

# Beyond 6.170

Mark A. Herschberg

1  
3/20/02

# Who am I and why am I worth your time?

Speaker:

*Mark A. Herschberg '95*  
*hershey@alum.mit.edu*

2  
3/20/02

## Agenda

- Motivation- What did I want to know?
- Best ideas in software engineering
- The BIG POINT!

3  
3/20/02

## Design Patterns

Concept:

- Catalog of well known, defined, design techniques

4  
3/20/02

## Anti-Patterns

Concept:

- Catalog of well known, defined...  
BAD IDEAS!

Anti-Patterns includes:

- Symptoms (smell)
- Consequences
- Causes
- Solution

5  
3/20/02

## Refactoring

Definition:

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure. It is a disciplined way to clean up code that minimizes the changes of introducing bugs. In essence when you refactor you are improving the design of the code after it has been written.

From "Refactoring: Improving the Design of Existing Code" by Martin Fowler

Purpose:

To make the code easier to maintain.

6  
3/20/02

## Refactoring

- Why do we need it?
  - Code misalignment:
    - Code need changes
    - Reuse abuse
    - Time pressure
    - Hacks
    - Too smart for our own good
  - Entropy
  - Bugs/performance problems
- The most expensive part of software development is code maintenance.

7  
3/20/02

## Refactoring

### Steps in refactoring:

- Identifying a structural change is Needed
- Designing the solution
- Change what it is, into what I should be, with *many, many, very tiny steps*

8  
3/20/02

## Refactoring

- Isn't this just code cleanup?
  - Yes and No: It's structured cleanup
  - Refactoring formalized cleanup and cataloged methods

9  
3/20/02

## Code Reviews

- Formal process of peer review
- Benefits:
  - Makes you a better coder
    - When reviewing
    - When being reviewed
  - Keeps everyone on the same page
- Code Reviews are a must!!!

10  
3/20/02

## Quality Assurance

### Quality Assurance

- Blackbox
- Whitebox
- Unit
- Integration
- System

95% of you won't see the brick wall until you hit it. 60% of you still won't see it.

11  
3/20/02

## Quality Assurance

- Where its been, and where it stands
- Necessary, and not so evil
- Appreciate QA, use QA, understand QA
- QA is everyone's responsibility

12  
3/20/02

## Quality Assurance

Key to reducing your bug count:

- Have a QA process not an Easter egg hunt.
- When a bug is found ask:
  - Ask: Where and why the bug was introduced?
  - Ask: Why was it not found earlier?
  - Then fix the process to not let it happen again!

This method is used on mission critical projects (e.g. NASA)

13  
3/20/02

## Documentation

How much writing will you do as a software engineer?

14  
3/20/02

## Documentation

A word cloud containing the following terms: Use Cases, Requirements, Specifications, Code Comments, Presentations, GUI design, Subsystem Design Doc, Architecture Document, Support Procedures, Integration Developers Guide, Status Reports, Administrator's Guide, Coding Guidelines, Project Plan, Schedule, Task List, Personnel List, Business Plan, Meeting Minutes, Process, Performance Reviews, Manual, Beta Plan, Storyboard, Agendas, Installation Guide, and Repairs.

15  
3/20/02

## Documentation (Planning)

Documents produced prior to coding

- Business Plan
- Test Plan
- GUI Designs
- Storyboard
- Use Cases
- Integration Doc
- Specifications
- Requirements
- Architecture Document
- Subsystem Design Doc

16  
3/20/02

## Documentation (Company)

Company documents relevant to a particular engineering project:

- Process Guidelines
- Coding Guidelines
- Support Procedure
- Beta Plan

Any of these may be project specific instead

17  
3/20/02

## Documentation (Project)

Non engineering project documents:

- Agenda
- Schedule
- Task List
- Status Report
- Presentation
- Project Plan
- Meeting Minutes
- Performance Review
- Personnel List

18  
3/20/02

## Documentation (Code)

Documentation relating to code:

- Code comments (all types)
- Bug Report
- Metrics Report

Doc early, doc often.

*Code is not complete until it is documented!*

19  
3/20/02

## Documentation (External)

Documentation delivered with code:

- Readme
- Installation Guide
- User Manual
- Administrator's Guide
- Developer's Guide

20  
3/20/02

## Communication

Communication is the key to software development!

- Written
- Oral
- Formal
- Informal
- Engineers
- Business folk
- Sales & Marketing
- VC
- Legal

Communication must be documented or it will be forgotten!  
(Ex: telephone game)

21  
3/20/02

## Review

- 5 very important topics
  - Patterns / Anti-Patterns
  - Refactoring
  - Code Reviews
  - Quality Assurance
  - Documentation

- I didn't learn a single one at MIT!

22  
3/20/02

## Misaligned Courses

Applicable Classes

Basis:

- 6.042J
- 6.046J

Applicable:

- 6.170
- 6.033
- 6ThG

Grounding Classes

- 6.001
- 6.002
- 6.003
- 6.004
- 18.03
- 6.033
- 6.034
- header
- elective
- elective
- elective

23  
3/20/02

## Education

- College didn't teach you everything. In fact, it hardly taught you anything!
- At best, college gave you the fundamental CS background necessary to understand how computer models and solutions work.
- How to make models and solutions yourself, is an entirely different story....

- *Continue to learn!*

24  
3/20/02

## Education

### Where to learn:

- Books
  - 2 a year min
  - Reading groups
- Magazines
- Conferences
- Training Programs
- Newsgroups / Web sites
- Peers

25  
3/20/02

## Summary

- What was the point?
  - There's a wealth of useful information out there, not yet in your curriculum
  - You're learning has hardly begun
- Where to go from here?
  - Make a plan to learn
  - Find jobs which help you learn

26  
3/20/02

## Q&A

- Questions and feedback (good and bad) welcome
- Contact info
  - Mark A. Herschberg
  - [hershey@alum.mit.edu](mailto:herschey@alum.mit.edu)
- Slides will be on the 6.170 web site

27  
3/20/02