Testimation

Applicator Suite

User Instructions | v1.0

for Use Case / User Story | TEstimator[1]
| TEstimator[2]
| Testimator[3]
| REstimator
| @Risk

Author/s | Testimation Team

Engineering tools for the science of estimation & risk management

1. Introduction

**** **Important Note** ****
The purpose of this document is to act as a companion to the Applicator Suite. Please refer to it as required. For brevity & practicality, most images & screenshots within the Applicator Suite of Tools have been omitted from this document due to the intended companion relationship.
**** **End Note** ****

Have you ever wondered why your computer crashes so frequently but your car always starts? The answer is simple. Humans have been making things for many thousands of years, but the commercial software industry is only decades old. We have leveraged off this hardware manufacturing experience to develop a suite of products that are predictive & true risk management tools. At Testimation, we believe that risk management should rarely be about qualitative high-level statements, rather, it should be about hard quantitative numbers expressed as a percentage value. For example; "it's too risky" is a very different statement to "there's a 23% probability of success". The former is common practice in risk management, the latter is Testimation.

We are passionate about software quality & engineered solutions. We believe that for far too long in the Information Technology **(IT)** sector, emphasis has been on development costs rather than product quality. This emphasis has arisen, not because organizations are mean with money, but because a scientific approach to inform business sponsors of risk has not been mainstream in **IT**. However, in the engineering sector, the specifications of failure probabilities are commonplace. Stories appear too often in the media about catastrophic software failures inconveniencing or endangering customers. We believe that the primary cause of this is due to business sponsors simply not being aware of the risk of failure because it has (*probably*) never been presented to them as a percentage value. Typically, only "words" are used to describe the risk of software failure. The problem with this approach is that it is highly subjective & depends as much upon the person receiving the information, as it does upon the person sending it. Imagine if a pharmaceutical company released experimental cures based upon the concept of "it should be fine", rather than statistically meaningful clinical trials; how ready would you be to give their drugs to your children?

At Testimation, we want to change things ……

When risk is presented as a numerical probability of failure, human reaction to this information is far more sober than when it is presented in qualitative forms only. If your **IT** providers or procurement advisers are giving you "words" & not numbers, then you should turn to Testimation. "Words" are easy to produce, they require little effort & one does not need to be an expert to "talk" (*e.g. sales people*); but only experts will give you hard numbers from the application of engineering methodologies. If your software development estimates are being delivered to you without the number of tests being specified, or without the probability of finding Defects being expressed with hard numbers, then you are playing Russian roulette & you may need to re-define success throughout the Project Life-Cycle to make it appear to stakeholders that you have delivered successfully; this occurs far more commonly than you may realise. However, with our tools & services, you can design the Quality Assurance strategy that suits your budget & time constraints, with confidence that you're aware of the risk & the potential impact upon your customers, users & business.

The Testimation Team brings formalized & verified engineering techniques to the disciplines of estimation & risk management; specifically to the software development industry. It utilizes the same Quality Assurance principles as the manufacturing sector & high performance industries such as aerospace, commercial aviation & medical research. Pharmaceutical companies do not release products to the market unless the potential failure rate is numerically understood & the risk mitigated. So why is society so prepared to release software to users when the defect-free confidence is only described with words (*if at all*) & not hard numbers? …. We have addressed this question by developing engineering tools for the science of estimation & risk management. Our suite of tools empowers clients & service providers to present scientific arguments supporting their Quality Assurance & risk management profiles. Others talk of risk in subjective & salesmanship language; we quantify it with hard science & hard numbers. Our products & services offer four key features:

1. The worlds first suite of scientifically formulated, web-delivered estimation tools
2. The number of tests required to be executed for the level of Quality Assurance specified by project governance
3. The probability of finding defects in accordance with project governance specifications
4. Development, testing & project management effort

2. The Big Picture

The Testimation Team has decades of experience in Hardware + Software Quality Assurance, Process Re-Engineering & Risk Management across multiple industries, such as Information Technology, Business Technology, Manufacturing, PetroChem, Utilities, Food, Government, Financial, Insurance, Construction, Aluminium & Retail. From our experience, we have learned that some industries possess significantly greater levels of Quality Assurance & Risk Management controls than others. For example, in 1969, the United States of America placed men upon on the moon with less computing power on the Apollo 11 command module than exists on your smartphone today. This historical fact speaks volumes about "what can be" & "what is" within the **IT** sector. The harsh reality of 21st century **IT** is that, although our technology has dramatically advanced over recent history, good & healthy engineering practices have (*largely*) not been introduced or totally ignored, sector wide, across **IT**. This fact is surprisingly easy to prove; for example, how many software development projects have you seen that have ever referenced the relevant international standards in relation to the product being built, or the development process being undertaken? If you're honest with yourself, you'll probably answer "never", or at least, "very rarely". Many international software standards exist & many more are being developed as you read this document.

Whilst it is exceedingly common practice in the mainstream engineering sector to design solutions according to general & specific standards, within the **IT** sector this does not seem to occur with any significant frequency. The primary reason for this may be because software developers are often not formally trained or qualified engineers & therefore, lack some of the fundamental design skills acquired during the university indoctrination process. One is not required to be university educated & examined to be titled - "a developer", but one is required to possess these attributes in order to be titled "an engineer" (*globally*). In fact as a generalization, in many places around the world, the only professions whose signatures constitute a legal document are doctor, lawyer & engineer; all of which require formalized & recognized university training. So why is this important? It is important because repeatable processes require standardization, & standardization comes from national & international bodies of acceptance such as the International Organization for Standards **(ISO)**. For example, imagine that you had to start your car in a different manner each time you intended to use it. Imagine that one time you use the key, the next time you tap on a rear wheel, & after that you must do something different each time you wish to start the car. The lesson here is that effective design, development, delivery & User Acceptance models require standardization, & standardization is a hallmark characteristic of a formal engineering design process. A process involves the execution of actions & one of the very first required actions is estimation. The Software Development Life-Cycle is no exception; a significant portion of it involves;

The Software Testing Life-Cycle .... It all begins with an estimate

1. Why is estimation important to a client?
    - Because this is when a client sets the budget

2. Why is estimation important to a service provider?
    - Because this is when profit or loss is defined

3. When can our Product Suite be used?
    - In situations where the quality capability should be known
        1. Bids / Tenders, Projects, Audits, Operations, UAT
        2. Assessments of Service Providers & Vendor Software

4. What does using our Product Suite mean for you?

Users	Uses
1. Board of Directors	1. Save money by forecasting accurately & efficiently
2. CEO's, CIO's, CTO's, CPO's	2. Define, manage & mitigate risk properly
3. TM's, DM's, PM's, RM's	3. Design Quality Assurance strategies fitting your budget
4. Business Sponsors	4. Conformity, repeatability & transparency of process
5. Procurement	5. The subjectivity plaguing estimation has largely evaporated
6. System Integrators	6. Estimates are Engineering based
7. Security Agencies	7. Facilitates Continuous Improvement Processes
8. Software Houses	8. Respond rapidly to changes in scope
9. Corporations	9. Scale from an Historical Solution to a Projected Solution
10. Government	10. Calculate the probability of finding Defects
11. Military	11. Calculate the required number of tests

3. The Applicator Product Suite

A. Composition & Architecture

The Applicator Product Suite incorporates five (5) key Use Cases (User Stories). Each of these five (5) User interactions executes a distinctly different process. The Product Suite is composed of the following:

1. TEstimator^{1} | estimates the minimum & maximum number of tests[i]

2. TEstimator^{2} | estimates the exact number of tests when scaling from an Historical Solution to a Projected Solution

3. TEstimator^{3} | estimates the exact number of tests when scaling from an Historical Solution to a Projected Solution

4. REstimator (*Stand Alone*) | estimates the minimum number of Regression tests

5. @Risk (*Stand Alone*) | User Acceptance Testing **(UAT)** Risk Visualisation

Additional embedded capabilities are:

1. AEstimator | compares two estimates (*side by side*)

2. PEstimator | estimates the test effort in Man-Days

[i] *i.e.* The lower & upper limits

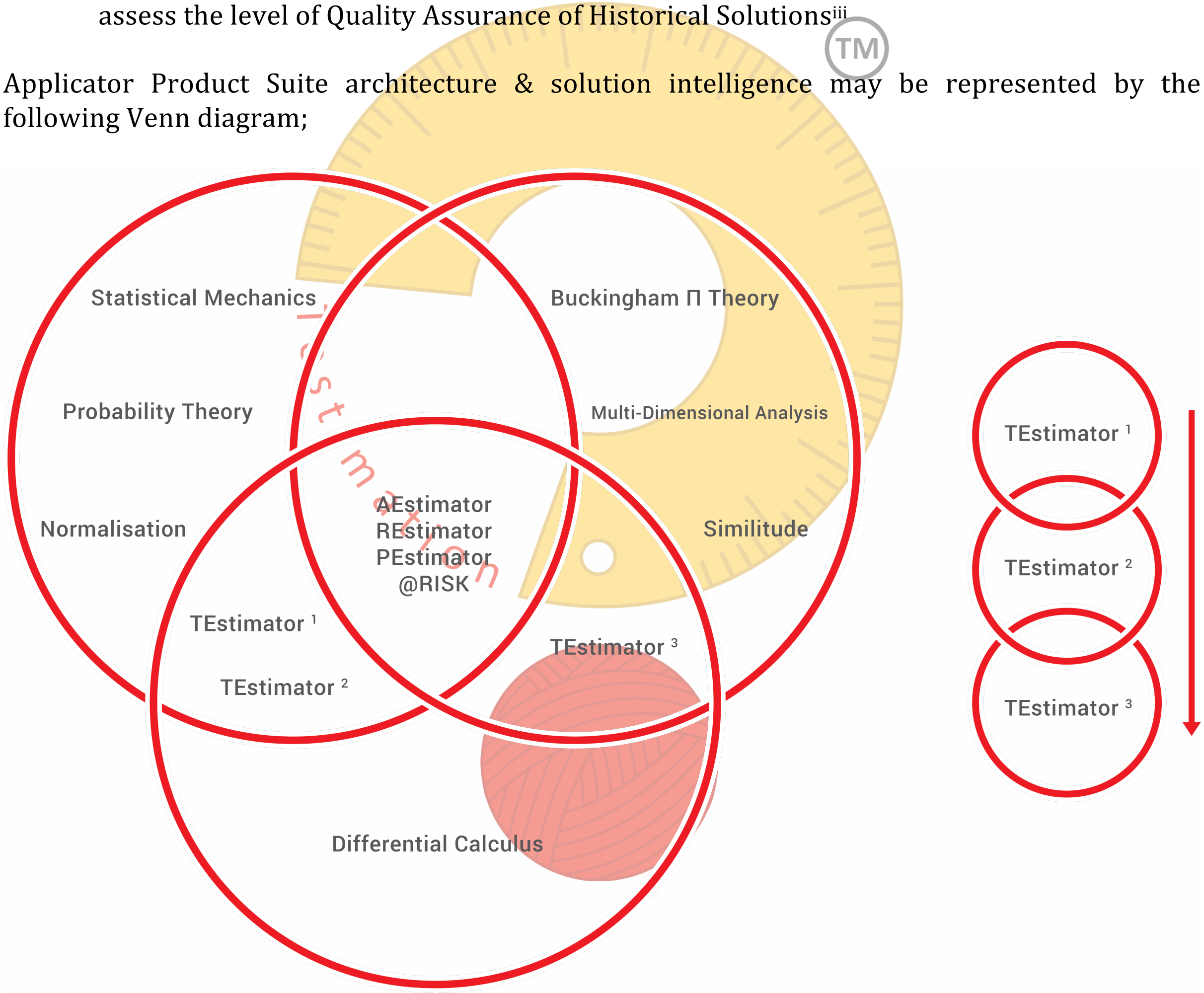
Key differences between TEstimator[1,2,3] are:

1. TEstimator[1] requires one (1) input
    - Scope only
2. TEstimator[2,3] require three (3) inputs
    1. Historical Scope
    2. Historical Test Cases
    3. Projected Scope

Key differences between TEstimator[2,3] are:

1. TEstimator[2] can scientifically assess the level of Quality Assurance of Historical Solutions, but is less flexible[ii] than TEstimator[3]
2. TEstimator[3] provides greater flexibility than TEstimator[2], but cannot scientifically assess the level of Quality Assurance of Historical Solutions[iii]

Applicator Product Suite architecture & solution intelligence may be represented by the following Venn diagram;



[ii] *i.e.* The Quality Override function is optionally activated
[iii] *i.e.* The Quality Override function is mandatorily activated

B. Feature List

What is a Feature & Why?

A Feature is any Input or Output. They are Feature because they do not occur naturally; a person is required to physically design, build & implement them. Even if the Features are repeated, it takes human will & effort to repeat them on different screens or reports. Everything, without exception, in all Applications, require Lines-of-Code to be created, copied or modified to perform the functions we desire. The choice of Input & the form of Output requires human decision-making in terms of need & suitability. For these reasons, Inputs & Outputs are Features because they require the mind of a human being for them to become tangible.

1. TEstimator[1]

A. User interfaces
   1. Quick Estimate View
   2. Lean Estimate View
   3. Basic View
   4. Advanced View

B. User Input Metric
   5. Functional Processes
   6. Use Cases
   7. User Stories
   8. Test Scenarios

C. Users can specify prioritization & risk as part of their testing strategy
   9. In terms of Calibration Factor[iv]
   10. In terms of Risk Quotient[v]

D. Progression Testing
   The minimum number of recommended Progression Test Cases
   11. In terms of Dynamic Information Tests
   12. In terms of Test Cases
   The maximum number of recommended Progression Test Cases
   13. In terms of Dynamic Information Tests
   14. In terms of Test Cases

E. Regression Testing
   15. The optimal number of manual Regression Test Cases
   16. The optimal number of automated Regression Test Cases

F. Risk Management
   17. Risk Exposure[vi] {β}
   18. Risk Mitigation[vii] {α}
   19. Test Coverage of Development (%)
   20. System Unaffected by Development (%)
   21. Test Case complexity (%)
   22. Defect complexity (%)

G. General
   23. Graphically displays Quality Assurance architecture & strategy {α, β}
   24. Random configuration generator for User training purposes

---

[iv] *i.e.* Test Case priority
[v] *i.e.* The level of Acceptable Risk
[vi] % System-wide impact of Development {β}
[vii] % Probability of finding Defects {α}

2. TEstimator²

H. User Input Metric
 25. Functional Processes + Test Cases
 26. Use Cases + Test Cases
 27. User Stories + Test Cases
 28. Test Scenarios + Test Cases

I. Test Case Ceiling Guide
 - Historical Solutions
 29. 99% Coverage Confidence
 30. 99.6% Coverage Confidence
 31. 99.84% Coverage Confidence
 32. 99.93% Coverage Confidence
 33. 99.97% Coverage Confidence

 Projected Solutions
 34. 99% Coverage Confidence
 35. 99.6% Coverage Confidence
 36. 99.84% Coverage Confidence
 37. 99.93% Coverage Confidence
 38. 99.97% Coverage Confidence

J. Progression Testing
 39. The minimum number of recommended Progression Test Cases
 40. The maximum number of recommended Progression Test Cases

K. Risk Management
 - Historical Solutions
 41. Risk Exposure[viii] {δ}
 42. Risk Mitigation[ix] {γ}
 43. Test Coverage of Development (%)
 44. System Unaffected by Development (%)
 45. Test Case complexity (%)
 46. Defect complexity (%)
 47. Test Approach undertaken in terms of Calibration Factor[x]
 48. Test Approach undertaken in terms of Quotient[xi]
 49. Assessment of the level of Quality Assurance of the Historical Solution
 - Projected Solutions
 50. Risk Exposure[xii] {ζ}
 51. Risk Mitigation[xiii] {ε}
 52. Test Coverage of Development (%)
 53. System Unaffected by Development (%)
 54. Test Case complexity (%)
 55. Defect complexity (%)
 56. Test Approach to be undertaken in terms of Calibration Factor[xiv]
 57. Test Approach to be undertaken in terms of Quotient[xv]
 58. Assessment of the level of Quality Assurance of Historical Solutions

---

[viii] % System-wide impact of Development {δ}
[ix] % Confidence that all Defects were found {γ}
[x] *i.e.* Test Case priority
[xi] *i.e.* the level of Acceptable Risk
[xii] % System-wide impact of Development {ζ}
[xiii] % Confidence that all Defects will be found {ε}
[xiv] *i.e.* Test Case priority
[xv] *i.e.* The level of Acceptable Risk

L. General

59. Users can specify Quality Override as part of their interpretation of the testing strategy in the Historical Solution[xvi]
60. Users can specify Risk Override as part of their testing strategy for the Projected Solution[xvii]
61. Graphically displays the Quality Assurance architecture & strategy, of Projected Solutions & Historical Solutions side by side {δ, γ, ζ, ε}
62. Intelligence preventing unrealistic or impossible User-Defined Historical Solutions influencing Projected Solutions (*via WorkFlow prevention logic*)[xviii]
63. Random configuration generator for User training purposes

3. TEstimator[3]

M. User Input Metric

64. Thirteen variants of User input metric available[xix]
65. User-Defined Notes field (*Historical Solution*)
66. User-Defined Notes field (*Projected Solution*)

N. Progression Testing

67. The number of recommended Progression tests for new Projected Solutions based upon Historical Solutions

O. Risk Management

- Historical Solution

68. Defect-Free Confidence: see {η}

- Projected Solution

69. Defect-Free Confidence: see {θ}

P. General

70. Graphically displays the Quality Assurance architecture & strategy, of Projected Solutions & Historical Solutions (*overlaid*): see {η, θ}
71. Users can specify Quality Override as part of their interpretation of the testing strategy in the Historical Solution[xx]
72. Users can specify Risk Override as part of their testing strategy for the Projected Solution[xxi]
73. Intelligence preventing unrealistic or impossible User-Defined Historical Solutions influencing Projected Solutions[xxii]
74. Random configuration generator for User training purposes

4. REstimator

Q. User Input Metric

75. Test Case Repository **(TCR)** Size (*i.e.* the overall number of Test Cases)
76. % Estimated impact upon the Test Case Repository by the Project
77. User-Defined Sampling Confidence %
78. Average number of Dynamic Information Tests **(DIT's)** per Test Case

---

[xvi] *i.e.* the level of Executed Risk
[xvii] *i.e.* The level of Acceptable Risk
[xviii] Users are prevented from moving to the AEstimator screen
[xix] *e.g.* Requirements, lines of code, security threats, business scenarios etc.
[xx] *i.e.* the level of Executed Risk
[xxi] *i.e.* The level of Acceptable Risk
[xxii] *i.e.* via Normalization of the estimate at the PEstimator screen

R. Risk Management

- Sampling Distribution

79. Number of Functional Processes associated with the User-Defined Sampling Distribution Confidence
80. Number of Test Cases associated with the User-Defined Sampling Distribution Confidence
81. Risk Mitigation: *i.e.* system-wide coverage of the User-Defined Sampling Distribution[xxiii] {ν}

S. General

82. Number of Functional Processes associated with the existing Test Case Repository
83. Number of Functional Processes impacted by the Project changes
84. Number of existing **TCR** tests impacted by Project changes
85. Number of **TCR** tests associated with the User-Defined Sampling Distribution Confidence
86. System-wide coverage of existing **TCR**: see {κ}
87. System-wide coverage of existing **TCR** impacted by Project changes: see {λ}
88. Graphically displays Quality Assurance & Risk profiles of the **TCR** & the User-Defined Sampling Distribution sub-set {ν, κ, λ}
89. Random configuration generator for User training purposes

5. @Risk

T. User Input Metric

90. Functional Processes
91. Use Cases
92. User Stories
93. Test Scenarios
94. User Acceptance Tests (*i.e.* Test Cases)

U. Risk Management

95. Defect complexity (%)
96. % Probability of User Acceptance Testing **(UAT)** finding Defects[xxiv] {ρ}
97. % Functional Processes impacted by Project changes[xxv] {β}
98. % Functional Processes impacted by Project changes[xiii] & covered by **UAT**[xxvi]

V. General

99. Graphical display {α, β}
100. Random configuration generator for User training purposes

6. AEstimator

W. User Input Metric

101. Test Cases

X. Risk Management

102. TEstimator^1 Risk Quotient
103. TEstimator^2 Risk Quotient
104. Alternative Estimate Risk Quotient
105. TEstimator^1 Calibration Factor
106. TEstimator^2 Calibration Factor
107. Alternative Estimate Calibration Factor

[xxiii] % Probability of finding Defects {ν}
[xxiv] *i.e.* System-wide {ρ}
[xxv] % System-wide impact of Development {β}
[xxvi] Test Coverage of Development

108. TEstimator[1] **DIT's** per Test Case
109. TEstimator[2] **DIT's** per Test Case
110. Alternative Estimate **DIT's** per Test Case
111. % TEstimator[1] Risk Exposure: see {β}
112. % Alternative Estimate Risk Exposure: see {β}
113. % TEstimator[1] Risk Mitigation[xxvii] {α}
114. % Alternative Estimate Risk Mitigation[xxviii] {ι}
115. % TEstimator[2] Risk Exposure {ζ}
116. % Alternative Estimate Risk Exposure {ζ}
117. % TEstimator[2] Risk Mitigation[xxix] {ε}
118. Alternative Estimate Risk Mitigation[xxx] {ι}
119. Test Coverage of Development (%) by TEstimator[1]
120. Test Coverage of Development (%) by Alternative Estimate (*TEstimator[1] screen*)
121. Test Coverage of Development (%) by TEstimator[2]
122. Test Coverage of Development (%) by Alternative Estimate (*TEstimator[2] screen*)
123. Side by side comparison between TEstimator[1] & the Alternative Estimate
124. Side by side comparison between TEstimator[2] & the Alternative Estimate
125. % Similarity between the User defined estimate & the alternative source by Area
126. % Similarity between the User defined estimate & the alternative source by Test Case population

Y. General

127. Graphically displays Quality Assurance between the User defined estimate & the alternative source
128. Random configuration generator for User training purposes

7. PEstimator

Z. User Input Metric

129. Progressive Test Case Design Throughput
130. Progressive Test Case Modification Throughput
131. Progressive Test Case Execution Throughput
132. Regressive Test Case Execution Throughput
133. Quality Assurance Personnel Resourcing
134. % Manual Progressive Testing
135. % Automated REstimator Test Cases
136. % Anticipated Test Case Failures

AA. Risk Management

- TEstimator[1]

137. Functional Processes
138. Test Cases
139. Risk Exposure[xxxi] {β}
140. Risk Mitigation[xxxii] {α}
141. Risk Mitigation[xxxiii] {ν}

- TEstimator[2]

142. Functional Processes
143. Test Cases

[xxvii] % Probability of finding Defects {α}
[xxviii] % Probability of finding Defects relating to TEstimator[1] {ι}
[xxix] % Probability of finding Defects {ε}
[xxx] % Probability of finding Defects relating to TEstimator[2] {ι}
[xxxi] % System-wide impact of Development {β}
[xxxii] % Probability of finding Defects via Progression Testing {α}
[xxxiii] % Probability of finding Defects via Regression Testing {ν}

144. Risk Exposure[xxxiv] {ζ}
145. Risk Mitigation[xxxv] {ε}
146. Risk Mitigation[xxxvi] {ν}

- TEstimator^{3}

147. Functional Processes
148. Test Cases
149. Risk Exposure[xxxvii] {ξ}
150. Risk Mitigation[xxxviii] {ο}
151. Risk Mitigation[xxxix] {ν}

- REstimator

152. Functional Processes
153. Test Cases

BB. Quality Assurance Breakdown

- TEstimator^{1}

154. Progressive TEstimator Test Cases to be Designed
155. Regressive TEstimator Test Cases to be Modified
156. TEstimator Test Cases to be Executed
157. REstimator Test Cases to be Executed
158. Anticipated Defects
159. Total Executable Test Cases

- TEstimator^{2}

160. Progressive TEstimator Test Cases to be Designed
161. Regressive TEstimator Test Cases to be Modified
162. TEstimator Test Cases to be Executed
163. REstimator Test Cases to be Executed
164. Anticipated Defects
165. Total Executable Test Cases

- TEstimator^{3}

166. Progressive TEstimator Test Cases to be Designed
167. Regressive TEstimator Test Cases to be Modified
168. TEstimator Test Cases to be Executed
169. REstimator Test Cases to be Executed
170. Anticipated Defects
171. Total Executable Test Cases

CC. Project Management

- TEstimator^{1}

172. Man-Days
173. Man-Weeks
174. Man-Months

- TEstimator^{2}

175. Man-Days
176. Man-Weeks
177. Man-Months

- TEstimator^{3}

178. Man-Days
179. Man-Weeks
180. Man-Months

[xxxiv] % System-wide impact of Development {ζ}
[xxxv] % Confidence that all Defects will be found via Progression Testing {ε}
[xxxvi] % Probability of finding Defects via Regression Testing {ν}
[xxxvii] % System-wide impact of Development {ξ}
[xxxviii] % Confidence that all Defects will be found via Progression Testing {ο}
[xxxix] % Probability of finding Defects via Regression Testing {ν}

DD. General

181. Graphically displays project Quality Assurance & Risk profiles {α, β, ν}
182. Graphically displays project Quality Assurance & Risk profiles {ζ, ε, ν}
183. Graphically displays project Quality Assurance & Risk profiles {ξ, ο, ν}
184. Random configuration generator for User training purposes

EE. Total Feature Count

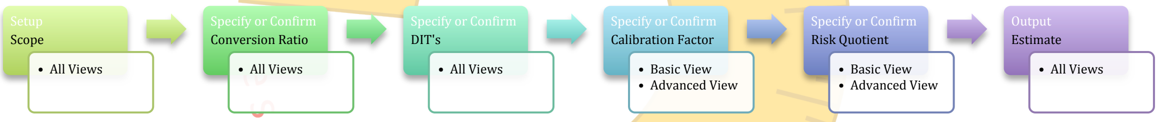
1. Number of TEstimator^1 Report features = 24
2. Number of TEstimator^2 Report features = 39
3. Number of TEstimator^3 Report features = 11
4. Number of REstimator Report features = 15
5. Number of @Risk Report features = 11
6. Number of AEstimator Report features = 28
7. Number of PEstimator Report features = 56
8. Total Number of Report features = 184
9. Number of Screen features ≅ Total Number of Report features ≅ 184
10. **Total Number of Applicator Product Suite features = 368+**

4. How to execute a TEstimator[1] Estimate

A. Applicability

Uses	User Interfaces	Users
1. Bids 2. Tenders 3. Risk Assessment	1. QuEstimate View *Quick Estimate* 2. LEstimate View *Lean Estimate* 3. Basic View 4. Advanced View	1. Board of Directors 2. Chief Executive Officers 3. Chief Technology Officers 4. Security Agencies 5. Service Providers 6. System Integrators 7. Project Managers 8. Development Managers 9. Quality Assurance Managers 10. Corporations 11. Government

B. Estimation Process



C. How to execute a QuEstimate or LEstimate

1. Inputs

1. **Setup Scope**
   1. Select Functional Processes, Use Cases, User Stories or Test Scenarios from the dropdown menu
   2. Specify the desired value by using the slider

2. Configuration

1. **Specify or Confirm Conversion Ratio**
   1. This is the average number of Functional Processes you expect to have per Use Case, User Story or Test Scenario
   2. The default value for Use Cases & User Stories is 5, the default value for Test Scenarios is 1
   3. If you do not know the Conversion Ratio, use the default value
2. **Specify or Confirm Dynamic Information Tests (DIT's) per Test Case**
   1. This is the average number of Dynamic Data Fields you expect to check per Test Case
   2. If you do not know the **DIT's** per Test Case, use the default value

D. How to execute an Estimate in Basic View

1. See "How to execute a QuEstimate or LEstimate"
2. Select a Calibration Factor Set-Point. The Calibration Factor **(CF)** is part of the System-Wide Test Approach, not just the code being Developed or Modified for the Project
    1. **CF** = 0.25 Targets Critical Priority Tests
    2. **CF** = 0.5 Targets Critical & High Priority Tests
    3. **CF** = 0.75 Targets Critical, High & Moderate Priority Tests
    4. **CF** = 1 Targets Critical, High, Moderate & Low Priority Tests
    5. **CF** = 1.25 Targets all Tests + 25% Redundancy
    6. **CF** = 1.5 Targets all Tests + 50% Redundancy
    7. **CF** = 1.75 Targets all Tests + 75% Redundancy
    8. **CF** = 2 Targets all Tests + 100% Redundancy
    9. See Framework Probabilities Information to assist with decisions
3. Select Risk Quotient Set-Point
    1. The Risk Quotient allows a User to target an acceptable level of risk in order to minimise expenditure
    2. See Framework Probabilities Information to assist with decisions[xl]

E. How to execute an Estimate in Advanced View

1. See "How to execute a QuEstimate or LEstimate"
2. See "How to execute an Estimate in Basic View"
3. Click the Calibration Factor Expand button (*if required*)
    1. The User has access to the Calibration Factor Apply Critical Value button
    2. This feature may be invoked to compute the minimum number of Test Cases providing 100(%) Test Coverage of Development.
    3. This feature provides a Developed or Modified code-only specific Test Case sizing solution when $\alpha = \beta$
4. Click the Risk Quotient Expand button (*if required*)
    1. The User has access to the Risk Quotient dial
    2. The User has access to the Risk Quotient Apply Critical Value button
    3. These features provide the User with the ability to reduce Test Case sizing by controlling risk
    4. These features may be invoked to compute the minimum number of Test Cases providing 100(%) Test Coverage of Development
    5. These features provide a Developed or Modified code-only specific Test Case sizing solution when $\alpha = \beta$

[xl] Particularly: The probability of finding at least one Defect

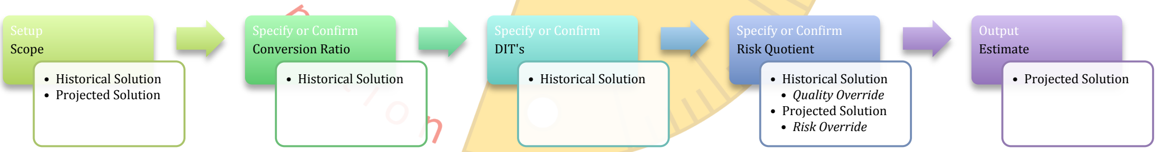
5. How to execute a TEstimator² Estimate

A. Applicability

1. The Projected Solution is scaled from an Historical Solution via the Defect-Free Confidence
2. The Defect-Free Confidence is calculated scientifically by default, or may be overridden & assigned a value by the User

Uses	Users
1. In-house Projects	1. Board of Directors
2. Bids	2. Chief Executive Officers
3. Tenders	3. Chief Technology Officers
	4. Security Agencies
	5. Service Providers
	6. System Integrators
	7. Project Managers
	8. Development Managers
	9. Quality Assurance Managers
	10. Corporations
	11. Government
	12. Military

B. Estimation Process



C. Procedure

1. Inputs

1. **Setup the Historical Solution**
   1. Select Functional Processes, Use Cases, User Stories or Test Scenarios from the dropdown menu
   2. Specify the desired value by using the slider
   3. Input the number of Executed Test Cases
2. **Setup the Projected Solution**
   1. Specify the desired value by using the slider

2. Configuration

1. **Specify Conversion Ratio**
    1. This is the average number of Functional Processes you expect to have per Use Case, User Story or Test Scenario
    2. The default value for Use Cases & User Stories is 5; the default value for Test Scenarios is 1
    3. If you do not know the Conversion Ratio, use the default value
2. **Specify Dynamic Information Tests (DIT's) per Test Case**
    1. This is the average number of Dynamic Data Fields you expect to check per Test Case
    2. If you do not know the DIT's per Test Case, use the default value
3. **Configure Quality Override**
    1. The default configuration at page load is a scientific assessment of the Historical Solution. At this juncture, the User must decide to override the scientific assessment (*or not*)
    2. If the User decides to utilise the scientific assessment; no further action is required
    3. If the User decides to override the scientific assessment, the User must assign a Defect-Free Confidence to the Historical Solution by utilising the Set-Point Slider or Risk Quotient Dial[xli]
    4. If the User wishes to return to the scientific assessment, it may be achieved by clicking the "Apply Critical Value" button underneath the Risk Quotient Dial
4. **Configure Risk Override**
    1. By inspection of the graphs & general Project considerations, the User decides the appropriate configuration for the Projected Solution
    2. If the Project is constrained by time or budget, the User may wish to balance the appetite for Testing Effort (*Test Cases*) with an acceptable level of Defect-Free Confidence
    3. The default Risk Quotient value is 0% (*yielding a Defect-Free Confidence of 99%*). If the User wishes to focus on the Projected Code Changes only & minimise the Test Effort, the "Apply Critical Value " button underneath the Risk Quotient Dial may be utilised; the effect of this action is reflected in the graphs
    4. If the Critical Value for the Risk Quotient appears negative, this indicates that the default scientific assessment of the Historical Solution has been overridden by the User & it is likely that insufficient Testing was performed. In this case, the User should consider utilising an alternative Historical Solution as a frame of reference for the Projected Solution

---

[xli] Examples

1. The User believes that the Historical Solution yielded an 80.22% probability that no Defects exist (*system-wide*); therefore, the User selects a Risk Quotient Set-Point of 50%. This relates to an 80.22% probability that all system-wide Critical Priority Tests were executed & passed successfully
2. The User believes that the scientific assessment of Defect-Free Confidence is too cautious & recalls that the Historical Solution was implemented without significant incident. The User believes that the system-wide Defect-Free Confidence is approximately 95%. Consequently, the User selects a Risk Quotient Set-Point of 23%
3. The User believes that the scientific assessment of Defect-Free Confidence is too cautious & recalls that the Historical Solution was implemented without significant incident. The User believes that the system-wide Defect-Free Confidence is approximately 99%. Consequently, the User selects a Risk Quotient Set-Point of 0%

3. Analysis | Historical & Projected Solutions

1. The Graphs generated communicate
    1. Risk Exposure
        1. Historical Solution
            1. The proportion of system-wide functionality that was Developed or Modified
        2. Projected Solution
            1. The proportion of system-wide functionality to be Developed or Modified
    2. Risk Mitigation
        1. Historical Solution
            1. The probability that the approach to Testing found all Defects
        2. Projected Solution
            1. The probability that the approach to Testing will find at least one Defect
    3. Test Coverage of Development
        1. Historical Solution
            1. The proportion of Development that was covered by the Test Approach
        2. Projected Solution
            1. The proportion of Development to be covered by the Test Approach
    4. The Targeted Distribution for the Historical Solution[xlii]
    5. The Targeting Distribution for the Projected Solution[xliii]
2. Test Execution Approach communicates
    1. Historical Solution
        1. Via Calibration Factor
            1. The composition of Critical, High, Moderate & Low Priority Test Cases, which should have been Executed
        2. Via Quality Override
            1. The composition of Critical, High, Moderate & Low Priority Test Cases the User claims were Executed
    2. Projected Solution
        1. Via Calibration Factor
            1. The composition of Critical, High, Moderate & Low Priority Test Cases which should be Executed
        2. Via Quality Override
            1. The composition of Critical, High, Moderate & Low Priority Test Cases the User claims will be Executed as a Scaled Solution

---

[xlii] Description
1. A visual representation of the combined effect of the Calibration Factor & Risk Quotient
2. Communicates the effect of the Calibration Factor & Risk Quotient upon the Targeted Distribution of Functional Processes that were Developed or Modified
3. Communicates that the User focused Testing around the mean number of Data Fields per Database Record or the most commonly used system-wide Functions

[xliii] Description
1. A visual representation of the combined effect of the Calibration Factor & Risk Quotient
2. Communicates the effect of the Calibration Factor & Risk Quotient upon the Expected (*Targeting*) Distribution of Functional Processes to be Developed or Modified
3. Communicates that the User expects to focus Testing around the mean number of Data Fields per Database Record or the most commonly used system-wide Functions

4. Quality Override

1. This section allows the User to Override the default (*scientific*) Analysis of the Historical Solution by specifying the perceived level of Risk Quotient retrospectively
2. After entering the appropriate information in the Scope section, the User must decide if the default Analysis is indicative of the actual results associated with the Historical Solution[xliv]
3. If the Risk Quotient slider is selected or the User applies the dial, then the default Analysis has been overridden by the User in favour of his / her own opinion. Consequently, the Projected Solution utilises the Overridden User-Specification to calculate the Required Test Cases for the Projected Solution
4. Critical Value[xlv]
    1. Unlike TEstimator[1], the "Apply Critical Value" button does not graphically illustrate the value of Risk Quotient precisely corresponding to 100(%) Test Coverage of Development. It is not permitted to do this because it is constrained by the <u>actual</u> number of Test Cases executed historically
    2. If the "Apply Critical Value" button cannot be clicked, or has been clicked, then the Projected Solution utilises the default Analysis of the Historical Solution based upon Statistical Mechanics & Probabilities
5. Defect-Free Confidence[xlvi]
    1. Definition: The probability of the Historical Solution being Defect-Free based upon User choices
    2. By selecting a Risk Quotient pre-set button on the slider, or using the Risk Quotient dial to specify a value anything other than the Critical Value indicated below the dial, the User claims that the Defect-Free Confidence of the Historical Solution was different to the scientific value predicted[xlvii]. For example, by selecting 0% Risk Quotient, the User claims that the number of Test Cases Executed for the Historical Solution yielded a Defect-Free Confidence of 99%. Similarly, if the User selects 50% Risk Quotient on the slider, the User claims that the Historical Solution had a Defect-Free Confidence of 80.22%

[xliv] Description
1. At page load, or when the User clicks the "Apply Critical Value" button, the application invokes the default Analysis; *i.e.* the scientific assessment of the Historical Solution based upon Statistical Mechanics & Probabilities
2. The default Analysis determines the number of Test Cases, which should have been executed if 99% of all functionality (*system-wide*) was Quality Assured; *i.e.* at a Calibration Factor of unity & Risk Quotient of 0%. The result of this calculation is subsequently utilised to determine the actual Calibration Factor (*see Indicator*) & Risk Quotient (*displayed as the Critical Value*)

[xlv] Description
1. Executed as a background calculation & represents the leanest Testing Solution for most situations (*i.e.* when $\gamma = \delta$)
2. Relates to the Functional Processes Developed or Modified for the Project exclusively

[xlvi] Note
1. The scientific assessment regarding the probability of having found all Defects is displayed as Risk Mitigation
2. When a Defect-Free Confidence value is enforced, the User overrides the scientific assessment in favour of User assessment

[xlvii] Displayed as the "Critical Value"

6. Risk Mitigation
    1. Definition: The probability that all Defects were found in the Historical Solution if the actual Quality Assurance Strategy followed the scientific basis (*default Analysis*) defined above
7. Quality Override Similarity
    1. Definition: The Proportional similarity between Defect-Free Confidence & Risk Mitigation within the Historical Solution
    2. Description: The Proportional similarity between User perception & scientific assessment within the Historical Solution

5. Risk Override

1. The Projected Solution utilises the Historical Solution as a frame of reference, scaling the Required Test Cases accordingly. After entering the appropriate information in the Scope section, the User must decide if the default Analysis (*i.e.* at page load) is desirable. This section allows the User to override the default value of Risk Quotient in the Projected Solution by using the slider & dial.[xlviii] The Projected Solution has two possible preconditions
    1. The Historical Solution utilises the default Analysis[xlix]
    2. The Historical Solution utilises a User Specified Value of Risk Quotient[l]
2. Critical Value
    1. If the “Risk Override Apply Critical Value” button is clicked, the value of Risk Quotient precisely corresponding to 100(%) Test Coverage of Development is applied. The result relates to the Functional Processes Developed or Modified for the Projected Solution exclusively. This implements the leanest Testing Solution for most situations $\{\varepsilon = \zeta\}$. Hence, when $\varepsilon > \zeta$ or $\varepsilon < \zeta$, the “Risk Override Apply Critical Value” button enforces the value of Risk Quotient such that $\varepsilon = \zeta$
    2. The Risk Override Critical Value may become negative (*depending upon User-Specifications*); indicating that the Historical Solution was under-tested from the perspective of scientific assessment
3. Example Estimate
    1. Historical Solution
        1. Setup: User Stories = 61, Executed Test Cases = 583, Conversion Ratio = 5, **DIT’s** = 10, Risk Quotient Set-Point 0% is applied[li]
        2. The default Analysis indicates that an insufficient degree of Testing was performed (*via two methods*)[lii]

[xlviii] The scientific prediction of the Projected Solution based upon Statistical Mechanics, Probabilities & Engineering principles of Similitude, such that the default Analysis determines the number of Required Test Cases which should be executed if 99% of all functionality (*system-wide*) was Quality Assured; *i.e.* at a Calibration Factor of unity & Risk Quotient of 0%

[xlix] *i.e.* The Critical Value of the Risk Quotient has been applied in Quality Override

[l] The scaled result is subsequently utilised to determine the actual Calibration Factor (*see Indicator*)

[li] *i.e.* Insisting that the Historical Solution incorporated 99% Defect-Free Confidence

[lii] Methods

1. The Test Execution Approach (*via Calibration Factor*) = Critical Priority Tests; meaning that only a proportion of Critical Priority Tests were executed
2. Graphically by $\gamma < \delta$

        3. However, the User rejects the default Analysis & insists that the Test Execution Approach (*via Quality Override*) was (*in reality*) Critical, High, Moderate & Low Priority Tests. Consequently, the User wishes to apply the Historical Solution as a baseline frame of reference to be reproduced in the Projected Solution
    2. Projected Solution
        1. Setup: User Stories = 61, Risk Quotient Critical Value = 0%
        2. Because the User has overridden the default Analysis, the Projected Solution is constrained by the Historical Solution such that the Required Test Cases must equal the Executed Test Cases because the Projected Solution intends to mimic the perceived success of the Historical Solution
4. Defect-Free Confidence[liii]
    1. Definition: The probability of the Projected Solution being Defect-Free based upon User choices
    2. By selecting a Risk Quotient pre-set button on the slider, or using the Risk Quotient dial to specify a value anything other than the Critical Value indicated below the dial, the User claims that the Defect-Free Confidence of the Historical Solution was different to the scientific value predicted[liv]. For example, by selecting 0% Risk Quotient, the User seeks a Defect-Free Confidence of 99%. Similarly, if the User selects 50% Risk Quotient on the slider, the User seeks a Defect-Free Confidence of 80.22% for the Projected Solution
5. Risk Mitigation
    1. Definition: The probability of finding at least one Defect in the Projected Solution at the Risk Override configuration specified
6. Risk Override Similarity
    1. Definition: The Proportional similarity between Defect-Free Confidence & Risk Mitigation within the Projected Solution
    2. Description: The Proportional similarity between User perception & scientific assessment within the Projected Solution

---

[liii] Note

3. The scientific assessment regarding the probability of having found all Defects is displayed as Risk Mitigation
4. When a Defect-Free Confidence value is enforced, the User overrides the scientific assessment in favour of User assessment

[liv] Displayed as the "Critical Value"

6. General Considerations

1. The importance of knowing the Defect-Free Confidence & Risk Mitigation probabilities (*together*) is for overall Risk Assessment & decision-making. The smaller the difference between these two probabilities, the lower the Net Risk & the greater the similarity between the Historical & Projected Solutions
2. Defect-Free Confidence Decision Table

Defect-Free Confidence	Test Cases Successfully Passed
99% - 97.43%	Critical (*all*), High (*all*), Moderate (*all*) & Low Priority
97.42% - 93.15%	Critical (*all*), High (*all*) & Moderate Priority
93.14% - 80.23%	Critical (*all*) & High Priority
> 0 - 80.22%	Critical Priority

3. TEstimator² User Scenarios
   1. The User applies[lv] the Critical Value Risk Quotient to the Historical Solution & the Risk Quotient of the Projected Solution is 0%. In this scenario, the User accepts scientific assessment over human recollection of historical events & the Required Number of Test Cases for the Projected Solution is calculated at a Defect-Free Confidence of 99%
   2. The User applies a Non-Critical Value Risk Quotient to the Historical Solution[lvi] & the Risk Quotient of the Projected Solution is 0%. In this scenario, The User overrides scientific assessment of the Historical Solution & declares the Defect-Free Confidence to be a User specified value[lvii]
   3. The User applies the Critical Value Risk Quotient to the Historical & Projected Solutions. In this scenario, The User accepts scientific assessment over human recollection of historical events & intends to confine the Test Coverage in the Projected Solution to the code under development only (*testing the broader system beyond the code under development will not occur*). This implements the leanest Testing Solution for most situations {$\varepsilon = \zeta$}
   4. The User applies a Non-Critical Risk Quotient value to the Historical Solution & the Critical Value Risk Quotient to the Projected Solution
   5. The User applies a Non-Critical Risk Quotient value to the Historical & Projected Solutions
4. Test Coverage of Development
   1. Definition: The proportion of Development covered by Testing
   2. Description: The scientific assessment of the proportion of Development covered by Testing. This does not include nor consider the perceived coverage by the User

[lv] This is the default setting at page load
[lvi] Test Coverage of Development should be considered prior to invoking this action
[lvii] *i.e.* The Defect-Free Confidence indicator on the left

6. How to execute a TEstimator^{3} Estimate

A. Applicability

Uses	Users
In-house Projects	1. Board of Directors
	2. Chief Executive Officers
	3. Chief Technology Officers
	4. Business Owners
	5. Security Agencies
	6. Corporations
	7. Government
	8. Military

The TEstimator^{3} Tool is a powerful & flexible instrument, facilitating the scaled reproduction of an Historical Solution to a Projected Solution. It may be used for any form of Quality Assurance, including Hardware as well as Software. The technique is based upon a fundamental Engineering technique termed Similitude (*typically applied to Thermodynamics & Fluid Mechanics*). A milestone for the development of Similitude was introduced by Edgar Buckingham via Dimensional Analysis (*now known as Buckingham Π Theory*). The basic principle of Dimensional Analysis & the derivation of the Π Grouping technique Developed by Edgar Buckingham, is such that a User (*a Scientist or Engineer*) may formulate an equation incorporating any number of unknown multidimensional variables, preceded by an experimentally determined factor (*traditionally expressed as "K"*). The value of "K" may be a constant or function, but it can only be determined experimentally & cannot be mathematically derived, only assigned an assumed value:

- For example
1. "$E = mc^2$" is a specific solution of "$E = K \bullet mc^2$" where "$K = 1$"
2. For this example in reality, "K" is not assumed to be any other value, only unity

A key feature of this technique is that there is no "wrong or right" combination of multidimensional variables to combine; it is entirely the personal choice of the individual formulating the governing equation. This becomes particularly important in the TEstimator^{3} Tool because its application is influenced by the logic being employed by the User. For this reason, a notes section has been added so that assumptions & other relevant information may be captured & recorded at the time of estimation. The examples shown in "TEstimator^{3} User Scenarios" demonstrate that a User may select a parameter from the dropdown list & relate it to a perceived level of Historical Quality Assurance via the "Quality Override - Risk Quotient". By doing this, the User combines all other factors into an undefined experimental value ("K"). The experimental value may be dependent upon (*for example*):

1. Public holidays
2. Resource utilisation
3. Sick leave
4. Skillsets & / or Skill Levels
5. Any reasonable influence or combination of influences etc.

Because the User is scaling from an Historical Solution to a Projected Solution, the value of "K" is not required to be known. The effect of this is such that all the benefits & problems naturally incorporated into the Historical Solution's experimental value "K" is also factored invisibly into the Projected Solution. This means that whatever negative influences transpired in the Historical Solution, are presumed to occur in the Projected Solution.

B. Estimation Process

C. TEstimator[3] User Scenarios

1. By example: Business Scenarios, Features, Integrated Systems, Interfaces, Number of Screens, Reports, **Requirements**, Servers
    1. A User wishes to determine a Rough-Order-of-Magnitude for the Quality Assurance Effort associated with a potential Project
    2. After some investigation, the User decides to use an Historical Solution as a frame of Reference
    3. The User decides to use the number of Requirements as their Historical Estimation Basis
    4. The User selects Requirements from the dropdown list & inputs the number of Historical Requirements & Executed Test Cases
    5. The User inputs an estimate for the number of Requirements expected in the Projected Solution; this does not need to be a value with high confidence (*it may be revised at a later time*)
    6. The User consults with relevant stakeholders & assigns a value of "Quality Override - Risk Quotient". The value assigned is intended to represent the system-wide Defect-Free Confidence of the Historical Solution. This is not just the Defect-Free Confidence of the specific code that was Developed or Modified for the Historical Solution, but also its impact upon the existing system & functionality. A possible indicator for this value may be the standard Regression Suite; if the significant majority of the Test Cases in the standard Regression Suite have passed in the normal manner, then the Defect-Free Confidence of the Historical Solution should be high. If the standard Regression Suite experienced more failures than anticipated, then the "Quality Override - Risk Quotient" value should be set low. The effect of this is to increase the Testing in the Projected Solution. If the User believes that the Historical Solution was a shining example of Quality Assurance, then the "Quality Override - Risk Quotient" should be set to 0%
    7. The User decides to set the "Risk Override - Risk Quotient" in-line with the available time & budget for the Project; Lower Risk = more Testing = greater cost
    8. A Projected Value for the Required Test Cases is computed

2. By example: **Lines-of-Code**, Modules, Objects
    1. A User wishes to determine a Rough-Order-of-Magnitude for the number of Unit Tests (Test Cases) associated with a potential Project
    2. The User decides to the use Lines-of-Code of an Historical Solution as a frame of Reference
    3. The User knows that "X" Lines-of-Code were Modified in the Historical Solution & that the Projected Solution will contain approximately "Y" Lines-of-Code
    4. The User knows that "X" Lines-of-Code required "Z" number of Unit Tests
    5. The User interprets the "Quality Override - Risk Quotient" (*in this context*) to reflect the system-wide coverage (*influence*) of the "X" Lines-of-Code upon the Historical Solution
        1. The User estimates that the "X" Lines-of-Code touched approximately 20.33% of system-wide functionality in some way (*directly or indirectly*)

    2. The User selects a "Quality Override - Risk Quotient" of 90% using the Risk Quotient Dial
  6. The User decides to set the "Risk Override - Risk Quotient" in-line with the available time & budget for the Project (*lower Risk = more Testing = greater cost*)
  7. A Projected Value for the Required Test Cases (*Unit Tests*) is computed

3. By example: Security Threats (*pings*)
   1. A User wishes to determine a Rough-Order-of-Magnitude for the potential number of Security Threats associated with a new Project
   2. The User selects an appropriate Historical Solution to utilize as a frame of reference for scaling
   3. The User describes the structure of his / her reasoning for selecting the Historical Solution in the Notes section. In this example, the User decides to relate the number of accessible ports to the number of pings
   4. The User concludes that "X" number of ports resulted in "Y" number of pings in the Historical Solution
   5. The User is informed that the Projected Solution will contain "Z" number of ports
   6. The User interprets the "Quality Override - Risk Quotient" (*in this context*) to reflect the system-wide potential for undetected Security Threats in the Historical Solution
   7. The User assumes a "Quality Override - Risk Quotient" of 21%; consequently, the User is 95.81% confident that no undetected Security Threats exist in the Historical Solution
   8. The User decides to set the "Risk Override - Risk Quotient" in-line with a ZERO Threat policy & selects 0%
   9. In the example above
      1. For the Historical Solution
         1. The Security Threats field is populated by "X" (*number of ports*)
         2. The Executed Test Cases field is populated by "Y" (*number of pings*)
      2. For the Projected Solution
         1. The Security Threats field is populated by "Z" (*number of ports*)
         2. The potential number of pings is calculated in the Required Test Cases field (*i.e.* the existence of one ping is one Test Case)

7. Definitions

A. Calibration Factor

1. Deals with system-wide tests, not just Project scope tests
2. This may include some or all of the Test Cases required to validate the Functional Processes being Developed or Modified within the scope of the Project
3. Refer to the Quality Assurance Strategy Graph to determine if the selected Calibration Factor provides the appropriate Test Coverage of Development
4. Set-Point 0.25 is selected
    1. All Critical Priority Tests are to be executed
    2. 80.22% probability of finding at least one Defect
5. Set-Point 0.5 is selected
    1. All Critical & High Priority Tests are to be executed
    2. 93.15% probability of finding at least one Defect
6. Set-Point 0.75 is selected
    1. All Critical, High & Moderate Priority Tests are to be executed
    2. 97.43% probability of finding at least one Defect
7. Set-Point 1 is selected
    1. All Critical, High, Moderate & Low Priority Tests are to be executed
    2. 99% probability of finding at least one Defect
8. Set-Point 1.25 is selected
    1. All Critical, High, Moderate & Low Priority Tests are to be executed + 25% Redundancy (*25% more tests than Set-Point "1" requires*)
    2. 99.6% probability of finding at least one Defect
9. Set-Point 1.5 is selected
    1. All Critical, High, Moderate & Low Priority Tests are to be executed + 50% Redundancy (*50% more tests than Set-Point "1" requires*)
    2. 99.84% probability of finding at least one Defect
10. Set-Point 1.75 is selected
    1. All Critical, High, Moderate & Low Priority Tests are to be executed + 75% Redundancy (*75% more tests than Set-Point "1" requires*)
    2. 99.93% probability of finding at least one Defect
11. Set-Point 2 is selected
    1. All Critical, High, Moderate & Low Priority Tests are to be executed + 100% Redundancy (*100% more tests than Set-Point "1" requires*)
    2. 99.97% probability of finding at least one Defect

B. Configuration

12. See Conversion Ratio
13. See DIT's per Test Case

C. Conversion Ratio

14. Applies to Use Cases, User Stories & Test Scenarios
15. Denotes the average number of Functional Processes associated with each of the above
16. Examples
    1. One (1) Use Case maps to five (5) Functional Processes (*on average*); therefore the Conversion Ratio is five (5)
    2. One (1) User Story maps to five (5) Functional Processes (*on average*); therefore the Conversion Ratio is five (5)

3. One (1) Test Scenario maps to one (1) Functional Processes (*on average*); therefore the Conversion Ratio is one (1)

D. Critical Value

17. The value of Calibration Factor precisely corresponding to 100Test Coverage of Development (%)
18. Relates to the Functional Processes Developed or Modified for the Project exclusively
19. Implements the leanest Testing Solution for most situations (*unless otherwise overridden deliberately by the User*)
20. Calculates the probability of finding at least one Defect associated with the above

E. Defect-Free Confidence

21. Historical Solution
    1. The probability of the Historical Solution being Defect-Free based upon User choices
22. Projected Solution
    1. The probability of the Projected Solution being Defect-Free based upon User choices

F. DIT's per Test Case

23. An acronym for Dynamic Information Tests per Test Case
24. The average number of fields per Test Cases containing dynamic information which must be verified with "yes or no" confirmation by a human being
25. Excludes all static content & fields

G. Dynamic Information Tests (DIT's)

26. The maximum number of "yes or no" Tests required to be executed by a human being
27. A "yes or no" Test may be described by the following question: is this field correct, "yes or no" ?
28. Yes = Test Passed, No = Test Failed
29. Partial or incomplete Pass categorisation does not apply
30. These Tests are confined to Dynamic content only; Static content is not applicable
31. These Tests may include any combination of Positive & / or Negative Tests required

H. Executed Test Cases

32. The actual number of Test Cases Executed

I. Framework

33. Test Approach focusing on Calibration Factor & Risk Quotient

J. Framework Probabilities

34. A visual representation of the combined effect of the Calibration Factor & Risk Quotient
35. Communicates the effect of the Calibration Factor & Risk Quotient upon the Expected (*Targeting*) Distribution of Functional Processes to be Developed or Modified

36. Communicates that the User expects to focus Testing around the mean number of Data Fields per Database Record
37. Communicates that the User expects to focus Testing around the most commonly used system-wide Functions

K. Functional Process

38. A train of Function Points enacted by the User to execute a workplace activity (*i.e. the User doing their job; Work Instructions*)
    1. Creating or Deleting Customer Accounts etc.
    2. Printing Weekly, Monthly or Annual Sales Reports etc.
    3. Billing Customers, logging Customer Complaints etc.
39. A suite of one or more appropriate Functional Processes is termed a Business Process
    1. Example 1: On-Boarding a new employee (*i.e. the Business Process*) may involve the following Functional Processes
        1. Creating an E-Mail Account
        2. Creating a User Application Account
        3. Creating a Financial Account (*salary, wage etc.*)
        4. Creating a Security Pass
        5. Creating User Access (*a Pass*) to the Bicycle Cage
    2. Example 2: Logging Customer Complaints is one Functional Process in the broader Business Process of Handling Customer Complaints

L. Progression Testing

40. Refers to tests relating to the Project Code being Modified or Developed

M. Quality Assurance Architecture

41. The overarching (*system-wide*) intended approach to Testing

N. Quality Assurance Strategy

42. The overarching (*system-wide*) resultant approach to Testing
43. It utilises the Calibration Factor & Risk Quotient parameters defined by the User in the Quality Assurance Architecture section
44. The key difference between Architecture & Strategy is that the Architecture expresses the intended (*desired*) approach to Testing via the Calibration Factor & Risk Quotient as independent parameters, whilst the Strategy expresses the combined effect of the Calibration Factor & Risk Quotient upon the approach to Testing
45. The Quality Assurance Strategy Graph communicates
    1. Risk Exposure: the proportion of system-wide functionality being Developed or Modified
    2. Risk Mitigation: the probability that the approach to Testing will find at least one Defect
    3. Test Coverage of Development: the proportion of Development covered by the Test Approach

O. Quality Override Similarity

46. The Proportional similarity between Defect-Free Confidence & Risk Mitigation within the Historical Solution

P. Regression Testing

47. Manual Regression Suite
    1. Refers to a statistically random sampling (*i.e.* a subset) of the Manual Test Cases predicted in the Progression Testing section
    2. Re-Execution of the statistical sample with a status of "Test Passed", yields 99% Confidence that the entire population of Manual Tests will pass if they were Re-Executed
    3. This statistical approach is predominantly suited to very large projects; i.e. with thousands of Manual Test Cases
    4. Statistical approaches may be applied to provide cost effective alternatives to the Quality Assurance of Software; particularly in compressed timelines
48. Automated Regression Suite
    1. Refers to a statistically random sampling (*i.e.* a subset) of the Manual Regression Test Cases predicted
    2. Automated Re-Execution of the statistical sample with a status of "Test Passed", yields 99% Confidence that the entire population of Manual Regression Tests will pass if they were Re-Executed
    3. Is appropriate if an intention exists to Re-Execute the subset multiple times (*typically: at least five times*)

Q. Required Test Cases

49. The number of Test Cases Required to be Executed

R. Risk Exposure

50. Is the proportion of system-wide functionality being Developed or Modified
51. Is governed by the number of Functional Processes under Development or Modification (*i.e.* Scope)

S. Risk Mitigation

52. General: the probability that the Quality Assurance Strategy will find at least one Defect
53. The probability that all Defects were found in the Historical Solution if the actual Quality Assurance Strategy followed the scientific basis (*default Analysis*)
54. The probability of finding at least one Defect in the Projected Solution at the Risk Override configuration specified

T. Risk Override Similarity

55. The Proportional similarity between Defect-Free Confidence & Risk Mitigation within the Projected Solution

U. Risk Quotient

56. Represents the acceptable level of risk
57. Reduces the cost of testing as the Risk Quotient increases
58. Reduces the probability of finding Defects as the Risk Quotient increases
59. Can be applied to merge the Test Approach (Calibration Factor) to the available Testing budget in Type 1 estimates. That is, the number of Test Cases which can be Designed & Executed within the time & financial constraints of the Project
    1. Decided by the User in isolation or consultation with the Quality Assurance Governance body
    2. Takes the form

   1. Quality Biased Testing: Range = 0-49%
   2. Neutrally Biased Testing: Single condition = 50%
   3. Risk Biased Testing: Range = 51-100%
3. Critical Value
   4. The value of Risk Quotient precisely corresponding to 100Test Coverage of Development (%)
   5. Relates to the Functional Processes Developed or Modified for the Project exclusively
   6. Implements the leanest Testing Solution for most situations (*unless otherwise overridden deliberately by the User*)
   7. Calculates the probability of finding at least one Defect associated with the above

V. Scope

60. The number of Functional Processes to be Developed or Modified within the context of a project

W. System Unaffected

61. Is the proportion of system-wide functionality not being Developed or Modified
62. Is the residual of the Risk Exposure (*governed by the Scope*)

X. Use Case

63. A suite of one or more Functional Processes
64. The term applied in WaterFall Methodology (*historically*), referring to the manner in which a User shall execute a Business Objective. That is, how the User shall interact with the Functions being Developed or Modified
65. Typically, a well-written Use Case contains approximately five (5) Test Scenarios; however, this depends upon many factors including the writing style & experience of the Use Case composer.
66. A poorly constructed Use Case will contain greater than ten (10) Test Scenarios. If this occurs, the User should consider decomposing the Use Case into multiple smaller ones
67. The WaterFall analogue of User Story within Agile Methodology

Y. User Story

68. A suite of one or more Functional Processes
69. The term applied in Agile Methodology, referring to the manner in which a User shall execute a Business Objective. That is, how the User shall interact with the Functions being Developed or Modified
70. Typically, a well-written User Story contains approximately five (5) Test Scenarios; however, this depends upon many factors including the writing style & experience of the User Story composer
71. A poorly constructed User Story will contain greater than ten (10) Test Scenarios. If this occurs, the User should consider decomposing the User Story into multiple smaller ones
72. The Agile analogue of Use Case within WaterFall Methodology

Z. Test Case

73. A Test Case is collection of DIT's to be executed simultaneously
74. An average Test Case contains an average number of DIT's defined by the User in the Quality Assurance Section
75. "Test Cases" refers to the maximum number of Test Cases that the Scope is expected to yield
76. "Test Case" refers to manual Tests only; no automated Testing is applicable

AA. Test Case & Defect Complexity

77. Test Case Complexity
    1. Example: The average Complexity as configured by the User in the Quality Assurance Architecture section
    2. 10 DIT's per Test Case configured by User (*DIT's = Dynamic Information Tests*)
    3. 50 DIT's per Test Case is the maximum permissible value of the dial
    4. Test Case Complexity = 10/50 = 20%
78. Defect Complexity
    1. Refers to the complexity of its discovery, not the complexity of its correction (*fixing*) by the Development Team
    2. It follows that the average complexity of a population of Test Cases is directly proportional to the average complexity of the population of Defects they raise

BB. Test Scenario

79. A synonym for Functional Process (*typically*)
80. One (1) Test Scenario is equivalent to one (1) Functional Process (*typically*)
81. A Use Case or User Story contains approximately five (5) Test Scenarios (*typically*)
82. The constitution & specific form of a Test Scenario is subjective & affected by the experience of the composer

8. Applicator Functional Processes

A. Functional Process List

A Functional Process is a train of Function Points leading to a required action or output via a pathway; typically this takes the form of User Work Instructions. The Functional Processes for this Application Suite are shown below[lviii]. Each Functional Process begins with a *jump-on* point[lix] & completes with a *jump-off* point[lx]. The jumping-on points are QuEstimate View, LEstimate View, Basic View, Advanced View, TEstimator[2], TEstimator[3], REstimator (*Stand Alone*) & @Risk (*Stand Alone*); the jumping-off point is always Save to File. Hence, the total number of Functional Processes within the Applicator Suite (*this product suite*) is shown to be twenty five (25), as follows:

1. TEstimator[1]
    1. QuEstimate View → *pathways* (*4 Pathways*)
        1. Save to File
        2. AEstimator → Save to File
        3. AEstimator → REstimator → Save to File
        4. AEstimator → REstimator → PEstimator → Save to File
    2. LEstimate View → *pathways* (*4 Pathways*)
        1. Save to File
        2. AEstimator → Save to File
        3. AEstimator → REstimator → Save to File
        4. AEstimator → REstimator → PEstimator → Save to File
    3. Basic View → *pathways* (*4 Pathways*)
        1. Save to File
        2. AEstimator → Save to File
        3. AEstimator → REstimator → Save to File
        4. AEstimator → REstimator → PEstimator → Save to File
    4. Advanced View → *pathways* (*4 Pathways*)
        1. Save to File
        2. AEstimator → Save to File
        3. AEstimator → REstimator → Save to File
        4. AEstimator → REstimator → PEstimator → Save to File
2. TEstimator[2] → *pathways* (*4 Pathways*)
    1. Save to File
    2. AEstimator → Save to File
    3. AEstimator → REstimator → Save to File
    4. AEstimator → REstimator → PEstimator → Save to File
3. TEstimator[3] → *pathways* (*3 Pathways*)
    1. Save to File
    2. REstimator → Save to File
    3. REstimator → PEstimator → Save to File
4. REstimator (*Stand Alone*) → Save to File
5. @Risk → Save to File

(*2 Pathways*)

[lviii] Total number of Functional Processes = 4 + 4 + 4 + 4 + 4 + 3 + 2 = 25

[lix] A starting Function Point

[lx] A finishing Function Point

B. Use Case (User Story) List

1. A User executes a TEstimator^{1} estimate
2. A User executes a TEstimator^{2} estimate
3. A User executes a TEstimator^{3} estimate
4. A User executes a Stand Alone REstimator estimate
5. A User executes a Stand Alone @Risk Visualisation analysis

- **Total number of Use Cases (User Stories) = 5**

C. How to Calculate (*or estimate*) the Conversion Ratio

The Conversion Ratio denotes the average number of Functional Processes associated with Use Cases, User Stories & Test Scenarios. Although Use Cases, User Stories & Test Scenarios have industry definitions, no Universally accepted rigorously precise definition exists; Use Cases, User Stories & Test Scenarios are highly subjective & significant variation exists within categories depending upon the composer. To overcome this practical impasse, the Applicator Product Suite standardises on the use of Functional Processes in place of Use Cases, User Stories & Test Scenarios. It achieves this via the Conversion Ratio such that the User assigns a definition of Use Cases, User Stories & Test Scenarios, in terms of the number of Functional Process each category possesses[lxi]. The default values within the Applicator Suite are:

1. One (1) Use Case maps to five (5) Functional Processes (*on average*); therefore the Conversion Ratio is five (5)
2. One (1) User Story maps to five (5) Functional Processes (*on average*); therefore the Conversion Ratio is five (5)
3. One (1) Test Scenario maps to one (1) Functional Processes (*on average*); therefore the Conversion Ratio is one (1)
4. The mathematical representation of the applicable forms are shown below

In our case:

$$Conversion\ Ratio = \frac{Functional\ Processes}{Use\ Cases\ or\ User\ Stories} = \frac{25}{5} = 5$$

Or

$$Conversion\ Ratio = \frac{Functional\ Processes}{Test\ Scenarios} = \frac{25}{25} = 1$$

[lxi] On average, over a population

D. User WorkFlows (*Functional Process Map*)