

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

Winter 2017

37 Review for Final DRAFT

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 19, 2017



Review for Final Exam

- Administrative details
- Design patterns
- Topics on the exam
- Structure of the exam
- Advice on exam preparation
 - ▶ Time management before the exam
 - ▶ Time management during the exam
 - ▶ How to study
- Questions? Feedback? Comments?

Administrative Details

- Today's slides are partially based on slides by Dr. Wassying and on van Vliet (2000)

Model View Controller (MVC)

- Separate computational elements from I/O elements
- Three components
 1. Model encapsulates the system's data as well as the operations on the data
 2. View displays the data from the model components, possibly multiple view components
 3. Controller handles input actions
- The controller may or may not depend on the state of the model
- The controller depends on model state when menu items are enabled or disabled depending on the state of the model

Design Pattern Properties

- A pattern addresses a recurring design problem that arises in specific design situations and presents a solution to it
- A pattern must balance a set of opposing forces
- Patterns document existing, well-proven design experience
- Patterns identify and specify abstractions above the level of single components (modules)
- Patterns provide a common vocabulary and understanding for design principles
- Patterns are a means of documentation
- Patterns support the construction of software with defined properties, including non-functional requirements, such as flexibility and maintainability

Describing Patterns

- Context: the situation giving rise to a design pattern
- Problem: a recurring problem arising in that situation, and
- Solution: a proven solution to that problem

The Proxy Pattern (from van Vliet (2000))

- Context: A client needs services from another component. Though direct access is possible, this may not be the best approach
- Problem: We do not want to hard-code access to a component into a client. Sometimes, such direct access is inefficient; in other cases it may be unsafe. This inefficiency or insecurity is to be handled by additional control mechanisms, which should be kept separate from both the client and the component to which it needs access.
- Solution: The client communicates with a representative rather than the component itself. This representative, the [proxy](#), also does and pre- and postprocessing that is needed.

Command Processor Pattern

- Context: User interfaces which must be flexible or provide functionality that goes beyond the direct handling of user functions. Examples are undo facilities or logging functions
- Problem: We want a well-structured solution for mapping an interface to the internal functionality of a system. All 'extras' which have to do with the way user commands are input, additional commands such as undo and redo, and any non-application-specific processing of user commands, such as logging, should be kept separate from the interface to the internal functionality.

Command Processor Pattern Continued

- Solution: A separate component, the **command processor**, takes care of all commands. The command processor component schedules the execution of commands, stores them for later undo, logs them for later analysis, and so on. The actual execution of the command is delegated to a supplier component within the application.

Topics on the Final Exam

- All of them
- From “introduction to software engineering” to “design patterns”
- Greater emphasis on the material since the midterm, such as
 - ▶ Specification
 - ▶ Verification

Types of Questions

- Short answer
- Complete the specification
- Write small bits of code (C, OCaml or Java)
- Design a module
- List the test cases for coverage
- Build the control flow graph
- ...

Time Management

- Time management before the exam
 - ▶ Make a schedule
 - ▶ Optimize the reward for spending your time and energy
 - ▶ Work smarter not harder
 - ▶ Schedule time for rest
- Time management during the exam
 - ▶ Divide time proportional to value of question
 - ▶ Start with what you know best
 - ▶ Leave nothing blank

How to Study?

- Better if an active, rather than a passive, process
- Do questions
 - ▶ From midterm, assignments
 - ▶ From the textbook
 - ▶ From other books
 - ▶ Make up your own
 - ▶ MIS for an ADT that you have studied
 - ▶ MIS for 2C03 assignments
 - ▶ Questions from last year's final
 - ▶ Post questions to WebCT

Questions?

- Software quality?
- Software principles?
- Module decomposition?
- MIS?
- Parnas tables?
- Fault seeding
- White box testing?
- Analysis?
- ...