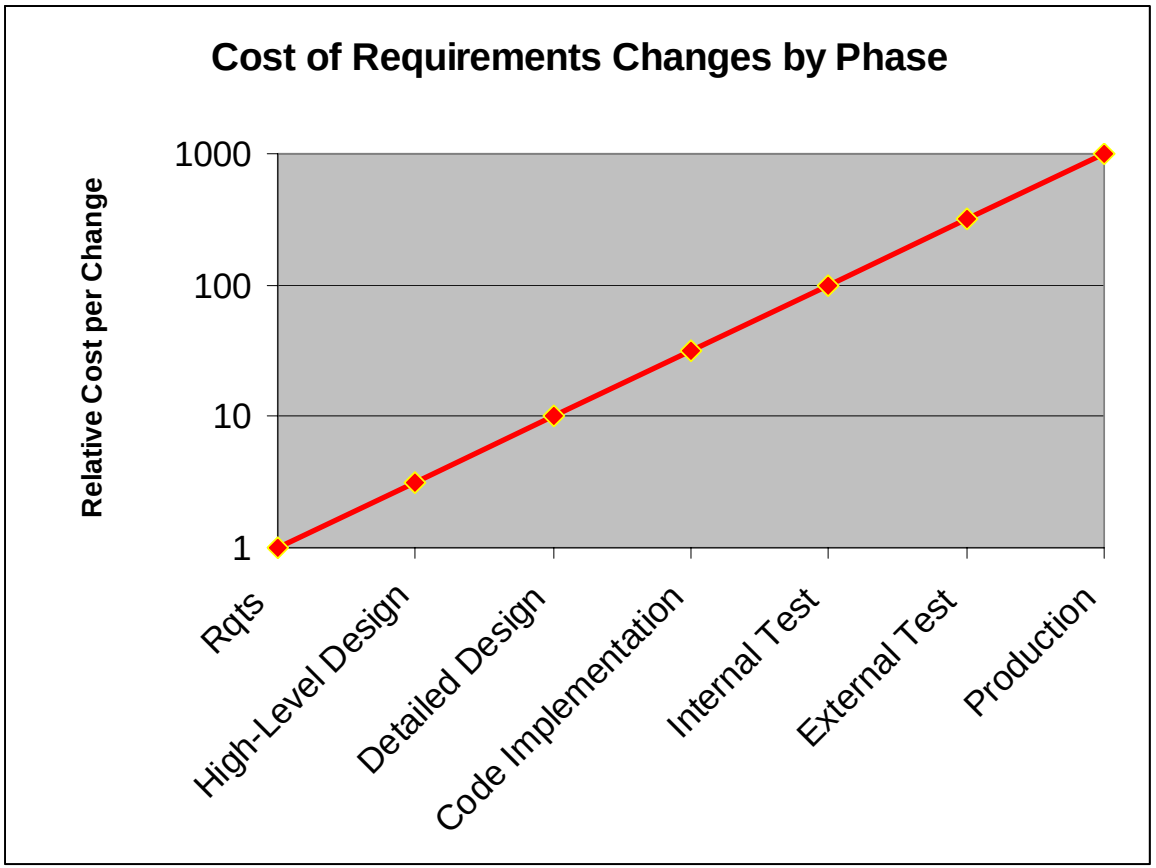
c = ***changed*** requirement count

d = ***deleted*** requirement count

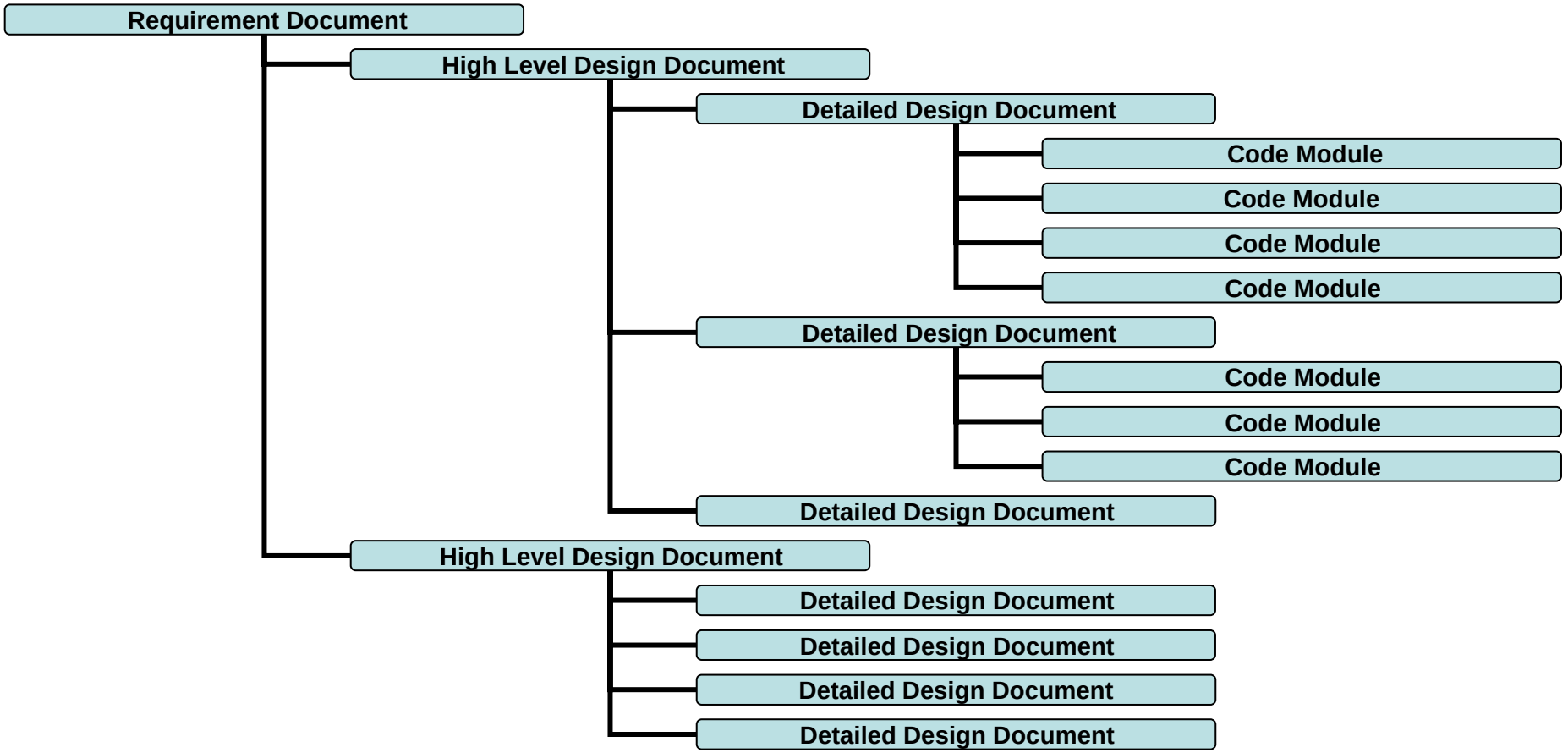
T = ***Total*** requirement count

Industry Experience

- A study by GTE, IBM and Motorola produced the illustrated relationship between the cost of changes to requirements and the phase in which a change occurs
- Note that the scale is logarithmic
- The following slide illustrates why there exists an exponential relationship



Cascading Rework Factor



Standard Formula Only Models Changes During The Requirements Phase

- The Requirements Definition Document must be updated
- During Code, the RDD, all specifications, High-Level Design documents, Detailed Designs, the multitude of code modules and test scripts need to be modified recursively
- During Test, all the above must happen, plus the tests need to be re-planned and re-run, the Build reconfigured, etc.
- In external testing, all installed versions must be reinstalled
- At each phase, many completed artifacts multiply into child artifacts. This is an exponential relationship
- The standard volatility measure does not adjust accordingly

Requirements Volatility Impact Function

An Elegant Solution

The Requirements Volatility Impact Function

- Based on the exponential relationship, we can model this cascade effect by multiplying the standard formula by an Impact factor
- The factor is a power of 10, whose exponent is a function of the number of phases past the start of design against baseline
- This assumes development in anticipation of baseline is minimized

$$V = \frac{a + c + d}{T} \cdot \left[10^{p/2} \right]$$

Where:

a = ***added*** requirement count

c = ***changed*** requirement count

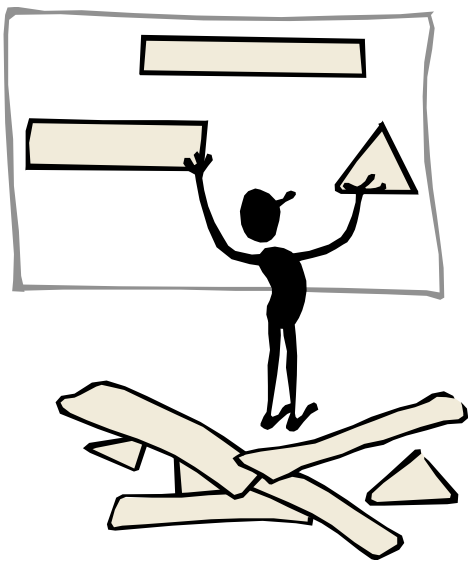
d = ***deleted*** requirement count

T = ***Total*** requirement count

p = number of ***phases*** past baseline

Adaptations To Different Environments and Needs

- Weighting the Kinds of Change
- Pre-Baseline Work
- Non-Waterfall Models



Adaptations: Weighting The Kinds of Change

- As is, the ratio is a unitless number, and will illustrate trends
- Different kinds of changes have predictable effects downstream
- If these are weighted relative to estimated Level of Effort, the ratio becomes a measure of the changes' relative impact on cost
- If the Changed (***c***) term is further weighted by the severity or significance of the change (***s***), the estimate can yield a more precise result

$$\frac{(a \cdot w_a) + (c \cdot w_c \cdot s) + (d \cdot w_d)}{T}$$

where:

a = added

c = changed

d = deleted

w_i = weight of the term type (a, c or d)

T = Total requirements count

s = severity of the change

Adaptations: Pre-Baseline Work

- Some projects consider all requirement "churn" (a + c + d) appropriate *during the Requirements phase*
- To alter the formula for this, subtract 1 from the power
- p = 0 during the Requirements phase, thus $10^{0/2} = 1$
- [1 – 1] = [0], canceling out that phase's churn without significantly affecting other phases' impact factor value

$$V = \frac{a+c+d}{T} \cdot \left[10^{p/2} - 1 \right]$$

Where:

a = ***added*** requirement count

c = ***changed*** requirement count

d = ***deleted*** requirement count

T = ***Total*** requirement count

p = number of ***phases*** past baseline

Adaptations: Non-Waterfall Models

- Large development projects employ an incremental delivery model in which successive ***builds*** accumulate to produce a release
- Each ***build*** follows its own waterfall development process, and can be measured for volatility
- The ***release***, however, cannot be said to be in one “phase” at a given time, so the model must be adapted to combine the component builds’ volatility components
- Spiral development models (e.g., RUP) use repeated ***iterations*** of a process
- Each ***iteration*** involves steps similar to the earlier phases of a waterfall model
- By defining ***p*** appropriately for each step in an iteration, the formula can be used per iteration, as for build in the previous model, then combined, as for releases

Adaptations: Combining Component Volatility Measures Into A Larger Whole

- Multiply each component's computed volatility value by the proportion of that component's requirements to the whole
- The sum of these products is the volatility impact measure for the whole

$$V = \sum_{i=1,2,\ldots} \left(V_i \cdot \frac{R_i}{R_T} \right)$$

Where:

V = Volatility impact computation

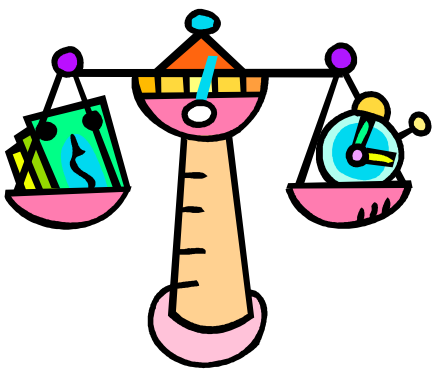
R = Requirements count

i = individual component

T = total count

Managing Requirements With This Function

- Estimate Impact
- Manage Client Expectations
- Set Limits
- Integrate Early and Late Process Management



Estimate Impact

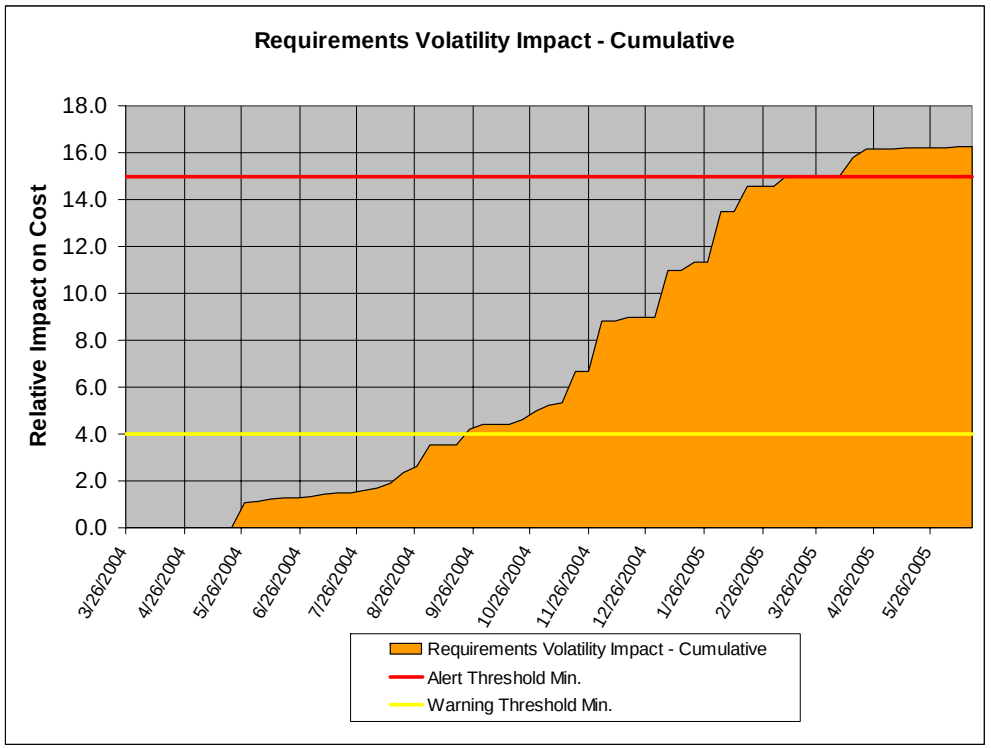
- The model allows you to perform “What-Ifs”, to estimate cost and schedule implications of projected changes to requirements in a release after baseline.
- Actual costs can be used to tune the model through correlations to the existing model’s functional value. The adjusted weights will yield an increasingly accurate estimate.
- Thresholds or control limits can be established based on contractual limits on changes, and used to make decisions on whether to include a set of changes in the current or next release.

Manage Client Expectations

- Visible trends have a visceral effect on viewers
- Use color carefully to express intuitive meaning

  Careful: colorblindness is common and colors change with media and resolution changes

- This kind of graph has changed client behavior. They respected the thresholds, once they were explained



Set Limits

- All Software Development expects a certain amount of requirements change after baseline. Good requirements management minimizes this
- Identify the percent of your estimate that accounts for an expected, acceptable amount of change after baseline. That is your Warning threshold (yellow line in the previous graph)
- Contractually agree with the client (internal or external) on the maximum amount of allowable further change, after which the contract must be renegotiated or a new release planned. That is your Alert threshold (red line in the previous graph)
- This could result in only one threshold, of course
- The graph will illustrate progress toward the limits

Integrate Early And Late Process Management

- One common complaint in software development is that the expertise and issues of the late process experts are not brought into play during the requirements phase
- This causes quality and cost problems by introducing late-process requirements changes, when the issues assert themselves late in the game
- Managing with the Requirements Volatility Impact measure has the effect of bringing the Test Managers and the O&M / Support Managers into the Requirements phase. If that was not happening before, it will when the Volatility model is set up and used

Summary

Summary

- Good measurements provide unambiguous information to answer management questions and support goal-directed decision-making
- Requirements volatility is a high-cost problem, whose solution has high-ROI potential
- Requirements Volatility Impact is relatively easy to implement
- This measure is used from the beginning of a project throughout the life cycle – that is its point!
- Software development can be a dark and surprising world. Use this measure to turn on the lights while you drive to success

Contact Information

Riley Rice

8283 Greensboro Drive
McLean, VA 22102
(703) 902-6781

rice_riley@bah.com
www.boozallen.com