

# Project Management Course Book

T++ Technical Skills Training Program

CUNY Institute for Software Design & Development (CISDD)

New York Software Industry Association (NYSIA)

Aron Trauring

Class 7 October 26th, 2004

# Class Agenda

**1:30-2:45** Use Cases (Presentation)

**2:45-3:00** Break

**3:00-3:45** Exercise 7 — Use Cases

**3:45-4:00** Break

**4:15-4:45** Break-out Presentation

**4:45-5:30** UML (Presentation)

## Use Case Presentation

### Use Cases

- Requirements and Analysis
- Story telling is oldest tool of human communication
- Use cases are little stories about the system
- Describe a set of scenarios, that is interactions, between the user and a system, related to a specific user goal
- Forms the basis of discussion between customer and design teams
- Stick to structured prose
- Use cases don't give me the GUI

### Alistair Cockburn - 'Writing Effective Use Cases'

- Powerpoint Presentation: <http://alistair.cockburn.us/crystal/talks/ucitap/usecasesintheorya.ppt>
- Powerpoint Presentation on FourM: <http://www.fourm.info/TPPPM/Presentations/UML/usecasesintheoryandpractice180.ppt>
- Use Case Template: <http://alistair.cockburn.us/usecases/uctempla.htm>
- Sample Requirements document: <http://alistair.cockburn.us/crystal/articles/srd/systemrequirementsdocument.html>

**See Alistair Cockburn presentation.**

**See Alistair Cockburn Use Case Template.**

**See Alistair Cockburn sample Requirements document.**

## Exercise 7 — Use Case

1. Choose two of the following projects and develop a set of use cases for it. Be sure to identify the overall scope and goal and the stake-holders. Refer to the Cockburn documents for help.

- (1) Getting your child to school on the first day of classes
- (2) Constructing a successful first date
- (3) Receiving a promotion
- (4) Developing a successful family vacation
- (5) Convincing your boss to adopt more agile project management methods

2. Someone different presents to the class.

# Unified Modeling Language (UML)

## Historical Background

- Structured Analysis gurus picked up ideas of OO especially with DOD backing of ADA in 1983
- Notational and methods wars raged for fifteen years
- 1997 Object Management Group issued first UML standard
- Standardized notation for modeling software systems \* A tool for improving communications
- UML is a notation not a method\
- Be agile viz. a barely sufficient use of UML
- Strong CASE is almost always overkill — death to the Waterfall!
- Sample Diagrams: <http://www.visualuml.com/Sample%20Diagrams.htm>

## Class Diagrams

- Analysis and Design
- The type of objects in the system and the static relationships between them
- Subtypes — 'Books are a type of Product'
- No one to one mapping between functional decomposition and classes
- Sub-functions may belong in different classes
- Use cases tell the domain modeler which aspects of the domain are interesting

## Three levels of Class Diagrams

- Conceptual — objects in real world — domain model
- Architectural — high-level system components.
- Implementational — lowest level classes
- Package — group of architectural classes — may form component for team or iteration

## Associations

- Relationship between instances of classes — customer takes orders
- Multiplicity — customer may make several orders over time (\*), order comes from one customer (1)
- Association end may have a role name — sales rep
- Navigability — direction of relationship

**Other Aspects of Class Diagrams**

- Attributes — tight associations
- Operations — processes class carries out
- Generalizations — subtypes — a corporate customer is a subtype of customer
- Constraint rules

**Interaction Diagrams**

- How groups of objects collaborate in some behavior
- Two kinds of Interaction Diagrams

**Sequence Diagrams**

- Life-line — object's life during the interaction
- Condition — when a message is sent ([needsReorder])
- Iteration Marker — message sent many times
- Asynchronous Message
- Object Deletion

**Collaboration Diagrams**

- Sequence shown by numbering message
- Simpler layout

**Packages**

- Groups of classes
- Can model in class and interaction diagrams

**State Diagrams**

- Modeling reactive systems
- Fully defined semantics allows for execution
- Event driven — Event[Guard]/Action
- Or / And
- Super-states

**Activity Diagrams**

- Work-flow diagram
- Business process modeling
- Parallel processing
- Activity — state of doing something — singing, processing invoice
- Sequencing of activity, with conditional and parallel behavior
- Fork/Join — parallel processing — different from flowcharts
- Sub-activities
- Swim-lanes — activities by responsibilities

**Physical Diagrams**

- Deployment Diagrams — physical relationships between hardware and software in delivered system
- Component Diagram — physical code packages and their dependencies