

CS 350, slide set 10

M. Overstreet  
Old Dominion University  
Spring 2006

---

---

---

---

---

---

---

---

Reading

- TSP text, Ch 9, 10
  - Remember, you are supposed to have read the chapter on your role from ch. 11-15
  - And ch. 16, 17, and 18.

---

---

---

---

---

---

---

---

Deadlines, Guidelines - 1

- Project: **due April 30.**
  - Submit to [cs350@cs.odu.edu](mailto:cs350@cs.odu.edu)
  - **I must be able to determine who did what.**  
Name (or names) of who did it must be included with each item.
  - **I must be able to determine what works.**  
Include actual output whenever appropriate.
- See web site checklist section for a complete list of due dates

---

---

---

---

---

---

---

---

## Project evaluation

- Individual project grades determined as follows
  - Your peer evaluations: 15%
  - My impression of your contributions ±15%
  - Forms: 35%
  - Evidence of execution 30%
    - Note that without execution, some forms must be incomplete
  - Quality/completeness of materials 10%
- As I go through each group's submissions, I will record what you've done. Your grade will be based on that list, your peer evals (if I can believe them), and my impression of your contributions

---

---

---

---

---

---

---

---

## Testing: selected case studies

- Remember: hard to get this kind of data
- Magellan spacecraft (1989-1994) to Venus
  - 22 KLOC – this is small
  - 186 defects found in system test
    - 42 critical
    - only 1 critical defect found in 1<sup>st</sup> year of testing
  - Project a success, but several software related emergencies
- Galileo spacecraft (1989 launch, 1995)
  - Testing took 6 years
  - Final 10 critical defects found after 288 weeks of testing

---

---

---

---

---

---

---

---

## PSP/TSP approach

- Find most defects before integration testing during:
  - Reviews (requirements, HLD, DLD, test plans, code)
  - Inspections
  - Unit testing
- Each of these activities is expensive, but testing is worse
- TSP goal: use testing to confirm that code is high quality.
  - May need to return low quality code for rework or scrapping
  - Data shows strong relationship between defects found in testing & defects found by customers

---

---

---

---

---

---

---

---

## Build and integration strategies: big bang

- Build & test all pieces separately then put them all together at the end and see what happens
- Out of favor
  - Debugging all pieces at the same time; harder to identify real causes of problems
  - Industry experience: 10 defects/KLOC;
    - All-too-typical: system with 30,000 defects

---

---

---

---

---

---

---

---

## B & I strategies: one subsystem at a time

- Design system so that it can be implemented in steps; each step useful
- First test minimal system
- After its components have been tested
  - Add one component at a time
    - Defects are more likely to come from new parts
- Not all systems admit to this approach

---

---

---

---

---

---

---

---

## B & I strategies: add clusters

- If system has components with dependencies among them, it may be necessary to add clusters of interacting components

---

---

---

---

---

---

---

---

## B & I strategies: top down

- Top-down integration
  - Integrate top-level components first
    - With lower-level components stubbed as necessary
- May identify integration issues earlier than other approaches
- I suggest this approach this project
  - Write top level routines first when feasible.
    - It calls stubbed functions.
  - As modules are available, they replace stubbed version

---

---

---

---

---

---

---

---

## Typical testing goals

- Show system provides all specified functions
  - Does what is supposed to do
- Show system meets stated quality goals
  - MTBF, for example
- Show system works under stressful conditions
  - Doesn't do "bad" things when other systems (e.g. power) fail, network overloads, disk full
- In reality, schedule/budget considerations may limit testing to most frequent or critical behaviors only

---

---

---

---

---

---

---

---

## Test log includes:

- Date, start and end time of tests
- Name of tester
- Which tests were run
- What code & configuration was tested
- Number of defects found
- Test results
- Other pertinent information
  - Special tools, system config., operator actions
- See sample test log, pg. 172

---

---

---

---

---

---

---

---

## Documentation - 1

- Probably needs another course
- Must write from perspective of user of documentation
  - Other programmers on team
  - Future maintenance programmers
  - Installers
  - Managers
  - Users
- Better to hire English majors to write documentation?
  - Easier teach them the computing part than to teach technical geeks how to write well?

---

---

---

---

---

---

---

---

## Documentation - 2

- Developers often do poor job
  - Even when proofing, omissions (what you forgot to tell reader) are often undetected since writer knows them
  - Student just finished MS thesis of software metrics of open-source code. Did not explain what KDSI meant until end of thesis! I missed it too!
- Guidelines: include
  - Glossary to define special terms
  - Detailed table of contents
  - Detailed index
  - Sections on
    - Error messages
    - Recovery procedures
    - Troubleshooting procedures

---

---

---

---

---

---

---

---

## Postmortem script

- We'll skip; works better if 3 cycles
- Will discuss in class; be ready to tell me
  - Where the process worked and where it did not
  - How did actual performance compare with expected?
  - Where did your team do well? Where not?

---

---

---

---

---

---

---

---

## PIP objectives

- While project is fresh, record good ideas on process improvements
- Implicit goal: be skeptical about TSP as the solution to all software problems
  - Each organization and problem domain probably has their unique problems; one size does not fit all
- But a request: be tolerant. Learn from others experience; don't reject too quickly

---

---

---

---

---

---

---

---

## Peer evaluations

- Use form from text
  - Includes your impression of who
    - had hardest role (% sum to 100)
    - had the most work (% sum to 100)
  - You must use team member names and roles (unlike the form)
- Written text
  - Your evaluation of TSP
  - Your evaluation of how well you fulfilled your role
    - What did you do that worked well?
    - What did you do that did not work well?

---

---

---

---

---

---

---

---