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

Winter 2018

06 Software Engineering Principles Continued (Ch. 3)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

January 16, 2018

06 Software Engineering Principles Continued (Ch. 3)

- Administrative details
- 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
- Questions on assignment?

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?

Abstraction

- Abstraction is the process of focusing on what is important while ignoring what is irrelevant
- Abstraction is a special case of separation of concerns
- Abstraction produces a model of an entity in which the irrelevant details of the entity are left out
  - Many different models of the same entity can be produced by abstraction
  - Abstraction models differ from each other by what is considered important and what is considered irrelevant
  - Repeated application of abstraction produces a hierarchy of models
- Refinement is the opposite of abstraction
- Over abstraction produces models that are difficult to understand because they are missing so many details

Abstract Data Type

What makes an Abstract Data Type Abstract?

Abstract versus Concrete

What is a more abstract way to state the following?

```
def bubbleSort(a):
    n = len(a)
    for i in range(n - 1):
        for j in range(n - 1 - i):
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
```

Anticipation of Change

- Anticipation of change is the principle that future change should be anticipated and planned for
- Also called design for change
- Techniques for dealing with change:
  1. Configuration management: Manage the configuration of the software so that it can be easily modified as the software evolves
  2. Information hiding: Hide the things that are likely to change inside of modules
  3. Little languages: Create little languages that can be used to solve families of related problems
- Since software is constantly changing, anticipation of change is crucial for the software development process

Anticipation of Change

Change should be anticipated for the development process, as well as the product. For instance, what can you do to anticipate changes in staffing?

Likely Changes

You have been asked to design a software version of the game Battleship. What are some likely changes?

Generality

- The principle of generality is to solve a more general problem than the problem at hand whenever possible
- Advantages
  - The more general a solution is the more likely that it can be reused
  - Sometimes a general problem is easier to solve than a specific problem
- Disadvantages
  - A general solution may be less efficient than a more specific solution
  - A general problem may cost more to solve than a more specific problem
- Abstraction is often used to extract a general solution from a specific solution

Generality for Computational Geometry

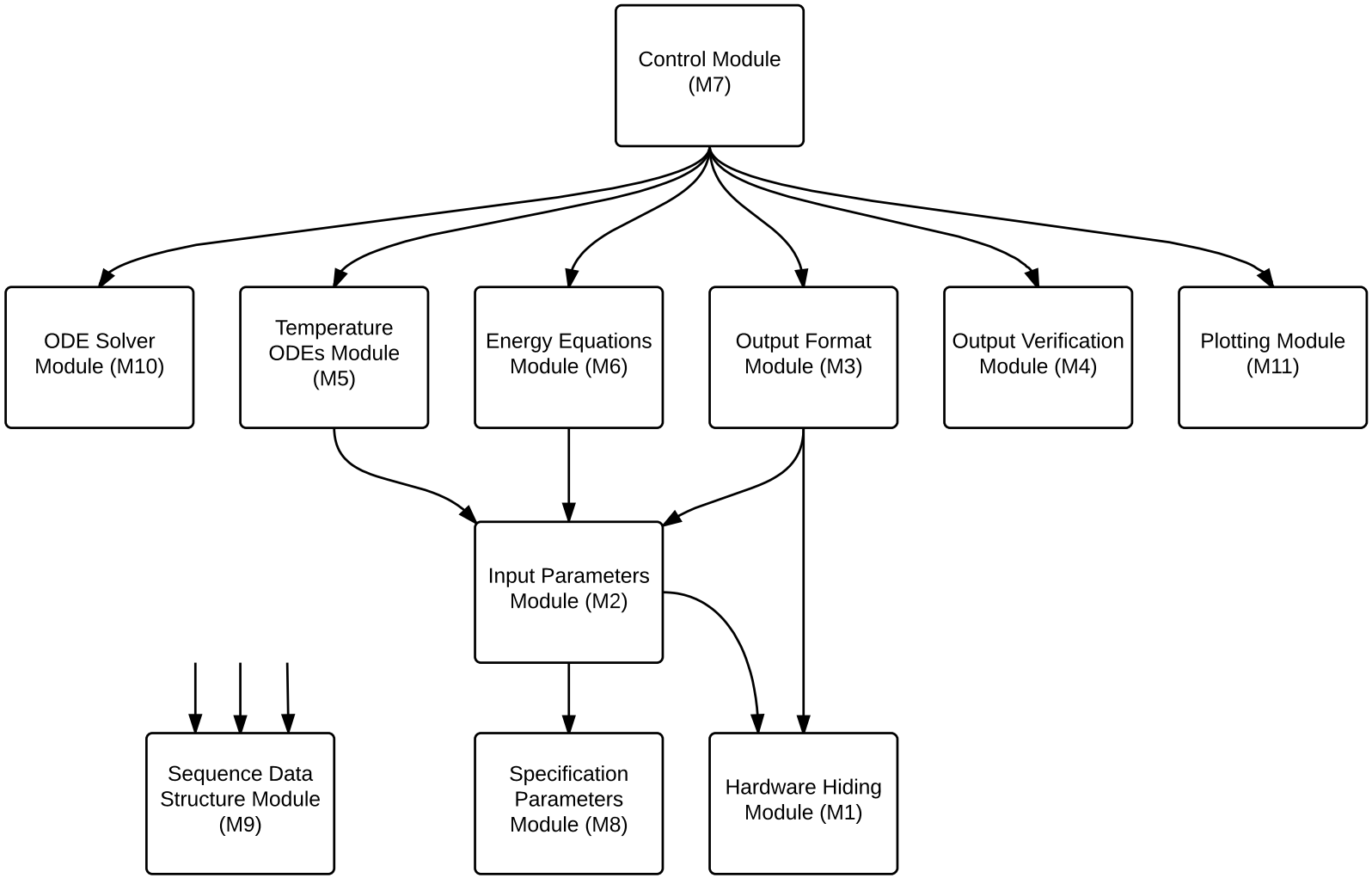
The n-dimensional volume of a Euclidean ball of radius R in n-dimensional Euclidean space

$$V_{2k}(R) = \frac{\pi^k}{k!} R^{2k}$$

$$V_{2k+1}(R) = \frac{2(k!)(4\pi)^k}{(2k+1)!} R^{2k+1}$$

- See Wikipedia page for Volume of an n-ball
- CGAL includes specific and general kernels
- Domain Specific Languages (DSLs) hold the promise of generality and performance

Generality of ODE Solver



Incrementality

- The principle of incrementality is to attack a problem by producing successively closer approximations to a solution
- Enables the development process to receive feedback and the requirements to be adjusted accordingly
- Techniques for developing software incrementally
  1. Rapid prototyping: Produce a prototype that is "thrown away" later
  2. Refinement: A high-level artifact (like a requirements specification or a higher-level design) is incrementally refined into a low-level artifact (like a lower-level design or an implementation)

Principles for High Quality Documentation

- Under quality, we should have mentioned these
- To achieve external qualities for documentation, there are some generally agreed on internal qualities
- Internal qualities can more likely be directly measured
- Following list of qualities based on IEEE guidelines for requirements (IEEE Std 830-1998)
  - Complete
  - Consistent
  - Modifiable
  - Traceable
  - Unambiguous
  - Correct
  - Verifiable
  - Abstract

Up Next

- Design (Chapter 4)
  - More on modularization
  - Mathematics for Module Interface Specification (MIS)
  - MIS
  - Abstract objects
  - Abstract Data Types (ADTs)
  - Generic ADTs
  - Object Oriented Design (OOD)