



## Outline

- n Your questions
- n Upcoming milestone deliverables
  
- n Testing in the project lifecycle
- n Analyzing scenarios using inference diagrams
- n Mistakes to avoid



## Your Questions

- n On class?
- n On project?
- n On homework?
- n On material we've discussed?
- n Other?




## Next Milestone – Preliminary Release

**Deliverables:**

- n Application sources and binaries
  - n One-step build for all sources
- n Latest spec & design documents
  - n Keep it short! Consider the feedback I gave in the informal discussions about what is and isn't important for customers / devs
- n Release notes
  - n Detailed instructions on how to run a (small) demo of your app
  - n Known issues with prioritization
- n Automated (unit and acceptance) tests
- n Up-to-date schedule
  - n Including what has been done and what remains to be done

**Issues to consider:**


- n Who is your audience – customers or developers? What do they expect from a preliminary release? What defines success?



## In Contrast with Final Release Deliverables...

All of the above plus:

- n Separate distributions for customers and developers
- n Separate user and technical documentation
- n Latest test plan
- n Automated tests (unit, acceptance, etc.) that have wider coverage
- n Known issues with priorities, expressed using Bugzilla tasks/tickets
  - n Using some other professional bug tracking system is okay too
- n CVS & Bugzilla "snapshots"




## CSE403

### Section 6:

### Testing in the Project Lifecycle

**Bonus: Common Mistakes to Avoid**  
**Bonus: Influence Diagrams**

Valentin Razmov, CSE403, Sp'05

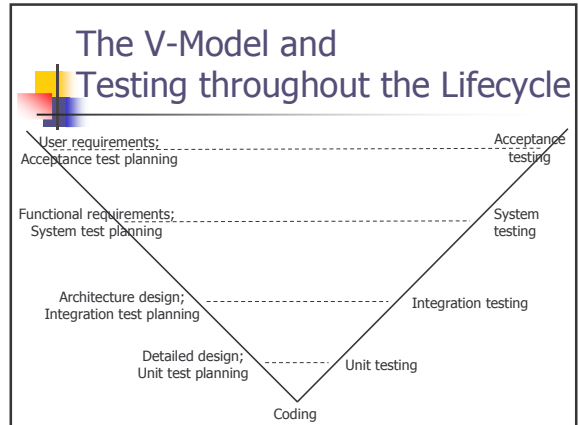


## Testing (recap)

- n Testing helps to establish if \_\_\_\_\_/ because \_\_\_\_\_.
  
- n Testing begins (when?) \_\_\_\_\_/ because \_\_\_\_\_.

## Testing (recap)

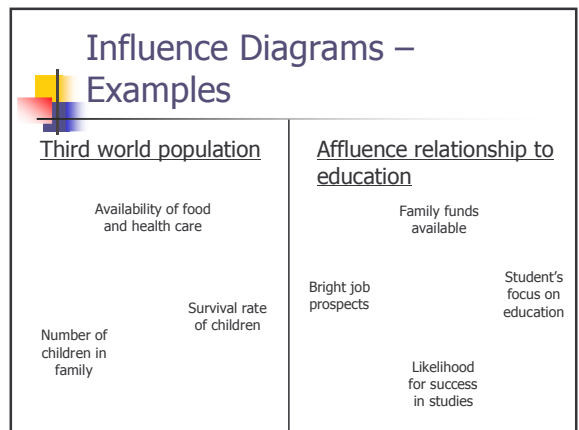
- How much does each dollar invested in early defect tracking save the project later on?
  - (A) \$0.25
  - (B) \$1.00
  - (C) \$7.00
  - (D) \$50.00
- Answer: \$3 - \$10
- What does that imply for how much emphasis you need to put on testing?



## Why Test?

- More stress you feel => less testing you do
  - "Testing takes precious time."
- Less testing => more bugs left in your code
- More bugs => more stress

How do we break this positive feedback loop?



## Test-Driven (a.k.a. Test-First) Development

### Why does it work?

- The sooner a defect is found, the cheaper it is.
- Less stressful programming experience
  - Not putting off the biggest unknown till last moment
- Predictable quality at all times. Product can ship on a short notice without stress.
- Increased quality of programmers' work; increased respect from others
- Customers have higher confidence in product.

## Mistakes Students in Previous SwEng Classes Have Made

### Testing-related:

- Not creating a testing framework from the start
  - Much harder to integrate later on
    - You'd be playing a catch-up game, which creates a significant disincentive and feels like wasting time.
  - Tests should be written at the same time that you write the code, ideally even before you write the code
    - Tests then serve as post-conditions
    - Psychologically advantageous
- No automated tests => it's hard to make any claim about code quality
  - You may have overlooked something.
  - You can't convince anyone else.
- Not having a version to play with (and test) until late
  - For games this is critical - you need time before the final release for hallway testing.

## Mistakes to Watch Out for *Now*

### Scheduling-related:

- Not leaving enough "safety net" time before major releases in case something unexpected happens
  - It often does happen in the most inopportune moment.
- Leaving too few resources (people) for a critical task that can't be delayed

### Communication-related:

- Failing to submit key required components (e.g., documentation, tests, etc.)
- Submitting code without clear instructions about how to run it if one starts from scratch
- Not having a backup person who knows how to put together deliverables and submit

## New Concepts

- One-step build
  - Very effective together with automated tests
  - Requires a "toolsmith", but can be simplified and done once or twice a day, starting it manually
- Code / module invariants
  - Repeated / regression testing checks if the invariants in the code still hold
- Test-first programming
  - Reduces stress and may increase code quality

## Favorite Related Quotes

- "Doing things right is not as important as doing the right things." (Drucker's Dictum)
- "Verification == Did we build the product right? Validation == Did we build the right product?" (Barry Boehm)
- "Doing things at the last minute is much more expensive than just before the last minute." (Randy Pausch)
- "If you haven't got time to do it right, you don't have time to do it wrong."
- "Good judgement comes from experience. Experience comes from bad judgement."
- "Failing to plan is planning to fail."
- "Work expands so as to fill the time available for its completion." (Parkinson's Law, 1957)

## One-minute Feedback

- What one or two ideas discussed today captured your attention and thinking the most?
- List any ideas / concepts that you would like to hear more about. Be specific.