Last week

Business proposal

Project Estimation: size, effort, schedule
- size: LOC, FP
- effort: man-months
- schedule: months

Scheduling

WinWord 1.0

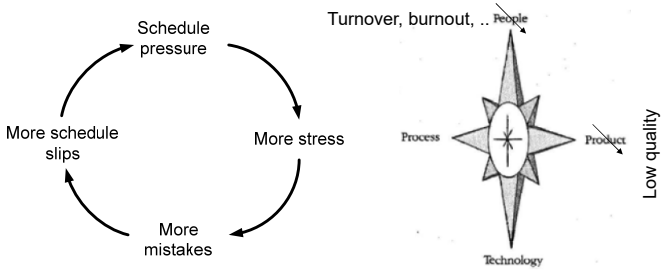
- Microsoft Word for Windows 1.0
  - Example of overly optimistic schedule
  - 5 years in development, 660 man-months, 249,000 lines of code
    - check nominal schedule tables → 26 months, 800 man-months
  - The final 5-year schedule was approximately five times as long as originally planned.

Report Date	Estimated Ship Date	Estimated Days to Ship	Actual Days to Ship	Relative Error
Sep-84	Sep-85	365	1887*	81%
Jun-85	Jul-86	395	1614	76%
Jan-86	Nov-86	304	1400	78%
Jun-86	May-87	334	1245	73%
Jan-87	Dec-87	334	1035	68%
Jun-87	Feb-88	245	884	72%
Jan-88	Jun-88	152	670	77%
Jun-88	Oct-88	122	518	76%
Aug-88	Jan-89	153	457	67%
Oct-88	Feb-89	123	396	69%
Jan-89	May-89	120	304	61%
Jun-89	Sep-89	92	153	40%
Jul-89	Oct-89	92	123	25%
Aug-89	Nov-89	92	92	0%
Nov-89	Nov-89	0	0	0%

Source: RD

Scheduling Problems

- Developers underestimate their projects by 20 – 30%.
- Average small-project estimate is off by > 100%.

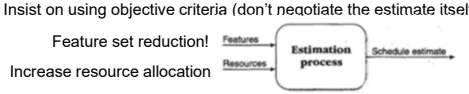


Silver bullet syndrome

40% of software errors are due to schedule pressure

Schedule Pressure

- Cause
  - wishful thinking by customers, managers, …
  - Little awareness of the software estimation methods
  - Poor negotiating skills
    - 75% developers are introverts (where only 33% of general population are)
    - Managers tend to be 10 years older and negotiate for a living
    - Developers oppose negotiating tricks (e.g. high initial estimates)
- Resolution
  - Principled Negotiation
    - Separate people from the problem (cooperate, explore options)
    - Focus on interests, not positions (find underlying needs)
    - Find options for mutual gain (phasing, fewer features, add resources)
    - Insist on using objective criteria (don't negotiate the estimate itself)



5

Feature Set Control

- Early project: feature set reduction
  - Minimal specification
  - Requirement scrubbing
  - Versioned development
- Mid-project: feature creep control
  - Setup a change-control board (to review/accept/reject changes)
  - Versioned development
  - Sort development cycles
- Late project: feature cuts
  - Eliminate low priority features
- Keep in mind:
  - 50% cut in project size yields a 75% reduction in resources and ~50% reduction in schedule

6

Your project status

- Software concept
- Requirement analysis
- Life-cycle planning & technology selection
- Architectural Design
- Detailed Design
- Coding & Testing
- System testing

7

Lecture 5

Virtualization

UML

8

Outline

- Server Virtualization
- UML

9

Virtualization

10

What is it?

- Server virtualization virtualizes servers
  - Virtual computers inside a real/host computer
  - One physical server machine (**host**) may host several virtualized servers (**guest**)
- Virtualizes computer hardware so that it can run on supported physical hardware
  - In business world, virtualization technology was adopted by enterprises to consolidate their servers to avoid server under-utilization.
  - Today, it is used for cloud-computing service providers to satisfy demands from various customers of different needs.

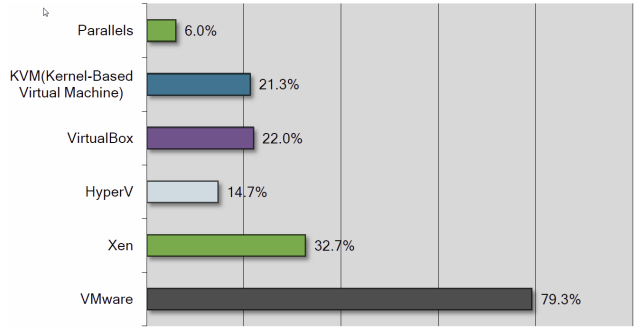
11

How ?

- Through virtualization software (Windows). They are called hypervisor or virtual machine manager …
  - VMWare (http://www.vmware.com)
  - Oracle VirtualBox (http://www.virtualbox.org)
  - Microsoft HyperV (http://www.microsoft.com/en-us/server-cloud/windows-server/server-virtualization.aspx)
  - Many others …

12

Virtualization Market Share



13

http://www.zenoss.com/in/virtualization_survey.html

Demonstration using VirtualBox

(needed for your next assignment)

15

Why ?

- Enterprises
  - Better server utilization
- Developers
  - Develop application for different platform without multiple computers or multi-boot.
- Regular users
  - Run applications written for old operating systems
- Virtualized machines make great testing environments ..
  - Clean (clean room)
  - Easy to restore, deploy, replicate
  - Test dangerous software (e.g. virus)

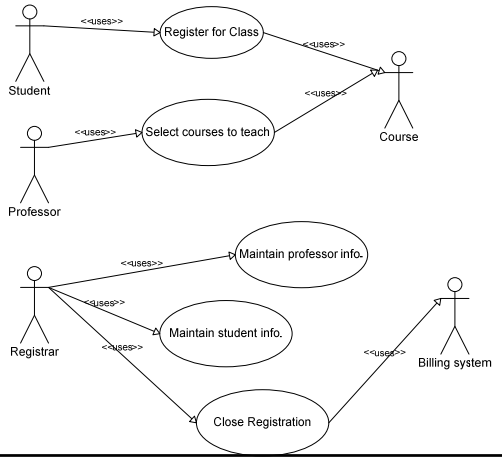
14

Introduction to UML

What is UML ?

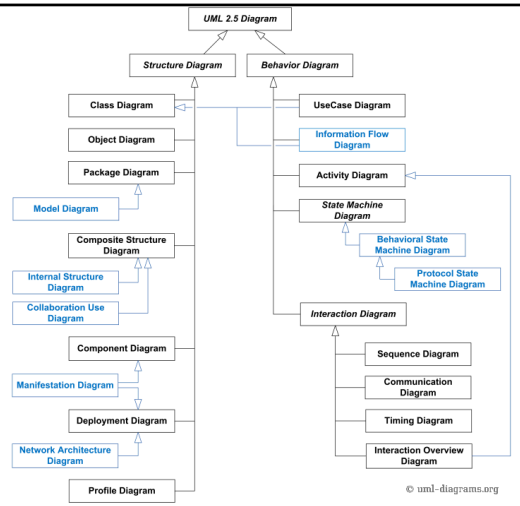
- UML: Unified Modeling Language
- Unified Modeling Language (UML) is a standardized (ISO/IEC 19501:2005), general-purpose modeling language in the field of software engineering. [Wikipedia]
- A standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
- UML uses mostly graphical notations.
- Current version: 2.5
- Complete spec can be freely downloaded at www.uml.org

Example: University Course Registration System



Reasons using UML

- UML is a technique to manage the complexity of object-oriented systems as they increase in scope and scale.
- Programming language independent
- Easier communication
- Supports iterative development (i.e., spiral model)
  - Supports both high level requirements/design in early spirals and detailed requirements/design later
- It helps users and software engineers to communicate about software requirements.
- Provides a blueprint for programmers to implement.



http://www.uml-diagrams.org/uml-25-diagrams.html

UML Modeling Tools

- There are several free and commercial software for creating UML diagrams
  - Microsoft Visio
    - https://products.office.com/en-us/Visio/flowchart-software
  - Rational Rose Modeler (now with IBM)
    - http://www-03.ibm.com/software/products/en/rosemod
  - Poseidon for UML
    - http://www.gentleware.com/new-poseidon-for-uml-8-0.html
    - The user guide is a good reference !

21

Diagrams to be introduced

- Use case diagram
- Class diagram
- Deployment diagram
- Sequence diagram
- Communication diagram
- Activity diagram
- State diagram

22

Use Case Diagram

23

Use Case Diagram

- Describes the requirements of the system, especially functional requirements.
- Use case diagram contains
  - Actors and their descriptions
  - Relationships between different entities
    - <<Include>>: implying that the behavior of the included use case is inserted into the behavior of the including use case
    - <<Extend>>: the behavior of the extension use case may be inserted in the extended use case under some conditions
    - {constraint}: indicate constraints

24

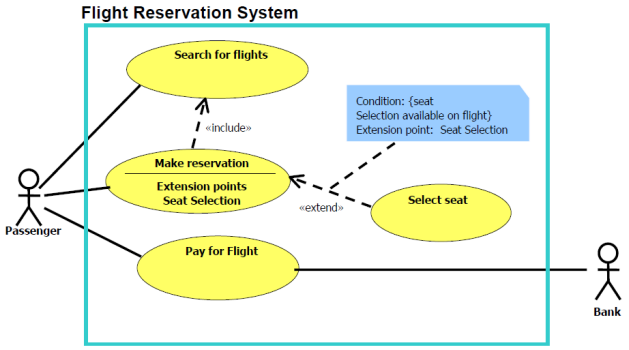
Elements of Use Case Diagram

An actor represents anything that interacts with the system.
- Actors are not part of the system, they represent roles a user of the system can play.
- An actor can actively interchange information with the system.
- An actor can be a passive recipient of information.
- An actor can be a giver of information.
- An actor can represent a human, a machine or another system.

A use case (instance) defines a sequence of actions a system performs that yields a result of observable value to an actor.
- A use case models a dialogue between an actor and the system.
- A use case is initiated by an actor to invoke a certain functionality in the system.
- A use case is a complete and meaningful flow of events.
- Taken together, all use cases constitute all possible ways of using the system.

25

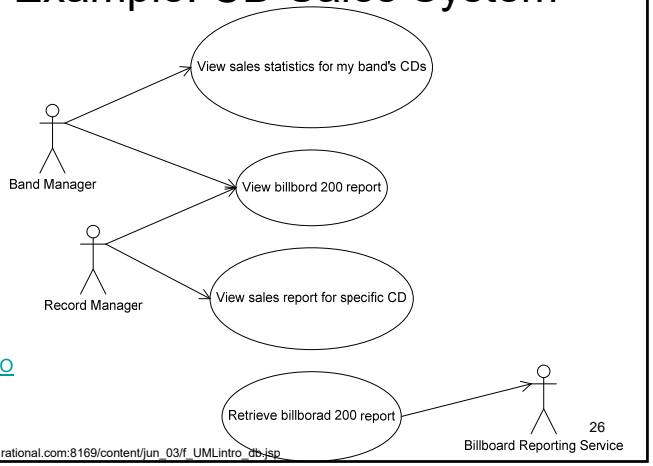
Sample Use-Case Diagram



27

Source: Introduction to UML 2.0, T. Quatrani, IBM

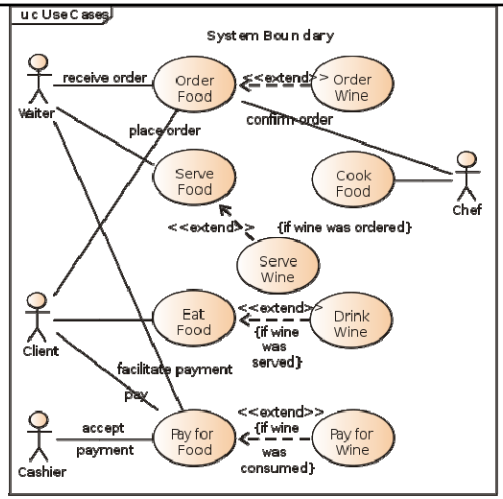
Example: CD Sales System



DEMO

26

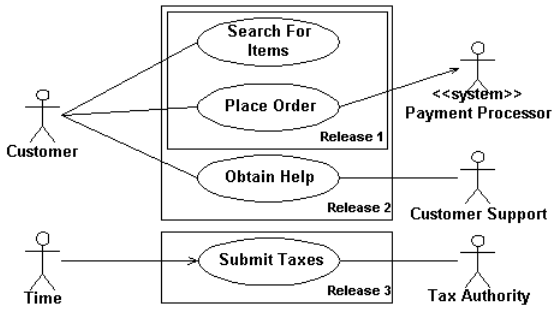
http://bronze.rational.com:8169/content/jun_03/f_UMLintro_db.jsp



28

http://en.wikipedia.org/wiki/Use_case_diagram

Sample Use Case Diagram

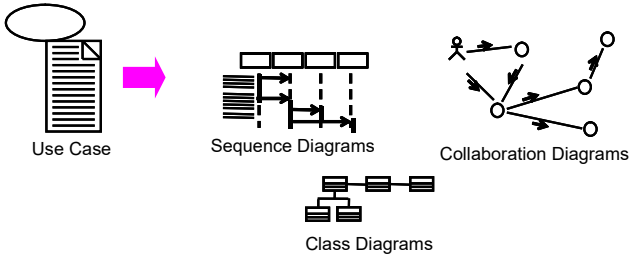


Use-Case Diagram with Staged Delivery!

Source: http://www.agiledata.org/essays/objectOrientation101.html

Class Diagram

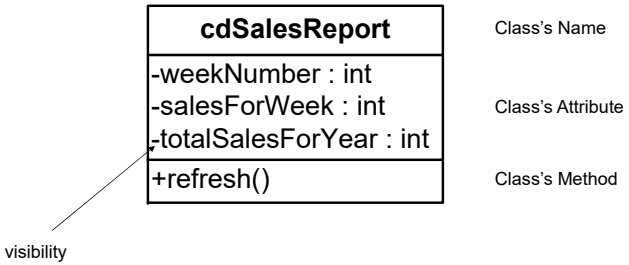
- Once the use case diagram is completed (in other words, the requirement specification is done), analysis and design take places.



Class Diagram

- Used in requirements, design and implementation:
  - Conceptual, to represent general entities in system
  - Specification, where we specify what each entity (class) will do (but not how)
    - List the methods/actions
  - Implementation
    - Detailed class diagram of actual software (Java or C++)
- List attributes, same as data model (to be introduced)
- List methods/operations/functions
  - Activities **naturally** associated with the data in the entity
- We often don't model everything—too hard to read
  - Focus on key parts of system

A class element

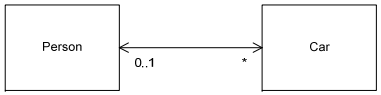


Relationships

Construct	Description	Syntax
Association	A relationship between two or more classifiers that involves connections among their instances.	knows uses refers
Aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	has references to
Generalization	A taxonomic relationship between a r and a more specific element	Is a kind of
Dependency	A relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element)	
Realization	A relationship between a specificati implementation	implements
Composition	If a class cannot exist by itself, and instead must be a member of another class, then that class has a Composition relationship with the containing class. A Composition relationship is indicated by a line with a filled diamond.	has owns

Association

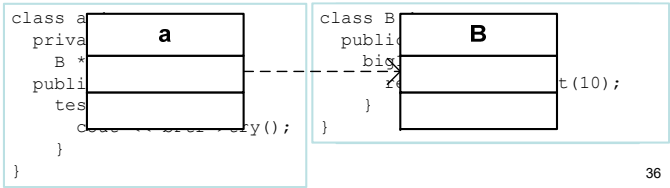
- Associations describe relationships between classes
  - Solid lines are used to describe associations
    - Multiplicity
    - Navigability: Unidirectional vs. bidirectional



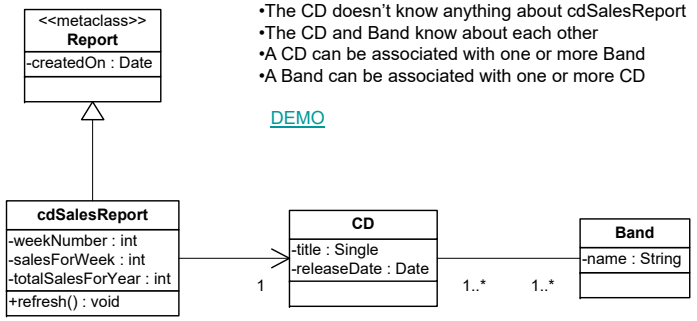
•A person may own 0 to many cars
•A car may have 0 or 1 owner

Dependency

- A dependency exists between two elements if changes to the definition of one element (the supplier) may cause changes to the other (the client).
  - one class sends a message to another;
  - one class has another as part of its data;
  - one class mentions another as a parameter to an operation.



Example

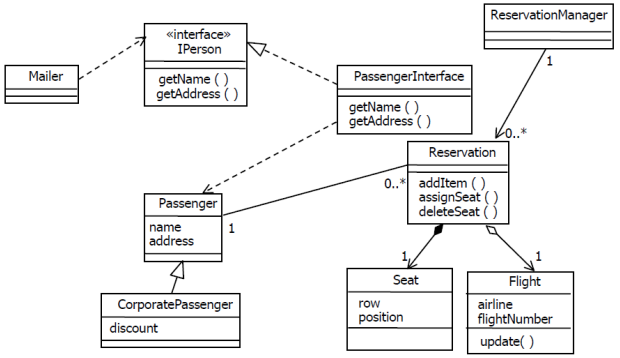


- The cdSalesReport inherits from the Report class
- A cdSalesReport is associated with one CD
- The CD doesn't know anything about cdSalesReport
- The CD and Band know about each other
- A CD can be associated with one or more Band
- A Band can be associated with one or more CD

DEMO

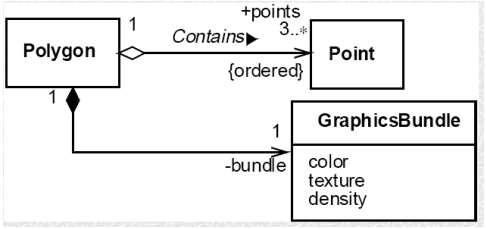
http://bronze.rational.com:8169/content/jun_03/f_UMLintro_db.jsp

Sample Class Diagram



Source: Introduction to UML 2.0, T. Quatrani, IBM

Example



- A polygon contains 3 or more points
- 3 or more points are aggregated in a polygon
- One polygon has one GraphicsBundle, and one GraphicsBundle belongs to exactly one polygon
- Without polygon, graphicsBundle cannot exist by itself.

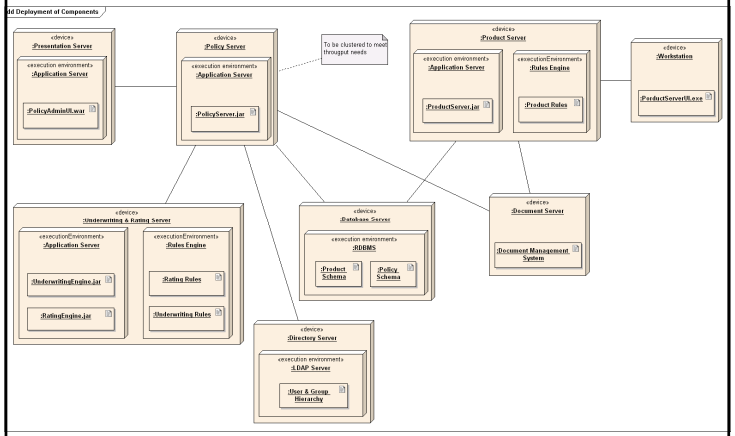
C. Kobryn (1999), "Object Modeling with UML: Introduction to UML"

Deployment Diagram

Deployment Diagrams

- Deployment diagrams show the physical relationship between hardware and software in a system.
- The solid lines represent communication links or dependencies.

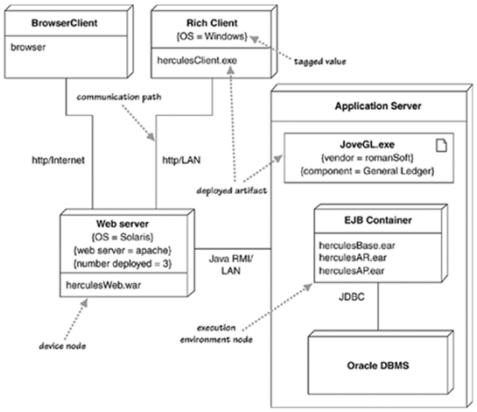
41



43

http://upload.wikimedia.org/wikipedia/commons/b/b9/Deployment_Diagram.PNG

Example



42

Reference: UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition, by Martin Fowler (2003)

Sequence Diagram

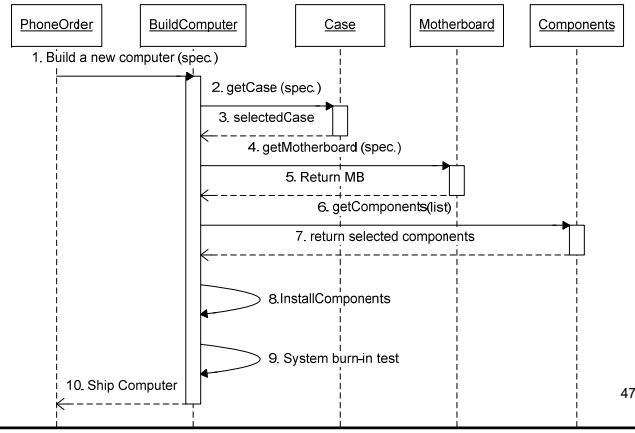
44

Sequence Diagram

- Shows a detailed or partial flow for a specific use case
- Shows the calls between different objects in their sequence
- 2-D:
  - Vertical: time/order
  - Horizontal: object instances

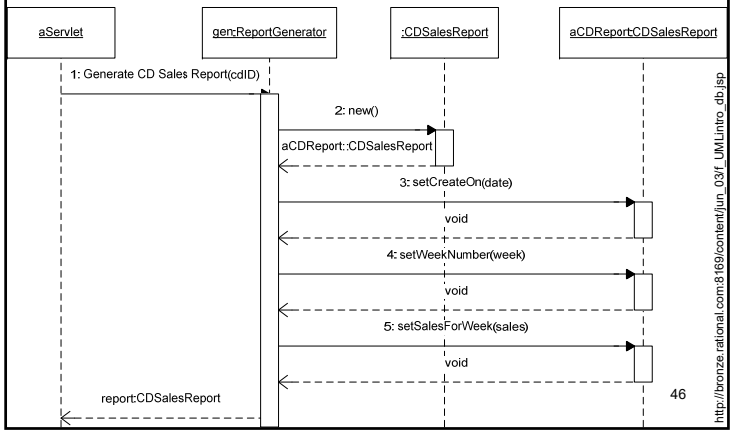
45

Example



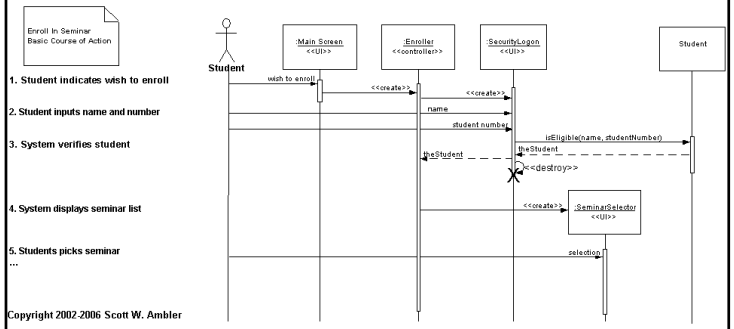
47

Example



46

Example



48

http://www.agiledata.org/essays/objectOrientation101.html