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

Winter 2018

05 Software Engineering Principles (Ch. 3)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

January 16, 2018

05 Software Engineering Principles (Ch. 3)

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

Administrative Details

- Assignment 1
  - Part 1: January 22, 2018
  - Partner Files: January 28, 2018
  - Part 2: January 31, 2018
  - Document any assumptions or exceptions
  - Specification for external interface (as seen by other programs)
- Questions on assignment?
- This week's tutorials
  - LaTeX
  - A1 Sample
- Consider installing VirtualBox (or equivalent) with a Linux VM

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

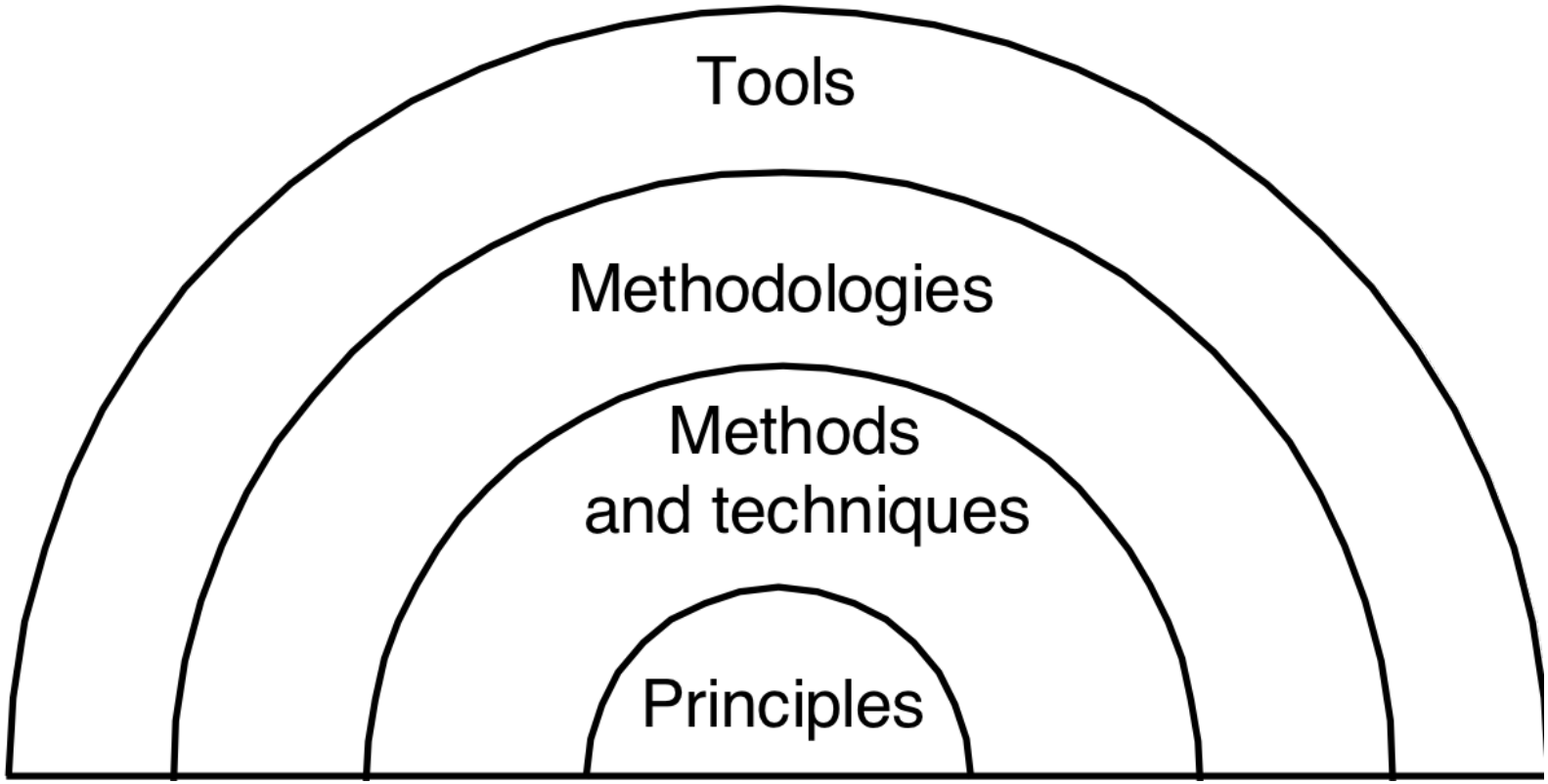
MEASURE (for Measuring Learning Outcomes)

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

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

Formality

The A1 assignment specification is formal. Is this statement True or False?

A. True

B. False

Formality

Your first year mathematics textbook is formal presentation of Calculus. Is this statement True or False?

A. True

B. False

Formal Versus Rigorous

Formal Version of Calculus "Textbook"

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

Separation of Concerns

What are examples of separation of concerns in traditional engineering?

What are examples of separation of concerns in software engineering?

Separation of Concerns: SE Examples

- Separation of requirements from design
- Separation of design from implementation
- Decomposition of a system into a set of modules
- The distinction between a module's interface and its implementation
- The distinction between syntax and semantics

Modularity

- A modular system is a complex system that is divided into smaller parts called modules
- Modularity enables the principle of separation of concerns to be applied in two ways:
  1. Different parts of the system are considered separately
  2. The parts of the system are considered separately from their composition
- Modular decomposition is the top-down process of dividing a system into modules
- Modular decomposition is a "divide and conquer" approach
- Modular composition is the bottom-up process of building a system out of modules
- Modular composition is an "interchangeable parts" approach

Examples of Modularity

What are examples of modularity in traditional engineering?

Properties of Good Modules

- To achieve the benefits of modularity, a software engineer must design modules with two properties
  1. High cohesion: The components of the module are closely related
  2. Low coupling: The module does not strongly depend on other modules
- This allows the modules to be treated in two ways:
  1. As a set of interchangeable parts
  2. As individuals

Zero Coupling?

Given that low coupling is desirable, the ideal modularization has zero coupling. Is this statement True or False?

A. True

B. False

Proposed Modularization for a Car

Suppose you decide to modularize the description of a car by considering the car as comprising small cubes 15 inches on a side.

1. Is the cohesion high or low?
2. Is the coupling high or low?
3. Propose a better modularization
4. In general, how should you decompose a complex system into modules?