A Consumer/Producer Approach to Risk-Driven Software
Reliability and Testing
Norman Schneidewind, Mike Hinchey

1. INTRODUCTION
We introduce the concept of using both consumer software and producer software in the analysis of risk as it relates to test criteria and strategies. There are two different versions of the software, one representing the consumer’s viewpoint, which requires reliable software at reasonable cost, and the other representing the producer’s viewpoint, trying to produce adequate software with the lowest effort.

As a using organization, the consumer does not conduct software design but does write requirements specifications that the producer uses to design the software. In addition, both the consumer and producer are responsible for quality control activities, such as inspection and testing.

2. RISK
We use the following definition of risk to account for, not only the probability of an undesirable event but, in addition, the consequences, as represented by failure count. Putting these two factors together, we define risk as the expected number of failures (i.e., probability * failure count).

2.1 Data Analysis
We generated sample failure data. We used a failure count distribution that is Poisson distributed. We generated 28 samples of failure count with means of 1.0 and .5 for the consumer and producer, respectively, based on the experience that the consumer will find more faults than the producer. We need to account for the severity of faults and failures when assessing the effectiveness of test criteria. Thus, we randomly assigned severities to faults and failures, using the set of values 1 through 5, where “1” is the most severe.

Then, the corresponding weights \( w_i \) and severity definitions are as follows:

<table>
<thead>
<tr>
<th>( w_i )</th>
<th>Severity Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (1 - 1 / 15) = .9333 )</td>
<td>Loss of life or system</td>
</tr>
<tr>
<td>( (1 - 2 / 15) = .8667 )</td>
<td>Affects ability to complete mission objectives</td>
</tr>
<tr>
<td>( (1 - 3 / 15) = .8000 )</td>
<td>Workaround available, therefore minimal effects on procedures—mission objectives met</td>
</tr>
<tr>
<td>( (1 - 4 / 15) = .7333 )</td>
<td>Insignificant violation of requirements or recommended practices, not visible to user in operational use</td>
</tr>
<tr>
<td>( (1 - 5 / 15) = .6667 )</td>
<td>Cosmetic issue which should be addressed or tracked for future action, but not necessarily a present problem</td>
</tr>
</tbody>
</table>

2.2 Test Criteria

1. Actual Probability of Accepting Software
The consumer accepts software with probability \( P_{ac} \), when \( P_{ac} < LQL \) Limit, where \( LQL \) Limit = (mean \( P_{ac} \) + 1 standard deviation).

Similarly, the producer accepts low quality software with probability \( P_{ap} \), when \( P_{ap} < AQL \) Limit, where \( AQL \) Limit = (mean \( P_{ap} \) + 1 standard deviation). Software is rejected and faults are removed for \( P_{ac} \geq LQL \) Limit and \( P_{ap} \geq AQL \) Limit; the number of tests \( n \) that are required for fault removal are noted.

2. Actual Risk of Accepting Software
The consumer incurs a risk of accepting software \( c(t, r_c) \), when \( c(t, r_c) < LQL \) Limit, where \( LQL \) Limit = (mean \( c(t, r_c) \) + 1 standard deviation). Similarly, the producer incurs a risk of accepting software \( p(t, r_p) \), when \( p(t, r_p) < LQL \) Limit, where \( LQL \) Limit = (mean \( p(t, r_p) \) + 1 standard deviation). Software is rejected and faults are removed for \( c(t, r_c) \geq LQL \) Limit and \( p(t, r_p) \geq LQL \), and the number of tests \( n \) that are required for fault removal are noted.

3. Maximize Probability of Acceptance
From the definitions, the probability that the consumer and producer of software will accept high quality software during test is \( (1 - P_{ac}) \) and \( (1 - P_{ap}) \), respectively. Therefore, we would like the product of these probabilities to be maximized. In addition, since \( A \) in equation (0.1) is to be maximized, the complement of the severity weight is applied.

\[
A = \max [(1 - w_i)(1 - P_{ap}) \times (1 - P_{ac})] \quad (0.1)
\]

4. Minimize Actual Risk
Another test criterion is to minimize, \( B \), the actual sum of consumer risk and producer risk. By applying the severity weight, since \( B \) is to be minimized, \( B \) is computed in equation (0.2):

\[
B = \min \left( w_i \right) \left[ c(t, r_c) + p(t, r_p) \right] \quad (0.2)
\]
where \( t \) is test time interval. To implement this criterion, we accept the software when \( B \) is minimum, and remove faults in all test time intervals where the criterion is not satisfied.

3. RESULTS
The results in Table 1 indicate that the risk criterion is superior.
Table 1. Example Fault Removal Scenario: 53 Total Faults (36 Consumer, 17 Producer)

<table>
<thead>
<tr>
<th>Test Criterion</th>
<th>Consumer</th>
<th>Fault Removal Effectiveness</th>
<th>Producer</th>
<th>Fault Removal Effectiveness</th>
<th>Number of Tests</th>
<th>Fault Removal Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize Probability of Acceptance Criterion</td>
<td>9</td>
<td>9 / (36*6) = .0417</td>
<td>2</td>
<td>2 / (17*5) = .0235</td>
<td>1, 3, 4, 5</td>
<td>11 / (53*11) = .0189</td>
</tr>
<tr>
<td>1. Remove faults for P_{ac} (t, r_c) &gt; LQL limit and P_{ap} (t, r_p) &gt; AQL limit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimize Risk Criterion</td>
<td>9</td>
<td>9 / (36*6) = .0417</td>
<td>10</td>
<td>10 / (17*12) = .0490</td>
<td>1, 2, 3, 4, 5</td>
<td>18 / (53*18) = .0199</td>
</tr>
<tr>
<td>2. Remove faults for μ(t, r_c) &gt; LQL Limit and μ(t, r_p) &gt; AQL Limit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>