Modeling

Chapter 3: Class diagrams: the essential
Chapter 4: Sequence diagrams

Appendix 1: an introduction to UML

Chapter 7.1.3 object class identification
Chapter 7.1.4 design models
Administrative stuff

• Quiz
• Warmup project due this Wednesday
• Project proposal due this Friday
Course Project Grading

• Milestone 1
  • Team members get the same grade

• Group performance
  • 75%

• Individual performance
  • Commit log
  • Self-evaluation + peer-evaluation
    • After milestone 3.b (for performance in milestone 2, 3.a, 3.b)
    • After milestone 5 (for performance in milestone 4.a, 4.b, 5)
Outline

• Review
  • What are requirements, Why?
  • Use case diagram
  • Use case description
  • Activity diagram

• Today: design modeling
  • OO
  • Class diagram
  • Sequence diagram
What are requirements?
What are requirements?

• The services the software should provide
+ • The constraints the software should follow

• Functional requirements
+ • Non-functional requirements
What is requirement engineering (RE)?

• The process of
  • Finding out
  • Analyzing
  • Documenting
  • Checking
these desired services and constraints
is Requirement Engineering
Who will read requirement document & why?
Who will read requirement document & why?

• Users/customers
• Design team
• Developers
• Testing team
What to put in a requirement document?
What are the requirements?

- Functional requirement
- Non-functional requirement
What are the requirements?

- Functional requirement
  - Specify functionality
  - Input, output, ...

- Non-functional requirement (how to measure them quantitatively?)
  - Performance
    - Time complexity, space complexity, scalability, throughput, latency, space
  - Security
  - Usability
  - Power & energy
  - Legal, ethical
  - Dependability
    - Security
    - Availability = available time / (service available time + service down time)
    - Reliability = how likely the service will go town at time T
Non-functional requirements

- Efficiency requirements
  - Usability requirements
  - Performance requirements
- Reliability requirements
- Portability requirements
- Interoperability requirements
- Ethical requirements
- Delivery requirements
- Implementation requirements
- Standards requirements
- Legislative requirements
- Safety requirements
- Privacy requirements
Non-functional requirements

• Try to use quantitatively measurable metrics to describe them
• Examples
Requirement document format

• IEEE standard
  • http://en.wikipedia.org/wiki/Software_requirements_specification
Where do ... diagrams fit in req. engineering?

• Help find out and describe requirements
  • Use case diagram
    • Use case description
    • Activity diagram
Use case diagrams

• A diagram includes
  • Actors
  • Use cases
  • System boundary

Use case text

• Use case name
• Use case priority
• Main scenario
  • Steps
• Extensions
  • Steps
Activity diagram

• Start, end
• Flow
• Fork/join
• Decision
• ...
Design

OO

Class Diagram
Sequence Diagram
What is the first P.L. you learned?
Object-Oriented Programming, Classes

• Class
  • Data + Operation

• Encapsulation
• Polymorphism
• Inheritance

• Enhance modularity!
Encapsulation

• “the packing of data and functions into a single component. The features of encapsulation are supported using classes. It allows selective hiding of properties and methods in a class by building an impenetrable wall to protect the code from accidental corruption.”
Encapsulation

• “the packing of data and functions into a single component. The features of encapsulation are supported using classes. It allows selective hiding of properties and methods in a class by building an impenetrable wall to protect the code from accidental corruption.”

• Implication to design?
Polymorphism

• “to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes”

• “the provision of a single interface to entities of different types.”

• Examples
Polymorphism

• “to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes”

• “the provision of a single interface to entities of different types.”

• Implication to design?
• Benefits?
• Problems?
Inheritance

• “a mechanism for code reuse and to allow independent extensions of the original software via public classes and interfaces.”

• Examples
Inheritance

• “a mechanism for code reuse and to allow independent extensions of the original software via public classes and interfaces.”

• Implication to design?
• Benefits?
• Problems?
Class diagram

• Describes the types of objects in the system
• Describes the static relationships among them

http://en.wikipedia.org/wiki/Class_diagram
How to decide/design classes?

• Data+operation
Components of class diagrams

• Class name
• Class properties
  • Attributes
  • Associations (could be bi-directional)

  visibility name : type [multiplicity] = default {property-string}
• Class operations
  Visibility name (parameter list) : return-type {property-string}
• Generalization
  • Inheritance (subclass, super class, interface, ...)
• Dependency  _ _ _ _ _ →
• Constraints {}
• * represents unknown number of CSClass property objects of a student object
• If we put a constant number, like 4, here, we should replace the “Set” data structure into Array
UndergraduateStudent and GraduateStudent are subclasses of Student, and inherit all the attributes and methods of Student. They both re-implement the registerClass function (polymorphism), and both inherit the superclass' implementation of displaySchedule.
How to turn class diagram to code

• A private attribute \( \rightarrow \) ??
• A * attribute/association \( \rightarrow \) ??
• Class declaration
  • Some attributes may not map to fields
Advanced Class-Diagram Features

• Composition ≠ vs. Aggregation
  • Belong to relationship
  • Composition: single owner, disappear with the owner

• Abstract class

• Template class

*We didn’t talk about this in lecture, so this will not appear in quiz/exam*
What are the constraints to set?

• Assertion
  • Pre-condition
  • Post-condition
  • Invariant

We didn’t talk about this in lecture, so this will not appear in quiz/exam
Sequence diagram

• Describes how objects collaborate/interact with each other in one scenario
Components of sequence diagram

• Participants
• Life-line
• Activation bar
• Message
  • Regular calls, self calls

• Creating and deleting object
• Loops and conditionals
  • loop, alt, opt

http://en.wikipedia.org/wiki/Sequence_diagram
Summary

• Requirement specification template
• Use-case diagram
• Activity diagram
• Class diagram
• Sequence diagram