Objectives

- Software Quality Metrics
- Static Analysis Tools
- Refactoring for Extensibility

Review

- What principle did we focus on last class?
- What is the typical fix for designing more flexible/ maintainable code?

SOFTWARE QUALITY METRICS

Metrics to Measure Software Quality

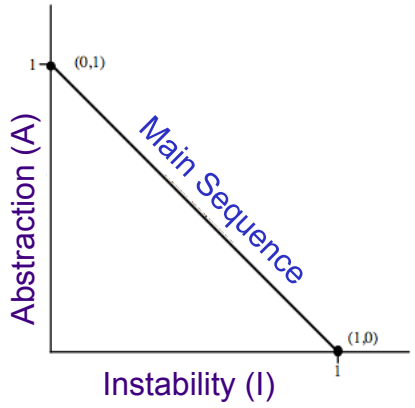
- Create metrics to help us figure out if our code is good and what we can improve
  - Add a little more science
- Examples: number of methods, # loc /method, # attributes/class
- Tricky: Not clear what is "good" number
  - Requires good judgment, experience
  - Metrics often should not be considered in isolation

Example Metrics

Metric	Description
Afferent Coupling (Ca)	Number of classes outside package that depend upon classes within package
Efferent Coupling (Ce)	Number of classes inside package that depend on classes outside package
Instability (I)	Ce / (Ca + Ce) → range [0,1] Indicates resilience to change
Abstractness (A)	Number of abstract classes divided by total number of classes in a package. 0→concrete, 1→abstract

Instability: How does this metric measure instability? What does a 0 or 1 mean?

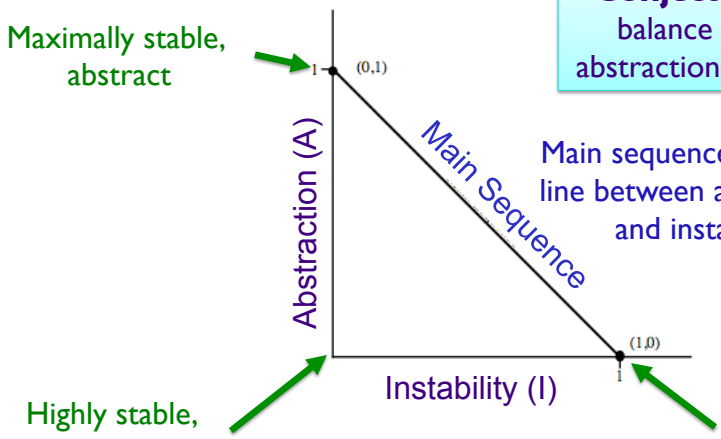
Main Sequence: Supports OCP



 [Martin 1994]

Main Sequence: Supports OCP

Conjecture: Want balance between abstraction and stability



Give an example of something in each category

 [Martin 1994]

Example: Lack of Cohesion of Methods (LCOM)

- A measure of a class's *cohesiveness*
- Calculated with the Henderson-Sellers method:
  - m(A) : # of methods accessing an attribute A
  - Calculate the average of m(A) for all attributes, subtract the number of methods m, and divide the result by (1-m)

$$\frac{\overline{m(A)} - \#\text{ of methods}}{1 - \#\text{ of methods}}$$

Analysis and Discussion: What does LCOM tell us?

$$\text{LCOM} = \frac{\overline{m(A)} - \#\text{ of methods}}{1 - \#\text{ of methods}}$$

m(A) is # methods accessing attribute A

- What is the relationship between m(A) and # of methods?
- What are the extremes?
  - Every method accesses every attribute?
  - Every attribute is accessed by one method?

Example: Lack of Cohesion of Methods (LCOM)

- A measure of a class's *cohesiveness*
- Calculated with the Henderson-Sellers method:
  - m(A): # of methods accessing an attribute A
  - calculate the average of m(A) for all attributes, subtract the number of methods m and divide the result by (1-m)
- Low value → a cohesive class
- Value close to 1 → a lack of cohesion
  - Suggests class might better be split into a number of (child) classes

Plugins

- Tons of available plug-ins
- Some better than others
  - Haven't been updated for awhile
  - Aren't highly ranked
- Documentation is often lacking
  - Look for new plugin in various menus, including Properties option, Views, Perspectives
- Try out new plugins
  - if you don't like it, you can uninstall

Eclipse Metrics Plugin

- Provides information about your classes
  - # of classes
  - # of lines of code per method
  - # of attributes
  - Coupling (afferent, efferent)
  - Instability
  - …
- Update site:
  - `http://metrics2.sourceforge.net`

FindBugs

http://findbugs.sourceforge.net/

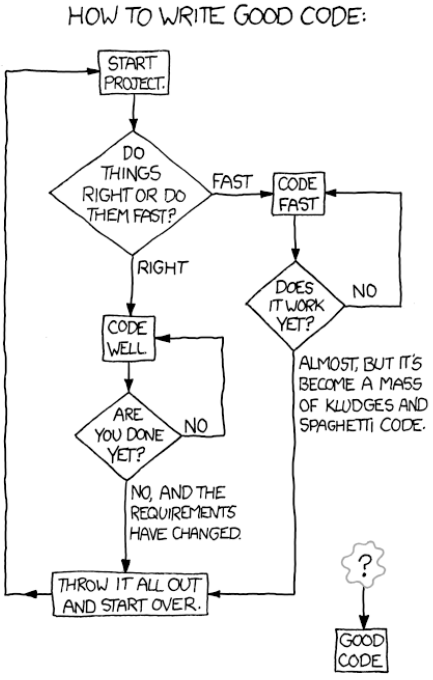
- Performs static analysis to look for bugs
  - Looks for "bug patterns"
- Types of errors
  - Correctness
  - Bad practice
  - "Dodgy" → style
- Eclipse Marketplace
  - Under `Help` menu
  - Search for FindBugs

Checkstyle Plugin

- Checks that Java code adheres to a set of coding standards
- Examples:
  - Identifies variables that should be declared final
  - Identifies code that is not open to extension and how to handle
  - Will report issues with tabbing/spacing
- Install through Eclipse Marketplace

PMD Reports

- Java source code analyzer
- Looks for possible bugs, poor coding practices
  - Duplicate code
  - Dead code
  - Empty if/while/catch blocks
  - Suboptimal code (e.g., Strings, StringBuffers)
- Install using Help→Install New Software → Add Update Site
  - PMD - http://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/



So what does this mean for us?

Good-Enough Design Discussion

Perfect Design

- ✓ Follows all design principles
  - OCP, Single Responsibility, no code smells, …
- – May not be possible
  - Infinite refactoring, development
- – Code never released

Good-enough Design

- – Not everyone agrees on design
- – Maintenance requires changes to a few places
- ✓ Code gets released to customers

Similar tradeoffs in testing

Roulette Discussion

- What do you like about the code?

Roulette Discussion

- Focus: how **open** is the code to adding **new** kinds of **bets** and how **closed** it is to **modification**?
  - How many classes know about the `Bet` class?
  - What code would need to be added to `Game` to allow the user to make another kind of bet that paid one to one odds and was based on whether the number spun was high (between 19 and 36) or low (between 1 and 18)?

Exam Next Wednesday

- Java Wrapup
  - Files
  - Streams
  - Jar files
  - classpath
  - Comparing Java and Python
- Software development
  - Software development models
  - Testing
    - Various types, techniques
  - Version control
  - Design principles
  - Code smells
  - Refactoring
  - GUIs