Lecture 12: Quality Estimation / Test Doc., Org. and Process Improvement (Test Maturity Model)

Spring 2022
Exam Dates

- Exam 1: Thursday, 19-May-2022 at 10:15-11:55, in room r1021 via Moodle -
capacity limit: 66
- Exam 2: Thursday, 26-May-2022, 10:15-11:55, in room r1021 via Moodle -
capacity limit: 66

You must receive

... at least 33% of the max. possible points from the homework assignments to
qualify for the exam and

... at least 10 marks in the exam to not fail the course.

In total, you need at least 50 marks to not fail the course.

- Retake Exam (resit): Monday, 13-June-2022 at 16:15-17:55, in room r1022
  via Moodle
  - Please note that you must register for the retake exam at the latest 3 days before
    the exam date
Exam Dates

• Exam 1: Thursday, 19-May-2022 at 10:15-11:55, in room r1021 via Moodle - capacity limit: 100
• Exam 2: Thursday, 26-May-2022 at 10:15-11:55, in room r1021 via Moodle - capacity limit: 100

You must receive ... at least 33% of the max. possible points from the homework assignments to qualify for the exam and ... at least 10 marks in the exam to not fail the course.

In total, you need at least 50 marks to not fail the course.

• Retake Exam (resit): Monday, 13-June-2022 at 16:15-17:55, in room r1022 via Moodle
  – Please note that you must register for the retake exam at the latest 3 days before the exam date

The “or” is exclusive → Exam 1 xor Exam 2
Exam Dates

- Exam 1: Thursday, 19-May-2022 at 10:15-11:55, in room r1021 via Moodle - capacity limit: 100
- Exam 2: Thursday, 26-May-2022, 10:15-11:55, in room r1021 via Moodle - capacity limit: 100

You must receive at least 33% of the max. possible points from the homework assignments to qualify for the exam and
... at least 10 marks in the exam to not fail the course.
In total, you need at least 50 marks to not fail the course.

- Retake Exam (resit): Monday, 13-June-2022 at 16:15-17:55, in room r1022 via Moodle
  - Please note that you must register for the retake exam at the latest 3 days before the exam date

Study Regulation: “If the student is not present at the exam, mark ‘F’ (fail) should be inserted into SIS. If the students was sick, he/she should present medical certificate to Ülle Holm who will cancel the result.”
Exam Dates

- Exam 1: Thursday, 19-May-2022 at 10:15 - 11:55, in room r1021 via Moodle - capacity limit: 100
- Exam 2: Thursday, 26-May-2022, 10:15 - 11:55, in room r1021 via Moodle - capacity limit: 100

You must receive … at least 33% of the max. possible points from the homework assignments to qualify for the exam and … at least 10 marks in the exam to not fail the course.
In total, you need at least 50 marks to not fail the course.

- Retake Exam (resit): Monday, 13-June-2022 at 16:15 - 17:55, in room r1022 via Moodle
  - Please note that you must register for the retake exam at the latest 3 days before the exam date.

The exams will be done via Moodle in the classroom!
Exam Dates

- Exam 1: Thursday, 19-May-2022 at 10:15-11:55, in room r1021 via Moodle - capacity limit: 100
- Exam 2: Thursday, 26-May-2022 at 10:15-11:55, in room r1021 via Moodle - capacity limit: 100

... at least 33% of the max. possible points from the homework assignments to qualify for the exam and
... at least 10 marks in the exam to not fail the course.
In total, you need at least 50 marks to not fail the course.

- Retake Exam (resit): Monday, 13-June-2022 at 16:15-17:55, in room r1022 via Moodle
  - Please note that you must register for the retake exam at the latest 3 days before the exam date.

IDs will be checked. If you do the exam online but are not in the classroom ➔ FAIL.
Exams in Moodle (last year’s screenshot)

Exam 1

This is the first exam option. Exam 1 takes place on Thursday, May 21 and starts at 10:15. You must be registered in SIS for exam 1 (any of the three rooms is ok) if you want to take this exam.

The exam lasts 100 min and consists of two parts.

After 100 min the exam closes automatically and whatever you have filled into the answers will be saved and used for marking.

The maximum number of points is 30.

Part 1 [22 points] will be marked automatically (multiple-choice).

Part 2 [8 points] will be marked manually.

NOTE: You do not need to submit. At the end of the exam time, Moodle will automatically close the exam and submit the data you have entered up to that point. However, if you are sure that you have finished before the end time, you may submit yourself. Just keep in mind that you cannot re-open the exam. You only have one attempt.

Exam1-Part1+Part2

Part 1 (questions 1-22) of the exam resembles what you did in the quizzes. Each question has one correct answer. These questions will be graded automatically.

Part 2 (questions 23+24) of the exam are open text question. Please answer each sub-question by writing in the open text field. Clearly state to which sub-question (a, b, c ...) your answer relates. These questions will be graded manually.
Exams in Moodle (last year’s screenshot)

Exam 1
This is the first exam option. Exam 1 takes place on Thursday, May 21 and starts at 12 pm. You must be registered for CR456 for exam 1 (any of the three rooms is ok) if you want to take this exam.

The exam lasts 100 min and consists of two parts.

After 100 min the exam closes automatically and whatever you have achieved during Part 1 is recorded.

The maximum number of points is 30.

Part 1 [22 points] will be marked automatically.

Part 2 [8 points] will be marked manually.

NOTE: You do not need to submit. At the end of the exam you need to stop working and click the stop button. The exam cannot be opened.

Exam1-Part1+Part2

Part 1 (questions 1–22) of the exam resembles what you did in the quizzes. Each question has only one possible answer. These questions will be graded automatically.

Part 2 (questions 23+24) of the exam are open text question. Please answer the sub-question by writing in the open text field. Clearly state to which sub-question (a, b, c ....) your answer relates. These questions will be graded manually.

More details in Lecture 13
Status after HW7: Almost all active students have qualified for the exam.
Lectures

• Lecture 1 (10.02) – Introduction to Software Testing
• Lecture 2 (17.02) – Basic Black-Box Testing Techniques: Boundary Value Analysis & Equivalence Class Partitioning
• Lecture 3 (03.03) – BBT advanced: Combinatorial Testing
• Lecture 4 (10.03) – Basic White-Box Testing Techniques: Control-Flow Coverage
• Lecture 5 (17.03) – BBT adv.: State-Transition, Metamorphic, Random Testing
• Lecture 6 (24.03) – Test Levels, Test Tools, Test Automation
• Lecture 7 (31.03) – BBT adv.: Exploratory Testing, Behaviour Testing
• Lecture 8 (07.04) – BBT adv.: GUI / Visual Testing, Usability Testing, A/B Testing
• Lecture 9 (14.04) – Security Testing of Mobile Applications
• Lecture 10 (21.04) – WBT adv.: Data-Flow Testing / Mutation Testing
• Lecture 11 (28.04) – WBT adv.: Symbolic Execution, Static Code Analysis, Review
• Lecture 12 (05.05) – Defect Estimation / Test Documentation, Organisation and Process Improvement (Test Maturity Model)
• Lecture 13 (12.05) – Exam Preparation
• Lecture 14 (19.05) – Advanced Topics (optional)
White-Box Testing Techniques

• Control-Flow Testing
• Data-Flow Testing
• Mutation Testing
• Symbolic Execution
• Static Code Analysis
• Reviews

Recap from last lecture!

Lecture 11
Reading Techniques

• Ad hoc
• Checklist-based
• Defect-based
• Usage-based
• Perspective-based
# Ad-hoc / Checklist-based / Defect-based Reading

<table>
<thead>
<tr>
<th>Omission</th>
<th>Omission</th>
<th>Data Type Inconsistencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality&lt;br&gt;Performance&lt;br&gt;Interface&lt;br&gt;Environment</td>
<td>Missing Functionality&lt;br&gt;Missing Performance&lt;br&gt;Missing Environment&lt;br&gt;Missing Interface</td>
<td>1 Identify all data objects mentioned...&lt;br&gt;1a. Are all data objects mentioned...&lt;br&gt;...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commission</th>
<th>Commission</th>
<th>Incorrect Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambiguity&lt;br&gt;Inconsistency&lt;br&gt;Incorrect&lt;br&gt;Wrong</td>
<td>Ambiguous Information&lt;br&gt;Inconsistent Information&lt;br&gt;Incorrect or Extra Func.&lt;br&gt;Wrong Section</td>
<td>1 For each functional requirement identify...&lt;br&gt;1a. Are all values written to each input...&lt;br&gt;...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ad Hoc</th>
<th>Checklist</th>
<th>Ambiguities or Missing Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1 Identify the required precision, response...&lt;br&gt;1a. Are all required precisions indicated?&lt;br&gt;...</td>
</tr>
</tbody>
</table>

Defect-based Reading
Usage-based Reading

1 – Prioritize Use Cases (UCs)
2 – Select UC with highest priority
3 – Track UC’s scenario through the document under review
4 – Check whether UC’s goals are fulfilled, needed functionality provided, interfaces are correct, and so on (report issues detected)
5 – Select next UC

Source:
Perspective-based Reading

- Scenarios
  - Decrease overlap (redundancy)
- Purpose
  - Improve effectiveness
Structure of Lecture 12

• Quality Estimation
• Lab 11
• Test Planning & Documentation
• Test Organization
• Test Process Improvement (Test Maturity Model)
Quality – ISO 25010 Standard

ISO 25010 Standard
Quality – How to measure / predict it?

ISO 25010 Standard
Quality Prediction

• Based on product, process, and people properties

• Examples:
  • Quality [#Faults] = f(detected #Faults)
  • Quality [#Failures per Time/Effort unit] = f(Test Effort | Time)
    or = f(observed #Failures)
  • Quality [#Faults | Faulty: yes/no] = f(Code Size | Complexity | Churn)
  • Quality [Issue Resolution Time] = f(Product & Process & People Features)
  • Quality [#Feature requests | Sentiment about Features]
    = f(User Feedback)
  • Quality [Energy | Performance impact] = f(Code Refactoring)
Quality Prediction

• Based on product, process, and people properties
• Examples:
  • Quality [#Faults] = f(detected #Faults)
  • Quality [#Failures per Time/Effort unit] = f(Test Effort | Time)
    or = f(detected #Failures)
  • Quality [#Faults | Faulty: yes/no] = f(Code Size | Complexity | Churn)
  • Quality [Issue Resolution Time] = f(Product & Process & People Features)
  • Quality [#Feature requests | Sentiment about Features]
    = f(User Feedback)
  • Quality [Energy | Performance impact] = f(Code Refactoring)

Which of the listed example models could relate to reviews?
Quality Prediction

• Based on product, process, and people properties

• Examples:

  • Quality [#Faults] = f(detected #Faults)
  • Quality [#Failures per Time/Effort unit] = f(Test Effort | Time)
    or = f(detected #Failures)
  • Quality [#Faults | Faulty: yes/no] = f(Code Size | Complexity | Churn)
  • Quality [Issue Resolution Time] = f(Product & Process & People Features)
  • Quality [#Feature requests | Sentiment about Features]
    = f(User Feedback)
  • Quality [Energy | Performance impact] = f(Code Refactoring)
Capture-Recapture – Defect Estimation

Remember:
Defect and Fault (and Bug) are synonyms
Capture-Recapture – Defect Estimation
Capture-Recapture – Defect Estimation
Capture-Recapture – Defect Estimation
Capture-Recapture – Defect Estimation

- Situation: Two inspectors are assigned to inspect the same product
  - $d_1$: #defects detected by Inspector 1
  - $d_2$: #defects detected by Inspector 2
  - $d_{12}$: #defects detected by both inspectors
  - $N_t$: total #defects (detected and undetected)
  - $N_r$: remaining #defects (undetected)

$$N_t = \frac{d_1 d_2}{d_{12}} \quad N_r = N_t - (d_1 + d_2 - d_{12})$$
Capture-Recapture – Example

- Situation: Two inspectors are assigned to inspect the same product
  - $d_1$: 50 defects detected by Inspector 1
  - $d_2$: 40 defects detected by Inspector 2
  - $d_{12}$: 20 defects detected by both inspectors
  - $N_t$: total defects (detected and undetected)
  - $N_r$: remaining defects (undetected)

\[
N_t = \frac{d_1 d_2}{d_{12}} = \frac{50 \cdot 40}{20} = 100 \quad N_r = ?
\]
Capture-Recapture – Example

- Situation: Two inspectors are assigned to inspect the same product
  - $d_1$: 50 defects detected by Inspector 1
  - $d_2$: 40 defects detected by Inspector 2
  - $d_{12}$: 20 defects detected by both inspectors
  - $N_t$: total defects (detected and undetected)
  - $N_r$: remaining defects (undetected)

\[
N_t = \frac{d_1 d_2}{d_{12}} = \frac{50 \cdot 40}{20} = 100 \quad N_r = 100 - (50 + 40 - 20) = 30
\]
Advanced Capture-Recapture Models

• Four basic models used for inspections
  • Difference: Degrees of freedom (see next slide)

• Prerequisites for all models
  • All reviewers work independently of each other
  • It is not allowed to inject or remove faults during inspection
## Advanced Capture-Recapture Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Probability of defect being found is equal across ...</th>
<th>Estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Defect</td>
<td>Reviewer</td>
</tr>
<tr>
<td><strong>M0</strong></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Mt</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mh</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mth</strong></td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Advanced Capture-Recapture Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Probability of defect being found is equal across...</th>
<th>Estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Defect</td>
<td>Reviewer</td>
</tr>
<tr>
<td>M0</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mt</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mh</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mth</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Example: 3 Reviewers
Example:

Maximum-likelihood:

- $Mt =$ total marked animals (=faults) at the start of the $t$'th sampling interval
- $Ct =$ total number of animals (=faults) sampled during interval $t$
- $Rt =$ number of recaptures in the sample $Ct$
- An approximation of the maximum likelihood estimate of population size ($N$) is: $\frac{\text{SUM}(Ct \times Mt)}{\text{SUM}(Rt)}$

First resampling:

$M1 = 11$ (first reviewer)
$C1 = 9$ (second reviewer)
$R1 = 6$ (duplicates)
$N = \frac{11 \times 9}{6} = \frac{99}{6} = 16.5 \rightarrow 2.5$ not found

Second resampling:

$M2 = 14$ (first and second reviewer)
$C2 = 11$ (third reviewer)
$R2 = 10$ (duplicates)
$N = \frac{(11 \times 9 + 14 \times 11)}{6 + 10} = \frac{253}{16} = 15.8 \rightarrow 0.8$ not found

Bug 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
P1 X X X X X X X X X X
P2 X X X X X X X X X X
P3 X X X X X X X X X X

3 Reviewers

14 found

15 found
Mt Model

Maximum-likelihood:

- Mt = total marked animals (=faults) at the start of the t'th sampling interval
- Ct = total number of animals (=faults) sampled during interval t
- Rt = number of recaptures in the sample Ct
- An approximation of the maximum likelihood estimate of population size (N) is: $\frac{\text{SUM}(Ct \times Mt)}{\text{SUM}(Rt)}$
Quality Prediction

- Based on product, process, and people properties

Examples:

- Quality [#Faults] = f(detected #Faults)
- Quality [#Failures per Time/Effort unit] = f(Test Effort | Time)
  
  or = f(detected #Failures)

- Quality [#Faults | Faulty: yes/no] = f(Code Size | Complexity | Churn)
- Quality [Issue Resolution Time] = f(Product & Process & People Features)
- Quality [#Feature requests | Sentiment about Features]
  
  = f(User Feedback)

- Quality [Energy | Performance impact] = f(Code Refactoring)
Reliability Growth Models

- To predict the probability of future failure occurrence based on past (observed) failure occurrence
- Can be used to estimate
  - the number of residual (remaining) faults or
  - the time until the next failure occurs
  - the remaining test time until a reliability objective is achieved
- Application typically during system test
Reliability / Availability

• **Reliability:**
  • Probability that no failure occurs up to a certain point in time
  • often measured in terms of ‘Mean-Time-Between-Failures’ (MTBF)

  \[
  \text{Reliability} = \exp\left(\frac{-t}{\text{MTBF}}\right)
  \]
  
  if \( t = \text{MTBF} \) then \( \text{Rel} = 36.8\% \)

• **Availability**
  • 1 - Percentage of down-time in a given time frame
Reliability / Availability

• Reliability:
  • Probability that no failure occurs up to a certain point in time
  • often measured in terms of ‘Mean-Time-Between-Failures’ (MTBF)

\[
\text{Reliability} = \exp\left(\frac{-t}{\text{MTBF}}\right)
\]

if \( t = \text{MTBF} \) then \( \text{Rel} = 36.8\% \)

• Availability
  • 1 - Percentage of down-time in a given time frame
## Availability (high-level)

<table>
<thead>
<tr>
<th>Availability %</th>
<th>Downtime per year</th>
<th>Downtime per month</th>
<th>Downtime per week</th>
<th>Downtime per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.55555555% (&quot;nine fives&quot;)</td>
<td>162.33 days</td>
<td>13.53 days</td>
<td>74.92 hours</td>
<td>10.67 hours</td>
</tr>
<tr>
<td>90% (&quot;one nine&quot;)</td>
<td>36.53 days</td>
<td>7.31 hours</td>
<td>50.40 minutes</td>
<td>7.20 minutes</td>
</tr>
<tr>
<td>95% (&quot;one nine five&quot;)</td>
<td>18.26 days</td>
<td>3.65 hours</td>
<td>20.16 minutes</td>
<td>2.88 minutes</td>
</tr>
<tr>
<td>97%</td>
<td>10.96 days</td>
<td>1.83 days</td>
<td>5.04 minutes</td>
<td>43.20 minutes</td>
</tr>
<tr>
<td>98%</td>
<td>7.31 days</td>
<td>1.83 days</td>
<td>1.01 minutes</td>
<td>8.64 seconds</td>
</tr>
<tr>
<td>99% (&quot;two nines&quot;)</td>
<td>3.65 days</td>
<td>7.31 hours</td>
<td>6.05 seconds</td>
<td>43.20 seconds</td>
</tr>
<tr>
<td>99.5% (&quot;two nines five&quot;)</td>
<td>1.83 days</td>
<td>1.83 days</td>
<td>1.01 minutes</td>
<td>8.64 seconds</td>
</tr>
<tr>
<td>99.8%</td>
<td>17.53 hours</td>
<td>43.83 minutes</td>
<td>10.08 minutes</td>
<td>1.44 minutes</td>
</tr>
<tr>
<td>99.9% (&quot;three nines&quot;)</td>
<td>8.77 hours</td>
<td>87.66 minutes</td>
<td>20.16 minutes</td>
<td>2.88 minutes</td>
</tr>
<tr>
<td>99.95% (&quot;three nines five&quot;)</td>
<td>4.38 hours</td>
<td>21.92 minutes</td>
<td>5.04 minutes</td>
<td>43.20 seconds</td>
</tr>
<tr>
<td>99.99% (&quot;four nines&quot;)</td>
<td>52.60 minutes</td>
<td>4.38 minutes</td>
<td>1.01 minutes</td>
<td>8.64 seconds</td>
</tr>
<tr>
<td>99.995% (&quot;four nines five&quot;)</td>
<td>26.30 minutes</td>
<td>2.19 minutes</td>
<td>30.24 seconds</td>
<td>4.32 seconds</td>
</tr>
<tr>
<td>99.999% (&quot;five nines&quot;)</td>
<td>5.26 minutes</td>
<td>26.30 seconds</td>
<td>6.05 seconds</td>
<td>864.00 milliseconds</td>
</tr>
<tr>
<td>99.9999% (&quot;six nines&quot;)</td>
<td>31.56 seconds</td>
<td>2.63 seconds</td>
<td>604.80 milliseconds</td>
<td>86.40 milliseconds</td>
</tr>
<tr>
<td>99.99999% (&quot;seven nines&quot;)</td>
<td>3.16 seconds</td>
<td>262.98 milliseconds</td>
<td>60.48 milliseconds</td>
<td>8.64 milliseconds</td>
</tr>
<tr>
<td>99.999999% (&quot;eight nines&quot;)</td>
<td>315.58 milliseconds</td>
<td>26.30 milliseconds</td>
<td>6.05 milliseconds</td>
<td>864.00 microseconds</td>
</tr>
<tr>
<td>99.9999999% (&quot;nine nines&quot;)</td>
<td>31.56 milliseconds</td>
<td>2.63 milliseconds</td>
<td>604.80 microseconds</td>
<td>86.40 microseconds</td>
</tr>
</tbody>
</table>
Reliability Growth Models (RGMs)

Purpose:
Stop testing when

a) a certain percentage (90%, 95%, 99%, 99.9%, …) of estimated total number of failures has been reached

b) a certain failure rate has been reached

Cumulative #Failures ($\mu$) vs. Test Intensity ($\tau$)

$\mu(\tau) = \int_{0}^{\tau} \lambda_0 e^{-\frac{\lambda_0 t}{v_0}} dt$
Failure Data Format /1

1) Time of failure
2) Time interval between failures
3) Cumulative failure up to a given time
4) Failures experienced in a time interval

<table>
<thead>
<tr>
<th>Failure no.</th>
<th>Failure times (hours)</th>
<th>Failure interval (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>58</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>70</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>103</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>125</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>150</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>169</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>199</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>231</td>
<td>32</td>
</tr>
<tr>
<td>14</td>
<td>256</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>296</td>
<td>40</td>
</tr>
</tbody>
</table>
Failure Data Format /2

1) Time of failure
2) Time interval between failures
3) Cumulative failure up to a given time
4) Failures experienced in a time interval

<table>
<thead>
<tr>
<th>Time</th>
<th>Cumulative Failures</th>
<th>Failures in interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>90</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>120</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>150</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>180</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>210</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>240</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>270</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>300</td>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>
Failure Data Format /3
Many types of RGMs exist (>50)

- Exponential growth model (with asymptote)
- S-shaped growth model (without asymptote)
- Logarithmic growth model (without asymptote)
- Exponential splines model (without asymptote)
Model Selection

Many different RGMs have been proposed (>100)

To choose a reliability model, perform the following steps:

1. Collect failure data
2. Examine data (failure data vs. test time/effort)
3. Select a set of candidate models
4. Estimate model parameters for each candidate model
   Least squares method
   Maximum likelihood method
5. Customize model using the estimated parameters
6. Compare models with goodness-of-fit test and select the best
7. Make reliability predictions with selected model(s)
Quality Prediction

- Based on product, process, and people properties

Examples:
- Quality [#Faults] = f(detected #Faults)
- Quality [#Failures per Time/Effort unit] = f(Test Effort | Time)
  or = f(detected #Failures)
- Quality [#Faults | Faulty: yes/no] = f(Code Size | Complexity | Churn)
- Quality [Issue Resolution Time] = f(Product & Process & People Features)
- Quality [#Feature requests | Sentiment about Features] = f(User Feedback)
- Quality [Energy | Performance impact] = f(Code Refactoring)
General Idea

Features (with Importance)

Data (from past) = known characteristics (features) & known outcome

Modeling Method
- Regression
- Classification / Clustering
- Neural Networks (Deep Learning)
- etc.
using many techniques:
- Text mining
- Topic Modeling
- Decision Trees
- Random Forest
- etc.

Outcomes (predicted):
- #Defects
- #Failures
- Defective: yes/no
- Missing/Good/Bad
- Feature
- Time/Effort to resolve
- Severity
- Who shall resolve/fix?
- Impact?
- …

Model that predicts outcome based on characteristics (features)
Example 1: Predicting Fault-Proneness

OO-Metrics:
- Inheritance
- Coupling
- Cohesion
- …

Example 2a: Predicting IRT (Issue Resolution Time)

Static Features:
- Name
- Submitter
- Type
- ...
- Description

Text Mining

Issue

Issue Resolution Time

IRT

Prediction Model

Example 2b: Predicting IRT (Issue Resolution Time)

Static Features:
- Name
- Submitter
- ...
- Description

Dynamic Features:
- Comments
- ...

IRT:
- Before X?
- After X?

More Examples …

• Classifying Severity of Issues
• Assigning Developer to Issue
• Classifying User Reviews
  • Feature Request
  • Failure Report
  • Feature Evaluation
• …
Structure of Lecture 12

• Quality Estimation
• Lab 11
• Test Planning & Documentation
• Test Organization
• Test Process Improvement (Test Maturity Model)
Lab 11 – Document Inspection & Defect Prediction

Lab 11 (week 37: May 10 & 11) - Document Inspection and Defect Prediction (9 points)

HW 10 Instructions & Sample Documentation

Submission Deadlines:
- Tuesday Labs: Monday, 16 May, 23:59
- Wednesday Labs: Tuesday, 17 May, 23:59

- Penalties apply for late delivery:
  - 50% penalty, if submitted up to 24 hours late;
  - 100% penalty, if submitted more than 24 hours late

Instructions

Documentation:
Requirements List (User Stories)
Specification
- 2 Screens
- 1 Text
Lab 11 – Document Inspection & Defect Prediction (cont’d)

**Phase A:** Individual student work

Instructions

Requirements (6 User Stories)

Inspection of Specification against Requirements

Specification (excerpt) 2 screens & Text

Issue List (at least 8 defects in Spec)

1 Student

**Phase B:** Pair work

Issue List Student 1

Issue List Student 2

Student Pair

Consolidated Issue List

Remaining Defects Estimation

Table columns:
ID, Description, Location, Type, Severity
Lab 11 – Document Inspection & Defect Prediction (cont’d)

Lab/HW 11: Must work in pairs to be able to get full marks!
Lab 11 – Document Inspection & Defect Prediction (cont’d)

Lab/HW 11: If total number of students is odd, one 3-person team ok
Structure of Lecture 12

- Quality Estimation
- Lab 11
- Test Planning & Documentation
- Test Organization
- Test Process Improvement (Test Maturity Model)
Test Planning

- Objectives
- What to test
- Who will test
- When to test
- How to test
- When to stop

Elective course (Fall): Hands-on SW Testing
MTAT.03.294
IEEE 829-2008: Standard for Software and System Test Documentation

FIG. 7.4
Hierarchy of Test Plans

<table>
<thead>
<tr>
<th>Test Plan Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Test plan identifier</td>
</tr>
<tr>
<td>2. Introduction</td>
</tr>
<tr>
<td>3. Items to be tested</td>
</tr>
<tr>
<td>4. Features to be tested</td>
</tr>
<tr>
<td>5. Approach</td>
</tr>
<tr>
<td>6. Pass/fail criteria</td>
</tr>
<tr>
<td>7. Suspension and resumption criteria</td>
</tr>
<tr>
<td>8. Test deliverables</td>
</tr>
<tr>
<td>9. Testing Tasks</td>
</tr>
<tr>
<td>10. Test environment</td>
</tr>
<tr>
<td>11. Responsibilities</td>
</tr>
<tr>
<td>12. Staffing and training needs</td>
</tr>
<tr>
<td>13. Scheduling</td>
</tr>
<tr>
<td>14. Risks and contingencies</td>
</tr>
<tr>
<td>15. Testing costs</td>
</tr>
<tr>
<td>16. Approvals</td>
</tr>
</tbody>
</table>

- **Software quality assurance (V&V) plan**
  - **Master test plan**
    - Unit test plan
    - Integration test plan
    - System test plan
    - Acceptance test plan
  - Review plan: Inspections and walkthroughs
Test plan according to IEEE Std 829-2008 (Appendix II)

a) Test plan identifier
b) Introduction
c) Test items
d) Features to be tested
e) Features not to be tested
f) Approach
g) Item pass/fail criteria
h) Suspension criteria and resumption requirements
i) Test deliverables
j) Testing tasks
k) Environmental needs
l) Responsibilities
m) Staffing and training needs
n) Schedule
o) Risks and contingencies
p) Approvals
Test Plan (1)

a) Test plan identifier

b) Introduction
   – Product to be tested, objectives, scope of the test plan
   – Software items and features to be tested
   – References to project authorization, project plan, QA plan, CM plan, relevant policies & standards

c) Test items
   – Test items including version/revision level
   – Items include end-user documentation
   – Defect fixes
   – How transmitted to testing
   – References to software documentation

Slide not discussed in lecture; Only meant as background info
d) Features to be tested
   – Identify test design / specification techniques
   – Reference requirements or other specs

e) Features not to be tested
   – Deferred features, environment combinations, …
   – Reasons for exclusion

f) Approach
   – How you are going to test this system
     • Activities, techniques and tools
   – Detailed enough to estimate
   – Completion criteria (e.g. coverage, reliability)
   – Identify constraints (environment, staff, deadlines)
Test Plan (3)

g) Item pass/fail criteria
   – What constitutes success of the testing
   – Coverage, failure count, failure rate, number of executed tests, …
   – Is NOT product release criteria

h) Suspension and resumption criteria
   – For all or parts of testing activities
   – Which activities must be repeated on resumption

i) Test deliverables
   – Test plan
   – Test design specification, Test case specification
   – Test procedure specification, Test item transmittal report
   – Test logs, Test incident reports, Test summary reports

Slide not discussed in lecture; Only meant as background info
Test Plan (4)

j) Testing tasks
   – Including inter-task dependencies & special skills
   – Estimates

k) Environment
   – Physical, hardware, software, tools
   – Mode of usage, security, office space
   – Test environment set-up

l) Responsibilities
   – To manage, design, prepare, execute, witness, check, resolve issues, providing environment, providing the software to test

m) Staffing and Training needs
Test Plan (5)

n) Schedule
   – Test milestones in project schedule
   – Item transmittal milestones
   – Additional test milestones (environment ready)
   – What resources are needed when

o) Risks and Contingencies
   – Testing project risks
   – Contingency and mitigation plan for each identified risk

p) Approvals
   – Names and when approved
Test Case Specification – Why?

• Organization
  – All testers and other project team members can review and use test cases effectively

• Repeatability
  – Know what test cases were last run and how so that you could repeat the same tests

• Tracking
  – What requirements or features are tested?
  – Tracking information’s value depends on the quality of the test cases

• Evidence of testing
  – Confidence (quality)
  – Detect failures
Failure/Issue Report
(Test incidence report)

• Summary
• Incident Description
• Impact
Failure/Issue Report

(Test incidence report)

Summary

• This is a summation/description of the actual incident.
  – Provides enough details to enable others to understand how the incident was discovered and any relevant supporting information

• References to:
  – Test Procedure used to discover the incident
  – Test Case Specifications that will provide the information to repeat the incident
  – Test logs showing the actual execution of the test cases and procedures
  – Any other supporting materials, trace logs, memory dumps/maps etc.
Failure/Issue Report

(Test incidence report)

Incident Description

• Provides as much details on the incident as possible.
  – Especially if there are no other references to describe the incident.

• Includes all relevant information that has not already been included in the incident summary information or any additional supporting information

• Information:
  – Inputs
  – Expected Results
  – Actual Results
  – Anomalies
  – Date and Time
  – Procedure Step
  – Attempts to Repeat
  – Testers
  – Observers
Failure/Issue Report
(Test incidence report)

Impact

• Describe the actual/potential damage caused by the incident.
  – Severity
  – Priority

• Severity and Priority need to be defined so as to ensure consistent use and interpretation, for example:
  – Severity
    – Mission Critical - Application will not function or system fails
    – Major - Severe problems but possible to work around
    – Minor – Does not impact the functionality or usability of the process but is not according to requirements/design specifications

• Priority – The order in which the incidents are to be addressed
  – Immediate – Must be fixed as soon as possible
  – Delayed – System is usable but incident must be fixed prior to next level of test or shipment
  – Deferred – Defect can be left in if necessary doe to time or costs
Test results report

- Test cases executed
- Versions tested
- Defects found and reported

Figure 6. Defect distributions, showing number of defects detected over time, for the three studied projects. Ship dates indicated.
Standards

- IEEE 829-2008
  Standard for Software Test Documentation
- IEEE 1008-1993
  Standard for Software Unit Testing
- IEEE 1012-2012
  Standard for System and Software Verification and Validation
- ISO/IEC/IEEE 29119 Software Testing (5 parts)
  - replaces most of the older standards
Structure of Lecture 12

- Quality Estimation
- Lab 11
- Test Planning & Documentation
- Test Organization
- Test Process Improvement (Test Maturity Model)
7 approaches to test organisation

1. Each person’s responsibility
2. Each unit’s responsibility
3. Dedicated resource
4. Test organisation in QA
5. Test organisation in development
6. Centralized test organisation
7. Test technology centre

[Kit, Software Testing in the Real World Ch 13, 1995]
7 approaches to test organisation

1. Each person’s responsibility
2. Each unit’s responsibility
3. Dedicated resource
4. Test organisation in QA
5. Test organisation in development
6. Centralized test organisation
7. Test technology centre

[Kit, Software Testing in the Real World Ch 13, 1995]
1. Each person’s responsibility

+ Natural solution

- Testing own software
2. Each unit’s responsibility

+ Solves dependency problem

- Two tasks
  - Double competency?
3a. Dedicated resource

+ Solves multiple task problem
+ Single team

- Management of two types
- Competency provision
3b. Dedicated resource on a large scale

- Solves mgmt problem of 3a

- Where to put in the organization?
4. Test organisation in QA

+ Solves mgmt problem of 3b
- Teamwork problems?
- TDG lost in QAO
- PDG not responsible for final product
5. Test organisation in development

+ Solves mgmt problem of 4
+ May solve teamwork problem of 4

- TDG dependent on PD management communication & support
6. Centralized test organisation

+ Solves mgmt problem of 5
+ Career path for test mgmt

- VP key for test
- Teamwork at low level?
- Consistency of methods?
7. Test technology centre

- VP key for test
- Teamwork at low level?

+ Solves consistency problem of 6
Which organization should we choose?

- Depending on
  - size
  - maturity
  - focus

- The solution is often a mix of different approaches
Outsourced Testing = Testing-as-a-Service (TaaS)

- Company A subcontracts Company B to do their testing
- Company B manages the test project and assigns freelance testers to specific test tasks
- Example of Company B: Testlio
Good Software Tester

- Technical Skills
- Analytical Skills
- Passion
- Attitude
- Verbal and Written Communication
- Productivity
Structure of Lecture 12

• Quality Estimation
• Lab 11
• Test Planning & Documentation
• Test Organization
• Test Process Improvement (Test Maturity Model)
Process quality and product quality

- Quality in process
- Quality in product
- Project:
  - instantiated process
- ISO 25000:
  - Process quality contributes to improving product quality, which in turn contributes to improving quality in use
Process improvement models vs Test Process improvement models

• (Integrated) Capability maturity model (CMM, CMMI)
• Software process improvement and capability determination (SPICE)
• ISO 9001, Bootstrap, …

Test Process Improvement Models:
• Test maturity model (TMM, TMMi)
• Test process improvement model (TPI)
• Test improvement model (TIM)
• Minimal Test Practice Framework (MTPF)
• …
Test Maturity Model (TMM)

- Levels
- Maturity goals and sub-goals
  - Scope, boundaries, accomplishments
  - Activities, tasks, responsibilities
- Assessment model
  - Maturity goals
  - Assessment guidelines
  - Assessment procedure
Level 2: Phase Definition

- Institutionalize basic testing techniques and methods
- Initiate a test planning process
- Develop testing & debugging tools

Slide not discussed in lecture; Only meant as background info
Level 3: Integration

- Control and monitor the testing process
- Integrate testing into software life-cycle
- Establish a technical training program
- Establish a software test organization

Level 5: Optimization, Defect Prevention, and Quality Control
- Test process optimization
- Quality control
- Application of process data for defect prevention

Level 4: Management and Measurement
- Software quality evaluation
- Establish a test measurement program
- Establish an organization-wide review program

Level 3: Integration
- Control and monitor the testing process
- Integrate testing into the software lifecycle
- Establish a technical training program
- Establish a software test organization

Level 2: Phase Definition
- Institutionalize basic testing techniques and methods
- Initiate a test planning process
- Develop testing and debugging goals

Level 1: Initial

Slide not discussed in lecture;
Only meant as background info
Level 4: Management and Measurement

- Software quality evaluation
- Establish a test management program
- Establish an organization-wide review program

Slide not discussed in lecture; Only meant as background info
Level 5: Optimizing, Defect Prevention, and Quality Control

- Test process optimization
- Quality control
- Application of process data for defect prevention

Slide not discussed in lecture; Only meant as background info
Next Week

• Quiz 11 (starting today):
  – Quality Estimation
  – Test Documentation, Organization and Process Improvement

• HW 10: Finish and submit on time

• Lab 11:
  – Document Inspection and Defect Prediction

• Lecture 13:
  – Exam Preparation & Q/A Session
Guest Lectures (Pre-Recorded)

Guest Lectures (19 May) – Keynotes at ESEC/FSE 2019 in Tallinn

- Safety and Robustness for Deep Learning with Provable Guarantees
  - by Marta Kwiatkowska, University of Oxford, UK

- Living with Feature Interactions
  - by Joanne Atlee, University of Waterloo, Canada

Not relevant for exam!