

**SE 2AA4, CS 2ME3 (Introduction to Software
Development)**

Winter 2017

5 Software Engineering Principles (Ch. 3)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

January 16, 2017



Software Engineering Principles

- Administrative details
- Unix command of the day
- MEASURE example
- Measuring software qualities
- Software engineering knowledge units
- Software engineering principles
- Key principles
 - ▶ Rigour
 - ▶ Formality
 - ▶ Separation of concerns
 - ▶ Modularity
 - ▶ Abstraction
 - ▶ Anticipation of change
 - ▶ Generality
 - ▶ Incrementality

Administrative Details

- Notetaker needed for 2AA4
- Assignment 1
 - ▶ Final version now in repo
 - ▶ Files due by midnight January 28 (changed)
 - ▶ E-mail partner files by January 28 (changed)
 - ▶ Lab report due February 2
 - ▶ Using Python 2.7, doxygen, make, LaTeX, git (changed)
- Questions on assignment?
 - ▶ You may have to make assumptions if you find the description ambiguous
 - ▶ Feel free to incorporate robustness, but include doxygen comments to explain what you are doing
 - ▶ Do not add to the methods exported by the module
 - ▶ Do not add additional arguments to the method calls
 - ▶ Constructor versus Selector (Accessor) versus Mutator?

Administrative Details Continued

- Next week's tutorial will cover LaTeX and Assignment 1
- Following week will cover git and Assignment 1
- Strongly suggest installing VirtualBox (or equivalent) with a Linux VM
- Avenue discussion now open
 - ▶ More efficient than e-mail
 - ▶ Using this for the participation grade

Participation Grade

- Based on Avenue discussion
 - ▶ Read other posts
 - ▶ Create topics
 - ▶ Reply to questions

Unix Command of the Day

- `which`
- Returns the pathname of the file which would be executed in the current environment had its argument been typed at the command prompt
- `which` returns something like `/usr/bin/which`
- `test` returns something like `/bin/test`
- If you want to know your search path, type `echo $PATH`
- `which` is useful if you have more than one version of Python

MEASURE

- Web page for users
- Issue Tracking for Development
- Issue Tracking for Users

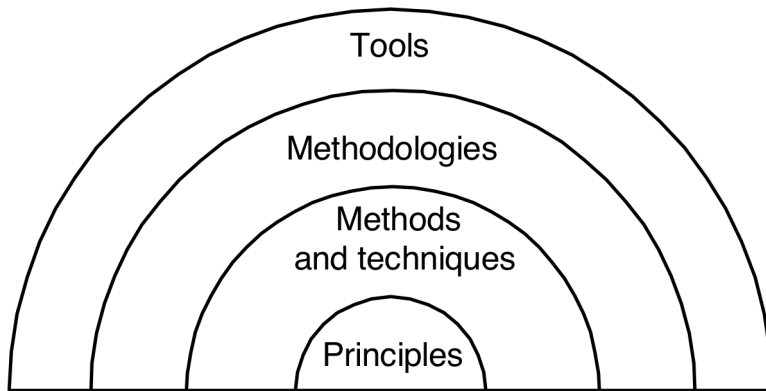
Measurement of Quality

- A software quality is only important if it can be measured
 - without measurement there is no basis for claiming improvement
- A software quality must be precisely defined before it can be measured
- Most software qualities do not have universally accepted
- Can you directly measure maintainability?
- How might you measure maintainability?
- Chapter 6 has some ideas for measuring software quality

Software Engineering Knowledge Units

- A **principle** is a general concept that is widely applicable in software engineering
- **Methods** are:
 - ▶ General guidelines to govern the execution of an activity
 - ▶ Rigorous, systematic and disciplined approaches
 - ▶ Example - following a template
- **Techniques**
 - ▶ Provide an approach to develop software, but they are more technical and mechanical than methods
 - ▶ Techniques have a more restricted applicability than methods
 - ▶ Example Hoar triple for correctness proof
- A **methodology** is a coherent collection of methods and techniques
- A **tool** is a device that supports the application of a method, technique, or methodology

Visual Representation



Tools

- What are some examples of tools for software development?

Software Engineering Principles

- Used to reduce complexity
- Form the basis for methods, techniques, methodologies and tools
- Can be used in all phases of software development
- Can be applied to both **process** and **product**
- The purpose of the principles is to improve quality, with a special emphasis on reliability and evolvability
- All of the key software engineering principles are also key principles of **mathematics** and **engineering as a whole**!
- Software engineering is often more explicit in identifying and using principles than in other branches of engineering

Key Principles

1. Rigour
2. Formality
3. Separation of concerns
4. Modularity
5. Abstraction
6. Anticipation of change
7. Generality
8. Incrementality

Rigour

- An argument is **valid** if the conclusion is a logical consequence of the premise
- **Rigour** is precise reasoning characterized by
 - ▶ Only unambiguous language is used
 - ▶ There are no hidden assumptions
 - ▶ Care is taken to ensure that all arguments are valid
- Rigour is achieved through the use of **mathematics** and **logic**
- Rigour should be systematically employed throughout the whole software development process

Formality

- **Formality** is reasoning in a **formal system** consisting of
 - ▶ A **language** with a **formal syntax** and a **precise semantics**
 - ▶ A set of **syntactic rules**
- A formal system enables reasoning to be **mechanized**
 - ▶ Reasoning is performed mechanically with computer assistance
 - ▶ Arguments are machine checked
 - ▶ Parts of the reasoning are automated
- The use of formality in software development has a high cost
 - ▶ The learning curve is very high
 - ▶ Tool support and knowledge bases are inadequate
 - ▶ The amount of detail involved is often overwhelming
- Nevertheless, **formality is the promise of the future!**

Formality versus Rigour

What are the advantages of formality over rigour? What are the disadvantages?

Formality

Every software development project uses at least one formal language. Is this statement True or False?

- A. True
- B. False

Separation of Concerns

- Separation of concerns is the principle that different concerns should be isolated and considered separately
 - ▶ The goal is to reduce a complex problem to a set of simpler problems
 - ▶ Enables parallelization of effort
- Concerns can be separated various ways
 - ▶ Different concerns are considered at different times
 - ▶ Software qualities are considered separately
 - ▶ A software system is considered from different views
 - ▶ Parts of a software system are considered separately
- Dangers
 - ▶ Opportunities for global optimizations may be lost
 - ▶ Some issues cannot be safely isolated (e.g. security)

Separation of Concerns

What are examples of separation of concerns in traditional engineering