Testing: basic terms - 1

- Unit testing: testing of single compilation unit; often requires driver & stubs
- Stress testing: test code under extreme conditions
- System testing: put all of the pieces together and test everything

Testing: basic terms - 2

- Beta testing: testing by “special” customers who understand errors are likely
- Acceptance testing: testing on customer’s site
- Reliability testing: testing to measure reliability
- Regression testing: testing ensure that changes did not break old features
In software reliability world, failures are treated as "random" events (even though they aren't)

Why can code sometimes behave differently even using the same data?

Each program has "input space":

The set of data the program will receive when running, including the fact that it will see some inputs frequently, others rarely.

Then probability of program failure is based on concept:

Randomly pick a data value from the program's input space (where selecting particular values reflects how frequently they will be seen when the program really runs).

Observe whether or not the program fails

Do this for all values in input space (properly weighted).

Pr(failure with a randomly selected input) is estimated by running many (perhaps millions) test cases

\[ \Pr(\text{failure}) = \frac{\text{number failures}}{\text{number tests}} \]

Simple concept. But involves several potentially hard problems:

What is the distribution of inputs the program will really encounter in use?

Input space usually too large to run program with all possible inputs

How do you tell when the program fails (assuming it doesn't crash)?

Typical industry interest is how often software will fail when used by their customers. This is a similar idea.

Depends on both customer's frequency of use and their typical input data.
N-Version Software

- Purpose: initially, increase software reliability
- One idea:
  - Develop one requirements document
  - Give copies to several different independent development groups
  - Each group develops complete system as if they were the only developer
    - No info exchange among groups!!
  - Run all "identical" systems:
    - Give each version the exactly the same input
    - Wait for each to produce required output
    - Compare each outputs & take majority as the correct output
- Question: does this work?

N-Version Software - 2

- Statistical concept: event independence
  - Knowing one event occurred doesn’t change probability of another event.
  - Independence $\iff Pr(A^B) = Pr(A) \times Pr(B)$
    - Often, this is not true
- Key Question:
  - Do separately developed versions really fail independently?
  - Or if one fails, does that make it more likely that another will fail also?

Failure Probabilities for N-Version System

- Assume a 3-version system
- Assume each version has a failure probability of $10^{-3}$
- Assume independent failures
- How can this system fail?
  - If at least two versions get the wrong answer.
  - This happens with probability $10^{-3} \cdot 10^{-3}$
Oracle Problem - 1

- When testing software, how do you detect all erroneous outputs?
  - Unsolved SE problem for many applications

Oracle Problem - 2

- Federal Aviation Administration (FAA) specifies that the failure rate for commercial aircraft of less than 10^-9.
- How many test cases should you run to determine this?
  - Generally considered a statistical problem: if the software is given a "random" input, what is the likelihood of the program producing incorrect output (or crashing or looping forever)?
- How do you detect wrong answers?
  - Remember, for 10^-9, need to run lots of test cases

Team Project: Software Reliability Estimator

Design (Version 1):
- Input desired number of tests: n
- Repeat n times // typical n: 100,000 to
// 1,000,000,000
- Generate random input from program's input space
- Run the program
- Check the answer
- If incorrect, increment error count
- Output failure probability estimate:
  (error count)/n
Version 1 Design Review

- What’s wrong with this design?
  
  (In your analysis, you should mention the Oracle Problem)

Version 2 Design

- Assume we have several versions of the same program, all "reasonably" correct
  
  Input desired number of tests, say n
  
  Repeat n times
    
    Generate random input from program’s input space
    
    Run each version with this input
    
    Compare all answers
    
    If disagreement, increment error count
  
  Output failure probability estimate:
    
    (error count)/n

Version 2 Design Review

- Questions:
  
  - How hard is it to run several programs from another program?
  
  - What happens if one of the programs being tested fails?
    
    • Say it generates a wrong answer?
    
    • Say it goes into an infinite loop
    
    • Say it generates a run-time error and is terminated by the operating system?
  
  - How do you generate random inputs?
    
    • What if inputs are text?
    
    • What if inputs are numeric?
    
    • Say it generates a wrong answer?
Reading

- TSP text, Ch. 4, 5
- Chapters 11 – 15, depending on your role.

Group Meetings

- Group meetings in recitation
- Used to meet, set group goals, set up project notebook
- Agenda:
  - Be prepared to describe your background to group
  - Be prepared to state your goals for group project

Outline

- More forms
- Development strategy
- Development plan
A development strategy

- Lots of different approaches to software development
  - Develop entire system
  - Develop system in small pieces
- Most prefer cycle (but not always possible):
  - Code a little
  - Test a little
  - Have something running early-on
    - Always be ready to run demo (if feasible)
- Focus on uncertainty early
  - Identify and resolve risks

Steps – Planning and schedule

- Plan first
  - Big picture first, then refine details
- Can do top-level planning
  - We're lucky: we have real deadlines
    - Hard constraints eliminate possibly confusing choices
  - Review outline from last slide set.
    - Be ready to discuss with group on Friday
      - Feasible
      - Does it need changing?

Steps – Conceptual design

- Must omit details
- Identify major pieces
  - What are they
  - How will they likely interact
  - What depends on what?
- Can some components be completed first and others brought in later
  - (Strategy: make sure you have something for the grader to run, even if some major parts are missing! -- if worst comes to worst)
Identify major risks

- Pieces you can’t code?
- How to test?
- What are problems common to working in teams?
  - Some team members will likely have experience
- Try to anticipate so you can mitigate
  - Key components have two developers: coder and tester
    - Either could step in for the other if necessary

Productivity comes from reuse (but from your own code!)

- Identify code you already own that might reduce effort in project
- Plan reuse library
  - During design
    - Look for reuse from this class
    - Identify new components that have potential for reuse

Strategy scripts: overview

<table>
<thead>
<tr>
<th>Purpose</th>
<th>To guide team through a TSPi development strategy; have preliminary size and time estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry criteria</td>
<td>Everyone’s read ch. 4. TSPi process discussed in class. Objectives discussed in class. Teams assigned and roles assigned. Teams have agreed on goals for team and their own work.</td>
</tr>
<tr>
<td>General</td>
<td>Dev strategy specifies order of tasks (code, test, etc.) How to divide work within group. It may change; keep it up to date. Time estimates provide basis for work allocation.</td>
</tr>
</tbody>
</table>
Development Strategy - steps

1. Strategy overview
2. Establish strategy criteria
3. Produce conceptual design
4. Select development strategy
5. Produce prelim. estimates
6. Produce prelim. schedules
7. Assess risks
9. Produce config. mgmt plan

Possible development strategy ideas:

- Form alliance with another group to exchange test data.
  - No copying of data, but pooling might be ok if planned and discussed with instructor
- Form alliance with another group to exchange test programs.
  - Rule: no exchange of source but look for other ideas
  - If anything like this is done; identify authors, etc.
- These will take time; are they worth it?
- Other ways to reduce work/improve grades?
- Note: intergroup activities are encouraged but be careful: plan, then check with instructor.

Development plan - 1

- One size does not fit all!
  - Simple tasks only require simple plans
  - Failure costs effect how much planning is appropriate
  - Tight schedules effect appropriate planning detail
  - Essence of teamwork in meeting commitments
    - Need to know what you’re being asked to commit so you can determine if you can finish by the deadline
    - Need to balance work among team members
Development plan - 2

- Need to track progress precisely
  - Need to know that when x% of money has been spent that the project is x% complete (or when x% of schedule has been used up, x% of project is complete)
- Hence planned value and earned value tracking
  - Planned value of a task is its percentage of total project effort
  - Earned value of a task is 0 until it is 100% complete (to keep us from deceiving ourselves)

Planned-value example

<table>
<thead>
<tr>
<th>Task</th>
<th>Plan hours</th>
<th>Cumulative hours</th>
<th>Plan Value</th>
<th>Cumulative Plan Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch</td>
<td>29.5</td>
<td>29.5</td>
<td>8.0</td>
<td>8.0</td>
</tr>
<tr>
<td>Planning</td>
<td>27.5</td>
<td>57.0</td>
<td>7.5</td>
<td>15.5</td>
</tr>
<tr>
<td>System test plan</td>
<td>17.0</td>
<td>74.0</td>
<td>4.6</td>
<td>20.2</td>
</tr>
<tr>
<td>Documentation</td>
<td>33.8</td>
<td>354.5</td>
<td>9.0</td>
<td>96.7</td>
</tr>
<tr>
<td>Postmortem</td>
<td>12.2</td>
<td>367.0</td>
<td>3.3</td>
<td>100.0</td>
</tr>
<tr>
<td>Total</td>
<td>367.0</td>
<td></td>
<td>100.0</td>
<td></td>
</tr>
</tbody>
</table>

Remember, these are team numbers, so divide by 5 to get average individual numbers!

TSPi Planning Overview
STRAT form

- List, item by item
  - Exactly what you plan to implement
  - Page and paragraph reference to Requirements Doc (from NASA)
  - (If the semester were longer) in what phase you plan to implement it. In 350, the planned order of development

Size summary (SUMS) form

- For each software unit, report estimates for:
  - size
  - base (if you are modifying existing code, this is the size of it)
    - deleted (code removed from base, if any)
    - modified (code modified in base)
  - code added
  - code reused
  - new + modified (this is the effort measure)
  - total in new unit
  - loc of new code that might be reused
- Repeat with actuals after unit is complete

TASK form

- For each task, include (one line/task):
  - planned time, by group member:
    - Team leader
    - Development mgr
    - Planning mgr
    - Quality/process mgr
    - Support manager
  - planned size/value
    - size
    - date
  - Planned value
    - actuals
      - hours
      - date
### Sample SCHEDULE form

<table>
<thead>
<tr>
<th>Week No</th>
<th>Date</th>
<th>Direct Hrs</th>
<th>Cum Hrs</th>
<th>Cum Planned Value</th>
<th>Team Hrs</th>
<th>Cum Hrs</th>
<th>Weeks Planned Value</th>
<th>Cum Planned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10/29</td>
<td>26.6</td>
<td>26.6</td>
<td>18.5</td>
<td>27.0</td>
<td>27.0</td>
<td>20.5</td>
<td>20.5</td>
</tr>
<tr>
<td>2</td>
<td>11/5</td>
<td>56.5</td>
<td>83.5</td>
<td>34.0</td>
<td>44.4</td>
<td>71.4</td>
<td>62.5</td>
<td>83.0</td>
</tr>
<tr>
<td>3</td>
<td>11/12</td>
<td>47.0</td>
<td>130.3</td>
<td>45.0</td>
<td>38.5</td>
<td>107.2</td>
<td>37.5</td>
<td>120.5</td>
</tr>
<tr>
<td>4</td>
<td>11/19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Form WEEK – for weekly reports

<table>
<thead>
<tr>
<th>Group Weekly Data</th>
<th>Planned</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project hours this wk:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project hours to date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Earned value to date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total hours for tasks completed to date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team member weekly data</td>
<td>Hours</td>
<td>Hours</td>
</tr>
<tr>
<td>Team leader</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planning manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality/Process manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development Tasks Completed</td>
<td>Hours</td>
<td>Planned</td>
</tr>
<tr>
<td>Issue/Risk Tracking</td>
<td>Issue/ Risk name</td>
<td>Status</td>
</tr>
<tr>
<td>Other Significant Items</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SUMQ form (Quality plan) - 1

<table>
<thead>
<tr>
<th>Summary Rates</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC/hr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% Reuse (of total LOC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% New Reuse (of new and changed LOC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percent Defect-free (PDF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In compile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In unit test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In build and integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect/page</td>
<td>Requirements inspection</td>
<td>Driver related</td>
</tr>
<tr>
<td>HLD inspection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defects/LOC</td>
<td>DLD review</td>
<td>Code review</td>
</tr>
<tr>
<td>DLD inspection</td>
<td>DLD inspection</td>
<td>Code review</td>
</tr>
<tr>
<td>Code review</td>
<td>Compile</td>
<td>Code inspection</td>
</tr>
<tr>
<td>Compile</td>
<td>Unit test</td>
<td>Build and integration</td>
</tr>
<tr>
<td>Code inspection</td>
<td>Build and integration</td>
<td>System test</td>
</tr>
</tbody>
</table>
### SUMQ form (Quality plan) -2

<table>
<thead>
<tr>
<th>Defect ratios</th>
<th>Planned</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>code review/compile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLD review/Unit test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development time ratios</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements inspection/Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLD inspection/HLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLD/code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLD review/DLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code review/code</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### A/FR Review rates

<table>
<thead>
<tr>
<th>DLD lines/hour</th>
<th>Code LOC/hour</th>
</tr>
</thead>
</table>

### Inspection rates

<table>
<thead>
<tr>
<th>Requirements pages/hour</th>
<th>HLD pages/hour</th>
<th>Code LOC/hour</th>
</tr>
</thead>
</table>

### SUMQ form (Quality plan) -3

<table>
<thead>
<tr>
<th>Defect-injection rates (Defects/hr.)</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build and integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Defect-removal rates (Defects/hr.)

<table>
<thead>
<tr>
<th>Requirements inspection</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build and integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SUMQ form (Quality plan) -4

<table>
<thead>
<tr>
<th>Phase yields</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements inspection</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Process yields

<table>
<thead>
<tr>
<th>% before compile</th>
<th>% before unit test</th>
<th>% before build and integration</th>
<th>% before system test</th>
<th>% before system delivery</th>
</tr>
</thead>
</table>
Plan Summary (SUMP)

<table>
<thead>
<tr>
<th>Product size</th>
<th>Plan</th>
<th>Actual</th>
<th>Actual %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements pages</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other text pages</td>
<td></td>
<td></td>
<td>may not be any</td>
</tr>
<tr>
<td>High-level design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed design lines</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base LOC (B) measured</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deleted LOC (D)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modified LOC (M)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Added LOC (A)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reused LOC (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total LOC (T)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total New and Changed LOC (N)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total New Reused LOC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimated Object LOC (E)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper Prediction Interval (70%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower Prediction Interval (90%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time in Phase (hours)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Plan steps - 1

<table>
<thead>
<tr>
<th>Step</th>
<th>Activities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Planning overview</td>
<td>How plan is produced</td>
</tr>
<tr>
<td></td>
<td></td>
<td>How quality plan is produced</td>
</tr>
<tr>
<td>2</td>
<td>Sizes estimates in STRAT</td>
<td>Starting with conceptual design, Planning Manager leads team in identifying missing parts and their sizes; record on STRAT form</td>
</tr>
<tr>
<td>3</td>
<td>Produce task plan</td>
<td>Planning manager leads team through producing task list with team and individual time estimates; enter in TASK form</td>
</tr>
<tr>
<td>4</td>
<td>Produce schedule plan</td>
<td>Planning manager obtains estimated hrs for each member plans to spend on project each week, and enters this in SCHEDULE form; produces team TASK and SCHEDULE forms; reworks if needed</td>
</tr>
</tbody>
</table>

TSPi Development plan script

<table>
<thead>
<tr>
<th>Purpose</th>
<th>To guide a team through producing individual and team task, schedule, and quality plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>#Team has development strategy and conceptual design</td>
</tr>
<tr>
<td></td>
<td>#Members have read text Ch. 5</td>
</tr>
<tr>
<td>General</td>
<td>The plan defines</td>
</tr>
<tr>
<td></td>
<td>#Time required to perform each task</td>
</tr>
<tr>
<td></td>
<td>#Planned value for each task</td>
</tr>
<tr>
<td></td>
<td>#The schedule plan gives</td>
</tr>
<tr>
<td></td>
<td>#Each person’s planned time for each project week</td>
</tr>
<tr>
<td></td>
<td>#Anticipated completion week for each task</td>
</tr>
<tr>
<td></td>
<td>#Planned value for each week</td>
</tr>
<tr>
<td></td>
<td>This will likely involve some iterations</td>
</tr>
<tr>
<td>Exit criteria</td>
<td>Completed team and individual TASK and SCHEDULE forms</td>
</tr>
<tr>
<td></td>
<td>#Reviewed and approved complete task list</td>
</tr>
<tr>
<td></td>
<td>Completed SUMP, SUMQ, and SUMS forms</td>
</tr>
<tr>
<td></td>
<td>Update project notebook</td>
</tr>
</tbody>
</table>
Plan steps - 2

5. Produce quality plan
Quality/process manager leads the team through:
- Reviewing team's quality objectives
- Estimating defects injected and defect-removal yields
- Generating and assessing trial SUMP and SUMQ plans
- Making need process adjustments to get a sat. plan

6. Produce individuals' plans
Planning manager helps individual make personal plans:
- Allocate tasks among team members
- Estimate time needed for each task
- Entering data in TASK and SCHEDULE forms
- Producing planned-value schedule & task comp. times

7. Balance team workload
Planning manager leads team through:
- Identifying workload imbalances
- Reallocating task to shorten schedule
- Producing balanced individual plans
- Producing the consolidation team plan (TASK, SCHEDULE, SUMP, SUMQ)

Example forms
- Read chapter 5
- Contains worked-out examples of all of these forms
  - Some forms I only outlined just to talk about terms used on the forms
- Also contains justifications for several of the steps
- It should help you in setting goals for yield, defect removal rates, etc.

Conceptual design
- Tell me exactly what you plan to implement
  - Include object names, but no implementation details
  - Give around 1 sentence description of what each object does
- Goal:
  - High level view of how system will function
  - What pieces need to be implemented; how they're related
- Format: Text, pictures:
  - Whatever is easy to produce
    - (you will need to keep it current)
  - Whatever communicates effectively
    - Within team
    - To instructor & grader
Products

- List all of the products you will produce
  - Names
  - Brief description
  - Size estimates
  - Time estimates
- Include:
  - Conceptual design
  - Test plans
  - Test data
  - Tasks
  - Stubbed routines (if needed)
  - Drivers (if needed)

Tracking the work

- Goal: track progress and quality
- TSPi tool can help with this
- Steps (examples)
  - Record time in time log
  - Enter week tasks are completed
  - TSPi tool can generate TASK and SCHEDULE data
  - Enter defect in defect recording log
  - Enter size of component
  - TSPi tool can generate SUMP form
  - TSPi tool can generate SUMQ form
  - Submit to planning manager for WEEK rpt

Reuse comments

- Try to reuse code developed thus far in this class
- Team must decide which version to use
  - Team can compare members’ versions and pick one
- If you can reuse any team member’s code from other classes, do so.
  - Be sure you document this: in plans and in code
  - If you find other LEGAL code sources (say, on net), let me know