Updated Schedule of Remaining Class-Related Deliverables

- **Fri, Aug 12 @ 10pm:** hw#4 responses due
- **Sun, Aug 14 @ 10pm:** final project release due
- **Mon, Aug 15, in class:** final project demos / presentations
- **Mon, Aug 15 @ 10pm:** peer review #3 due
- **Wed, Aug 17 @ 10pm:** hw#5 responses due; peer review #3 viewing and usefulness feedback due
- **Thu, Aug 18 @ 10pm:** final take-home exam due (online submission)
- **Fri, Aug 19, before class:** final take-home exam due (on paper)
- **Fri, Aug 19 @ 10pm:** final questionnaire due

Criteria for "Good Enough"

(James Bach, http://www.satisfice.com/articles.shtml)

- It has sufficient benefits.
- It has no critical problems.
- The benefits sufficiently outweigh the problems.
- In the present situation, and all things considered, further improvement would be more harmful than helpful.

- Is your product "good enough" now?

Criteria for "Good Enough"

(James Bach, http://www.satisfice.com/articles.shtml)

- It has sufficient benefits.
- It has no critical problems.
- The benefits sufficiently outweigh the problems.
- In the present situation, and all things considered, further improvement would be more harmful than helpful.

- Is your product "good enough" now?

- Key questions to ask when doing an evaluation:
  - Good enough for whom?
  - Good enough for what?

Lecture 15: Configuration Management, Software Maintenance, and Code Reviews (Part II)

Valentin Razmov

Outline

- Software maintenance
- Code reviews

- Bonus: Influence diagrams – a tool for analysis

Resources

- *Rapid Development*, by Steve McConnell
- *Code Complete*, by Steve McConnell
- *The Pragmatic Programmer*, by Andrew Hunt and David Thomas
- *Test Driven Development: By Example*, by Kent Beck

What is Software Maintenance?

- Producing new (versions of) software under the constraints of existing software
  - Backwards compatibility is often assumed / required
- Comprises all phases of the lifecycle, starting with requirements gathering
  - Yet another turn of the spiral

The "Double-Edged Sword" of Software Maintenance

- The hope is that you will get to the maintenance stage.
  - If your company has been selling successfully and there is demand for new versions
- Most developers hope that they won't have to deal with maintenance.
  - It's harder to maintain (someone else's) code than it is to write a new one.

In Reality…

- Maintenance is what you do most of the time.
- Maintenance is often an afterthought, which turns it into a nightmare.
  - You have to think ahead of time how you (or someone else) are going to maintain the code later.
  - Refactoring is crucial
    - … but it should be done regularly and must start early
- Maintenance is often given to junior developers.
  - "This way they'll learn the guts of the system better."
  - Shhht!!! – senior developers don't want to work on maintenance themselves.
  - Result: brittle code with little conceptual or design integrity; even more maintenance headaches to come.

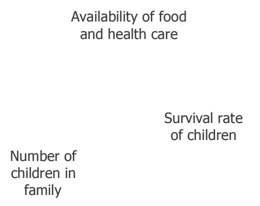
Influence Diagrams – Motivating Frequent Testing

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

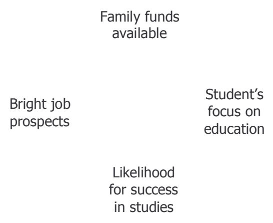
- How do we break this positive feedback loop?

Influence Diagrams – Examples from Other Domains

Third world population


Availability of food and health care
Survival rate of children
Number of children in family

Affluence relationship to education


Family funds available
Bright job prospects
Student's focus on education
Likelihood for success in studies

Influence Diagrams – Junior Devs Doing Maintenance

- Junior devs in maintenance => low code quality
- Low code quality => code is hard to maintain
- Hard to maintain code => those who have a choice prefer to avoid doing maintenance
- Fewer senior devs in maintenance => more junior devs in maintenance

- How do we break this positive feedback loop?

Code Reviews – What and Why?

- **What:** A practice whereby one needs to get a sign-off before committing changes / new code
- **Why:** (List two main reasons that you see.)

12 Aug 2005 CSE403, Summer'05, Lecture 15

Code Reviews – A Valuable Practice

- Ensures that more than one person has seen every piece of code
  - Prospect of someone else reviewing your code raises the quality threshold
  - Even if someone is absent, another person has a clue.
- Forces code writers to articulate decisions (and participate in the discovery of flaws)
- Allows junior personnel to get early hands-on experience without jeopardizing code quality
  - They get paired up with experienced developers who can answer subtle questions and provide valuable feedback

12 Aug 2005 CSE403, Summer'05, Lecture 15

Mechanics of Code Reviews

- Done before committing the code to the repository and incorporating it into a new build
- Review includes suggestions for improvement at a logical and/or structural level, to conform to an agreed set of quality standards.
  - Refactoring step after code review feedback, followed by a second code review
- Reviewer's job is to attest that code is maintainable and meets the quality standards
- Both code writer and reviewer are accountable for allowing the code to be committed.

12 Aug 2005 CSE403, Summer'05, Lecture 15

Code Review Tool Support

- Made easy by advanced tools that:
  - integrate with the configuration management system
  - highlight changes (i.e., *diff* function)
  - allow traversing back into history
- E.g.: Eclipse offers such support

12 Aug 2005 CSE403, Summer'05, Lecture 15

In Reality...

- Code reviews are a common practice.
- Code reviews can be perfunctory if:
  - they are not part of the (company) culture
    - "Let's just get it over with, because we've got important work to do."
  - they are done without proper tool support
    - "This is going to be a pain; let's pray it'll soon be over."

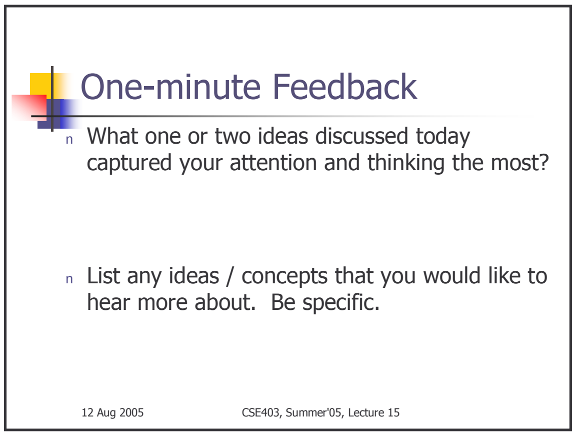
12 Aug 2005 CSE403, Summer'05, Lecture 15

Parting Thought…

"Don't do anything that you don't want to appear on the front page of the newspaper."

- Sound advice not only when it comes to writing code!

12 Aug 2005 CSE403, Summer'05, Lecture 15



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.