16.35

Aerospace Software Engineering

Reliability, Availability, and Maintainability
Software Fault Tolerance

Prof. Kristina Lundqvist
Dept. of Aero/Astro, MIT
Definitions

Software reliability
- The probability that a system will operate without failure under given conditions for a given time interval
- Expressed on a scale 0 to 1

Software availability
- Probability that a system is functioning completely at a given instant in time, assuming that the required external resources are also available.
- A system completely up and running has availability 1; on that is unusable has availability 0.
Software maintainability

- Probability that, for a given condition of use, a maintenance activity can be carried out within a stated time interval and using stated procedures and resources.

- Ranges from 0 to 1.
Catastrophic: A failure that may cause death or system loss

Critical: a failure that may cause severe injury or major system damage that results in mission loss

Marginal: a failure that may cause minor injury or minor system damage that results in delay, loss of availability, or mission degradation

Minor: a failure not serious enough to cause injury or system damage, but that results in unscheduled maintenance or repair.
## Failure Data

Interfailure Times (Read left to right, in rows)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>30</td>
<td>113</td>
<td>81</td>
<td>115</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>138</td>
<td>50</td>
<td>77</td>
<td>24</td>
<td>108</td>
<td>88</td>
<td>670</td>
</tr>
<tr>
<td>325</td>
<td>55</td>
<td>242</td>
<td>68</td>
<td>422</td>
<td>180</td>
<td>10</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>227</td>
<td>65</td>
<td>176</td>
</tr>
<tr>
<td>10</td>
<td>1071</td>
<td>6150</td>
<td>3321</td>
<td>1045</td>
<td>648</td>
<td>5485</td>
</tr>
</tbody>
</table>

**Type-1 uncertainty**

**Type-2 uncertainty**
Measuring Reliability, Availability, and Maintainability

- **MTTF**
  - Average of the interfailure times
- **MTTR**
- **MTBF**
  - \( MTBF = MTTF + MTTR \)
- **R**
  - \( R = \frac{MTTF}{1 + MTTF} \)
- **A**
  - \( A = \frac{MTBF}{1 + MTBF} \)
- **M**
  - \( M = \frac{1}{1 + MTTR} \)
Reliability Stability and Growth

- **Reliability stability**
  - If the interfailure times stay the same

- **Reliability growth**
  - If they increase

- **Probability density function**
  - $f(t)$

\[ \int_{t_2}^{t_1} f(t) \, dt \]

Reliability function $R(t) = 1 - F(t)$
Predicting next Failure Times from Past History
The Jelinsky-Moranda Model (1972)

- Assumes: no type-2 uncertainty
- Assumes: fixing any fault contributes equally to improving the reliability

<table>
<thead>
<tr>
<th>i</th>
<th>Mean Time to ith Failure</th>
<th>Simulated Time to ith Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>41</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>33</td>
<td>77</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
<td>64</td>
</tr>
<tr>
<td>9</td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td>10</td>
<td>56</td>
<td>34</td>
</tr>
<tr>
<td>11</td>
<td>67</td>
<td>183</td>
</tr>
<tr>
<td>12</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>13</td>
<td>111</td>
<td>17</td>
</tr>
<tr>
<td>14</td>
<td>167</td>
<td>190</td>
</tr>
<tr>
<td>15</td>
<td>333</td>
<td>436</td>
</tr>
</tbody>
</table>
Software Fault Tolerance
Fault vs. Failure

- How do faults occur?
- What is a failure?

- Faults represent problems that developers see
- Failures are problems that users or customers see
Handling Design Faults

- Prevention
- Removal
- Fault Tolerance
- Input Sequence Work Arouinds
Reliability is \textit{“the probability of failure free operation of a computer program in a specified environment for a specified period of time”}, where failure free operation in the context of software is interpreted as adherence to its requirements 

\textit{[Pressman 97].}

\[ \text{MTBF} = \text{MTTF} + \text{MTTR} \]

Safety
“The goal of [software] fault tolerance methods is to include safety features in the software design or source code to ensure that the software will respond correctly to input data errors and prevent output and control errors. The need for error prevention or fault tolerance methods is determined by the system requirements and the system safety assessment process.”
The function of software fault tolerance is to prevent system accidents (or undesirable events, in general), and mask out faults if possible.
Single Version Techniques

- Program structure and actions
- Error detection
- Exception handling
- Checkpoint and restart
- Process pairs
- Data diversity
Program Structure and Actions

- Modular Decomposition – Partitioning
  - Horizontal Partitioning
  - Vertical Partitioning
- Visibility & Connectivity
- System Closure
- Temporal Structuring
Error Detection

- Structured
  - Replication
  - Timing
  - Reversal
  - Coding
  - Reasonableness
  - Structural checks

- Ad-Hoc
  - Fault Trees
Exception Handling

- Interface Exceptions
- Local Exceptions
- Failure Exceptions
Checkpoint and Restart

- **Checkpoint**

- **Restart**
  - Static
  - Dynamic

![Diagram of Checkpoint and Restart Process]

- Input
- Execution
- Error Detection
- Output
- Retry
- Checkpoint Memory
- Checkpoint
Two identical versions of software

Separate processors
Data Diversity I

- Input Data Re-Expression
- Input Re-Expression with Post-Execution Adjustment
- Re-Expression via Decomposition and Recombination

Diagram:
- Input
- Re-expression -1
- Re-expression -n
- Selection Switch
- Checkpoint Memory
- Checkpoint
- Execution
- Retry
- Error Detection
- Output
Data Diversity II
Multi-Version Techniques

- Recovery Blocks
- N-Version Programming
- N Self-Checking Programming
- Consensus Recovery Blocks
- t/(n-1) Variant Programming
Recovery Blocks

Checkpoint Memory

Checkpoint

Primary Version

Alternate V-1

Alternate V- n

Selection Switch

Acceptance Test

Input

Output
N-Version Programming

Input

Version 1

Version 2

Version n

Selection Switch

Output
N Self-Checking Program

Input

Version 1

Acceptance Test

Version N

Selection Switch

Acceptance Test

Output

Version 1

Input

Version 2

Comparison

Selection Switch

Version N-1

Comparison

Output

Version N
t/(n-1) Variant Programming:
diagnosability measure to isolate the faulty units to a subset of size at most (n-1) assuming there are at most t faulty units
Web-based subject evaluations available, please fill them out

Wednesday: practice questions for the final exam