CS314 Software Engineering
Maintainable Software

Dave Matthews

Software Quality Characteristics
Software Maintenance Activities

- Corrective
  - bugs discovered and fixed
- Adaptive
  - system is adapted to changes in operating environment
- Perfective
  - new or changed requirements from users or stakeholders
- Preventive
  - increase quality or prevent future bugs

Maintainability Guidelines

1. Write short units of code
2. Write simple units of code
3. Write code once
4. Keep unit interfaces small
5. Separate concerns in modules
6. Couple architecture components loosely
7. Keep architecture components balanced
8. Keep your codebase small
9. Automate development pipeline and tests
10. Write clean code
Write short units of code

• Small units are easy to understand, test, and reuse.
• Limit the length of new code units, refactor existing code units.
• BCH guidelines
  – 7% > 60 lines of code
  – 22% > 30 lines of code
  – 44% > 15 lines of code

Write simple units

• Simple units make it easier to modify and test
• Limit the number of branch points (McCabe)
• Split complex units into simpler ones.
• BCH guidelines
  – 1.5% with McCabe > 25
  – 10.0% with McCabe > 10
  – 25.0% with McCabe > 5
Write Code Once

- Duplicated code requires changes need to be made in more than one place (bugs, enhancements)
- Write reusable, generic code or call methods
- Refactor to extract methods or superclasses
- BCH guidelines
  - 95% code is not redundant

Keep interfaces small

- Fewer parameters makes interfaces easier to understand, modify, and reuse
- Limit the number of parameters
- Extract parameters into objects
- BCH guidelines
  - 1% > 7 parameters
  - 3% > 4 parameters
  - 14% > 2 parameters
Separate concerns in modules

- Changes to loosely coupled code are easier to perform
- Avoid large modules to achieve loose coupling
- Assign responsibilities to separate modules, hide implementation details behind interfaces
- BCH guidelines
  - 7% with module/class fan-in >50
  - 14% with module/class fan-in >20
  - 22% with module/class fan-in >10

Couple architecture components loosely

- Independent components eases isolated maintenance allows separate maintenance, and eases testing
- Achieve a loose coupling between top-level components
- Minimize relative amount of public interfaces
- BCH guidelines
  - <14% of code resides in modules with external/public interfaces
Keep architecture components balanced

- Balanced components ease locating code and allow for isolated maintenance.
- Balance by the number and relative size of top-level components.
- Several components of approximately the same size.
- BCH guidelines
  - number of top-level system components near 9
  - adjusted Gini (inequality of distribution) coefficient < 0.71

Keep your codebase small

- Small products, projects, and teams are more successful.
- Keep your codebase small.
- Avoid codebase growth and actively reduce system size by not duplicating code, refactoring existing code, using third party libraries/frameworks, splitting up large systems into multiple smaller systems (decoupling)
- BCH guidelines
  - < 20 person-years of effort (about 175,000 lines of Java)
Automate tests

• Automated testing makes development predictable and less risky.
• Write automated tests using a test framework and run them regularly.
• BCH guidelines
  – aggregates unit complexity, component independence, and system volume

Write clean code

• Clean code is maintainable code
• Leave no trace
  – no unit level smells (size, complexity, interfaces)
  – no bad comments
  – no code in comments
  – no dead code
  – no long identifier names
  – no magic constants
  – no poorly handled exceptions