Software Engineering

Testing

CS 1530 Software Engineering Fall 2004

Reading

- Chapters 8 & 9

CS 1530 Software Engineering Fall 2004

Test team

- Professional testers: organize and run the tests
- Analysts: who created requirements
- System designers: understand the proposed solution
- Configuration management specialists: to help control fixes
- Users: to evaluate issues that arise
Performance tests
- Stress tests
- Volume tests
- Configuration tests
- Compatibility tests
- Regression tests
- Security tests
- Timing tests
- Environmental tests
- Quality tests
- Recovery tests
- Maintenance tests
- Documentation tests
- Human factors (usability) tests

Quality Tests
- Determine
  - reliability
    - probability that system will operate without failure (under given conditions) for a given time interval
  - availability
    - probability that system is operating at a given point in time
  - maintainability
    - probability that a maintenance activity can be carried out within a specified time interval

Table 9.3: Inter-failure times (read left to right, in rows).

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>24</th>
<th>40</th>
<th>168</th>
<th>180</th>
<th>206</th>
<th>246</th>
<th>264</th>
<th>324</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>40</td>
<td>168</td>
<td>180</td>
<td>206</td>
<td>246</td>
<td>264</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>40</td>
<td>168</td>
<td>180</td>
<td>206</td>
<td>246</td>
<td>264</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>40</td>
<td>168</td>
<td>180</td>
<td>206</td>
<td>246</td>
<td>264</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>40</td>
<td>168</td>
<td>180</td>
<td>206</td>
<td>246</td>
<td>264</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>40</td>
<td>168</td>
<td>180</td>
<td>206</td>
<td>246</td>
<td>264</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>40</td>
<td>168</td>
<td>180</td>
<td>206</td>
<td>246</td>
<td>264</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>24</td>
<td>40</td>
<td>168</td>
<td>180</td>
<td>206</td>
<td>246</td>
<td>264</td>
<td>324</td>
</tr>
</tbody>
</table>
Measuring & Modeling Reliability, Availability & Maintainability

- Observe failures of running system and record when they occur
  - interfailure times
- Uncertainties
  - type-1: don’t know when next failure will occur (and triggered by which fault)
  - type-2: don’t know if a fault fix will remove the failure (and improve reliability)

Mean Time Between Failure

- Assume we observe system with total of n+1 failures
- MTBF = (∑ t_i) / n, i = 1..n, where t_i is the time between failure i and i+1
- MTTR (Mean Time to Repair) = mean time it takes to repair
Mean Time Between Failures

- MTBF = mean time between failures (when the system is operational)
- MTBF = MTTF + MTTR

Quality Formulas

- Reliability
  - \( R = \frac{MTTF}{1+MTTF} \)
- Availability
  - \( A = \frac{MTBF}{1 + MTBF} \)
- Maintainability
  - \( M = \frac{1}{1 + MTTR} \)

Predicting Reliability

- Reliability stability
  - interfailure times stay the same
- Reliability growth
  - interfailure times increasing
- Predicting system failure is hard
  - can use a probability density function
Uniform Density

Distribution Function

- \( F(t) = \int_{t_1}^{t_2} f(t) \, dt \)
- \( F(t) \) gives the probability that the system fails between \( t_1 \) and \( t_2 \)
- \( R(t) = 1 - F(t) \) reliability function

Acceptance tests

- Pilot test: install on experimental basis
- Alpha test: in-house test
- Beta test: customer pilot
- Parallel testing: new system operates in parallel with old system
Test documentation

■ Test plan: describes system and plan for exercising all functions and characteristics
■ Test specification and evaluation: details each test and defines criteria for evaluating each feature
■ Test description: test data and procedures for each test
■ Test analysis report: results of each test

INPUT DATA:
Input data are to be provided by the LIST program. The program generates randomly a list of N words of alphanumeric characters; each word is of length M. The program is invoked by calling RUN LIST(N,M).

Case 1: Use LIST with N=5, M=5
Case 2: Use LIST with N=10, M=5
Case 3: Use LIST with N=15, M=5
Case 4: Use LIST with N=50, M=10
Case 5: Use LIST with N=100, M=10
Case 6: Use LIST with N=150, M=10

INPUT COMMANDS:
The SORT module is invoked by using the command RUN SORT(INBUF,OUTBUF) or RUN SORT(INBUF).

OUTPUT DATA:
If two parameters are used, the sorted list is placed in OUTBUF. Otherwise, it is placed in INBUF.

SYSTEM MESSAGES:
During the sorting process, the following message is displayed:
"Sorting ... please wait ..." Upon completion, SORT displays the following message on the screen:
"Sorting completed"
To halt or terminate the test before the completion message is displayed, press CONTROL-C on the keyboard.

Step N:

Step N+1:
Screen will ask for the name of the data file.
Type specified.
Menu will appear, reading
* modify file
* modify file
* modify file
Menu will be highlighted. Move cursor to "modify file" and press RETURN key.

Step N+2:
Screen will ask for record number. Type "4017",
Record number: 4017
X: 0042
Y: 0036
Soil type: clay
Percolation: 4 mtrs/hr
Vegetation: kudzu
Canopy height: 25 mtrs
Water table: 12 mtrs
Construct: outhouse
Maintenance code: 3T/4F/9R

Step N+3:
Press function key 9: modify
Entries on screen will be highlighted. Move cursor to VEGETATION field. "kudzu" will be highlighted and press RETURN key.
Entries in selected field will be highlighted. Move cursor to VEGETATION field --- or blank.
Press function key 9: Return to previous screen.
Menu will appear, reading
* modify file
* modify file
* modify file
To verify that the modification has been recorded place cursor next to "modify file" and press RETURN

Step N+4:
Screen will ask for record number. Type "4017",
Record number: 4017
X: 0042
Y: 0036
Soil type: clay
Percolation: 4 mtrs/hr
Vegetation: grass
Canopy height: 25 mtrs
Water table: 12 mtrs
Construct: outhouse
Maintenance code: 3T/4F/9R
Problem report forms

- Location
- Timing
- Symptom
- End result
- Mechanism
- Cause

Testing safety-critical systems

- Design diversity: use different kinds of designs, designers
- Software safety cases: make explicit the ways the software addresses possible problems
  - failure modes and effects analysis
  - hazard and operability studies
- Cleanroom: certifying software with respect to the specification

<table>
<thead>
<tr>
<th>Known cause</th>
<th>Unknown cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description of system behavior</td>
<td>Deductive analysis, including fault tree analysis</td>
</tr>
<tr>
<td>Inductive analysis, including failure modes and effects analysis</td>
<td>Exploratory analysis, including hazard and operability studies</td>
</tr>
</tbody>
</table>