

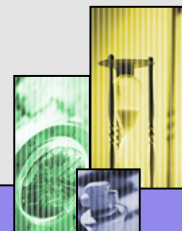
CIS*3430



Week One
Sept 13 & 15,
2005

Instructor Info

Name: Mark Wineberg
Email: wineberg@cis.uoguelph.ca
(use 'CIS*3430' at the beginning of the subject heading)
Office location: Reynolds 214
Office hours: Tues,Th 2:30-3:45
Teaching assistants: Michael de Ridder
Zijian Yan



Grading

Group Marks (40% of final grade)

- 5 labs @ 2% each = 10%
- Milestone 1 = 10%
- Milestone 2 = 10%
- Milestone 3 = 10%

Individual marks (60% of final grade)

- Java review quiz 2%
- Mini Assignments 4@2% = 8%
- Code Evaluation Assignment 5%
- Midterm Exam 20%
- Final Exam 25%

You must obtain 30 or more marks in the individual marks portion in order to pass the class

Text and Policies

Object-Oriented Analysis and Design, John Deacon,
Addison-Wesley, 2005

(sample chapters available..johndeacon.net)

- Come to class, I'm not posting notes as a rule
- Seminars are required (specific ones)
- Lates are not accepted
- Cheating/Plagiarism not tolerated
 - Collaborators.xml

Collaborators.xml

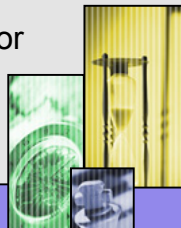
Course Topics

- Software Development Methods
 - Models and modelling
 - Subject Matter Model
 - Object Type Model
 - Technical Model
 - Techniques for Software Analysis and Design
 - Use Cases
 - UML
 - CRC cards
 - Prototyping
 - Requirements
 - Analysis
 - Design
 - Software Testing
 - Professionalism and Ethics
 - Maintenance, Installation and Configuration Management

Description	Date Available	Date Due (5pm unless otherwise noted)
Java Quiz	Sept 12	Sept 19
Mini Assignment 1	Sept 19	Sept 23
Mini Assignment 2	Sept 26	Sept 30
Teamwork Seminar	In Seminar (Sept 26_	At end of Seminar
Mini assignment 3	Oct 3	Oct 7
Requirements Gathering	In Seminar (Oct 3)	At end of Seminar
Use Case Seminar	In Seminar (Oct 10)	At end of Seminar
Mini Assignment 4	Oct 10	October 14
Project Milestone 1	Sept 19	October 21
Midterm	October 25	October 25 (in class)
CRC Card Seminar	In Seminar (Oct 24)	At end of Seminar
Paper Prototype Seminar	In Seminar (Oct 31)	At end of Seminar
Project Milestone 2	October 24	November 11
Code Evaluation Assign	November 14	November 25
Project Presentation Seminar	In seminar (Nov 28)	At end of Seminar
Project Milestone 3	November 14	December 2

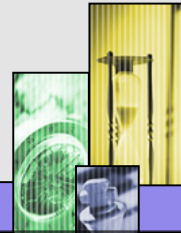
Group Project

- Imagine that you are in the employ of a company that creates electronic, multiplayer games.
- build a suite of games that are computerized reproductions of classic board games (the sort that many people play as children).
- Scrabble
- Risk
- Clue
- "Siamese" Chess (also called Partners Chess or Bughouse Chess)
- Chinese Checkers



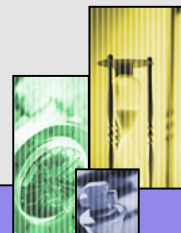
Forming Groups

- 5-6 members
- All must attend the same seminar
 - Doesn't matter which seminar though
- Groups may not cross class sections
- Submit form with no more than 6 names to Instructor by end of second week
 - I will 'flesh out' groups that are too small



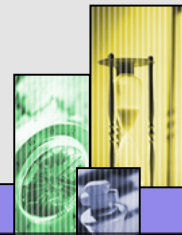
Announcements

- JUnit seminar
- No class seminars this week



Software Engineering

- The people who design software
- Repeatable
- Reliable
- Maintainable
- Programmers are the people who implement the designs
- Stats from:
 - Ontario Job Futures
 - Software Human Resource Council



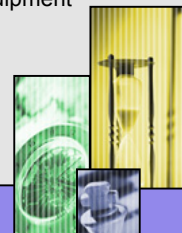
Software Engineer (ON)

Common Job Titles

Application Architect
Computer Software Engineer
Embedded Software Engineer
Software Architect
Software Design Engineer
Software Design Verification Engineer
Software Designer
Software Engineer
Software Testing Engineer
Systems Integration Engineer - Software
Technical Architect - Software
Telecommunications Software Engineer

Typical Employers

computer systems design and related services
finance and insurance companies
public administration
wholesale trade
scientific research and development services
communications equipment
manufacturing
software publishers
computer and peripheral equipment manufacturing



Programmer

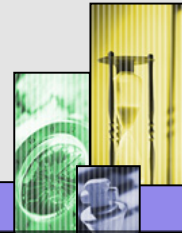
Application Programmer	computer systems design and related services
Business Application Programmer	finance and insurance
Computer Game Developer	companies
Computer Programmer	public administration
E-Commerce (Electronic Commerce)	software publishers
Software Developer	
Interactive Media Developer	
Multimedia Developer	
Operating Systems Programmer	
Programmer Analyst	
Scientific Programmer	
Software Developer	
Software Programmer	
Systems Programmer	
Web Programmer	

Duties (SE)

- Collect and document user's requirements to develop logical and physical specifications
- Research, evaluate and synthesize technical information to design, develop and test computer-based systems;
- Develop data, process and network models to optimize architecture
- Evaluate the performance and reliability of designs;
- Plan, design and co-ordinate the development, installation, integration and operation of computer-based systems;
- Assess, troubleshoot, document, upgrade and develop maintenance procedures for operating systems, communications environments and applications software;
- May lead and co-ordinate teams of information systems professionals in the development of software and integrated information systems, process control software and other embedded software control systems.

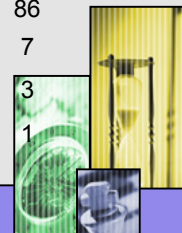
Duties (Programmer)

- Write, modify, integrate and test software code;
- Maintain existing computer programs by making modifications as required;
- Identify and communicate technical problems, processes and solutions;
- Prepare reports, manuals and other documentation on the status, operation and maintenance of software;
- Assist in the collection and documentation of user's requirements;
- Assist in the development of logical and physical specifications;
- Research and evaluate a variety of software products.



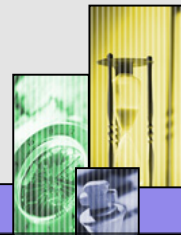
Characteristics

SE Characteristics	(%)	Programmer Characteristics	(%)
Male	81	Male	75
Female	18	Female	24
Full-Time	97	Full-Time	94
Part-Time	2	Part-Time	5
Self Employed	6	Self Employed	11
Employees	93	Employees	88
Usual place of work	89	Usual place of work	86
Worked at home	5	Worked at home	7
No fixed workplace address	2	No fixed workplace address	3
Worked outside Canada	2	Worked outside Canada	1



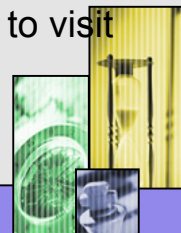
Skills

- strong problem solving and analytical skills.
- Knowledge of industry/business matters
- technical skills
- employers are placing more emphasis
 - written and verbal communication skills
 - project management skills
 - leadership and management skills
 - presentation skills
 - appropriate business experience.

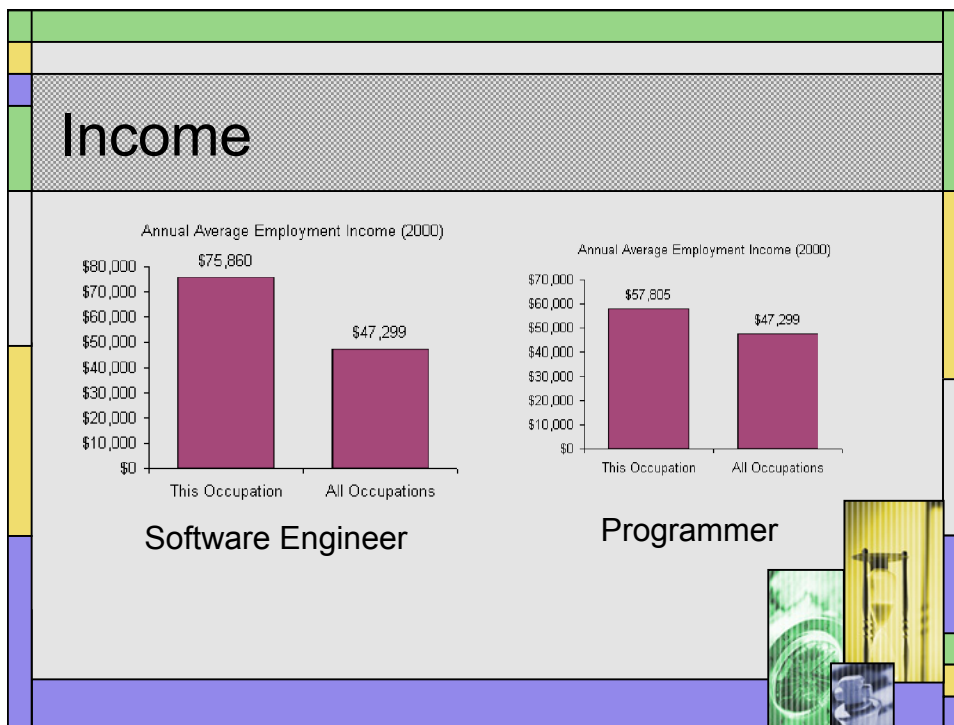


Work Environment

- typical 40 hour work week, with the occasional overtime.
- between 8:00am and 5:00pm, Monday to Friday, with some evening and weekend work
- primarily work indoors, usually in offices or computer laboratories.
- There may be frequent travel required to visit offsite clients.

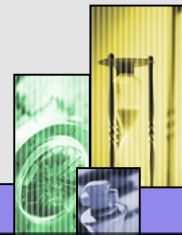


SE Industries of Employment (%)	
Professional scientific and technical services	50
Computer and electronic product manufacturing	18
Publishing industries	7
All Other Industries	24
Programmer Industries of Employment (%)	
Professional scientific and technical services	46
Credit intermediation and related activities	5
Federal government public administration	5
Publishing industries	4
Computer and electronic product manufacturing	4
Insurance carriers and related activities	3
Machinery equipment and supplies wholesaler-distributors	3
Broadcasting and telecommunications	2
Information services and data processing services	2
All Other Industries	22

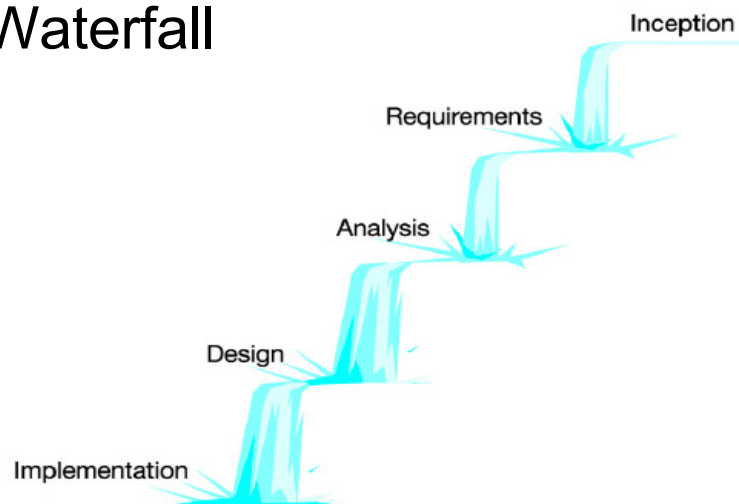


Computer Programming

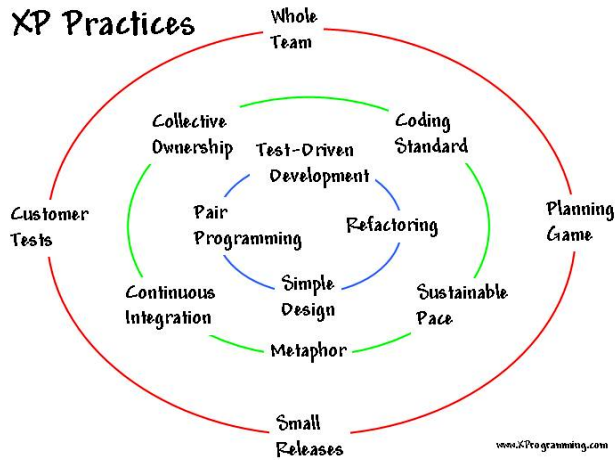
- The use has changed over the years
 - Expensive calculators
 - Computerization
 - Creating and Architecting Systems
- As the use changed, so did the method



Waterfall



Extreme Programming



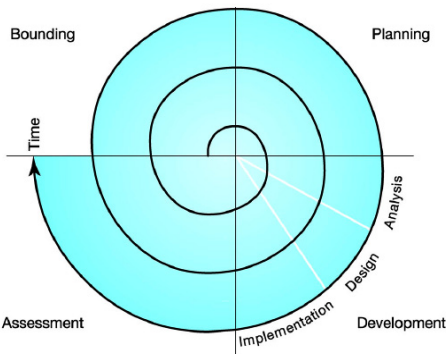
<http://www.xprogramming.com/images/circles.jpg>

The Spiral Approach

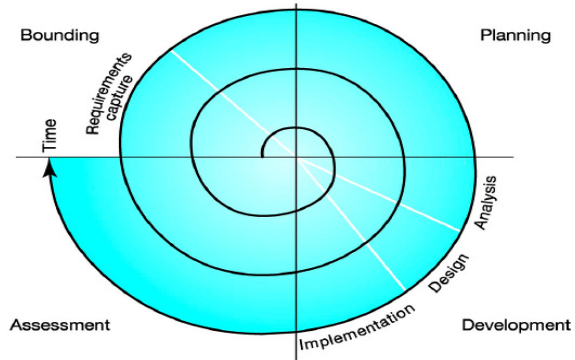
This depicts the phases of development in what is known as the spiral model.

We no longer kid ourselves that we can finish a phase to perfection and then move on to the next, never to go back.

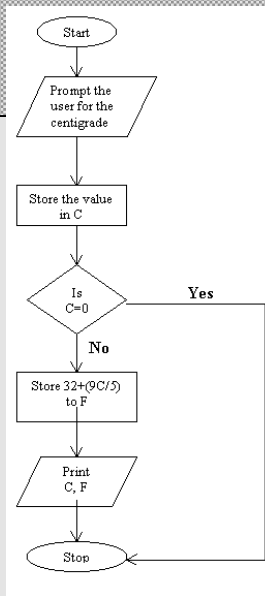
The details of the spiral model belong in a project control textbook, but you should be aware of it.



Requirements Capture In Spiral Model

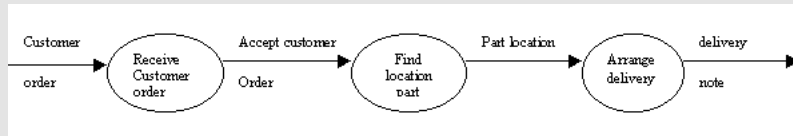


Flowchart



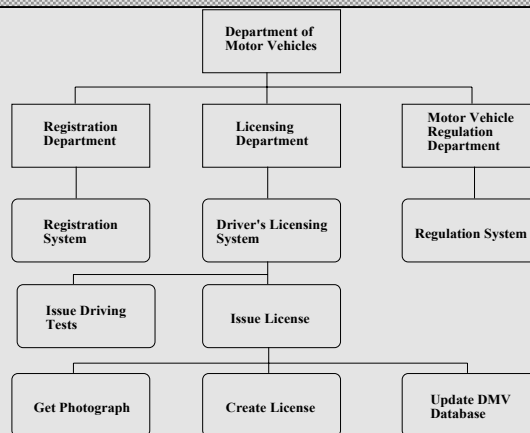
<http://www.nos.org/htm/sad2.htm>

Data Flow Diagram



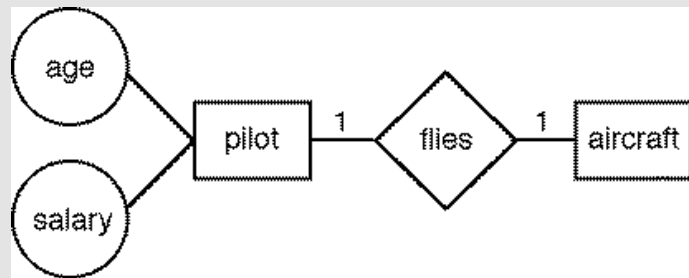
<http://www.nos.org/nim/sad2.htm>

Functional Decomposition



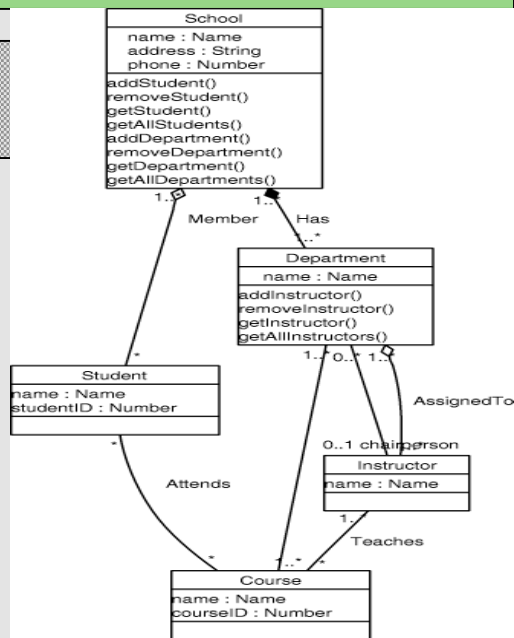
<http://www.sims.berkeley.edu/courses/is208/s01/Decomposition.doc>

Entity Relationship Diagram



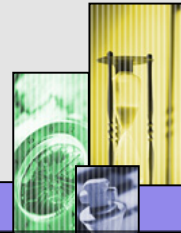
<http://kogs-www.informatik.uni-hamburg.de/~haarslev/publications/avitv196/node3.html>

Class Diagram

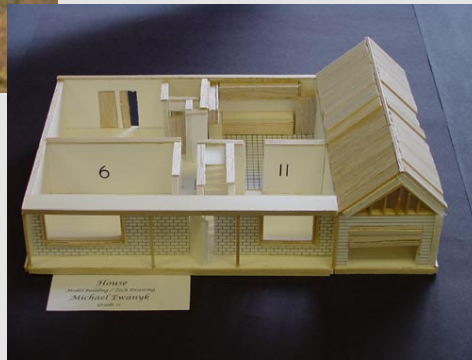


Case Studies used throughout book

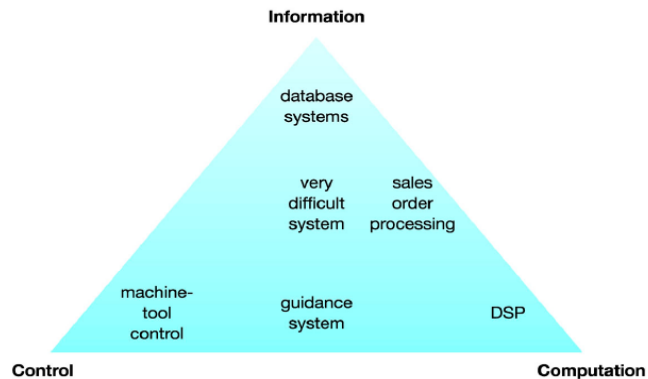
- Poirot- a crime system
- ALICE- a restaurant system
- STREPT- an academic system
- POLLY- a hotel system



Modelling and Models



Aspects of Computer Systems



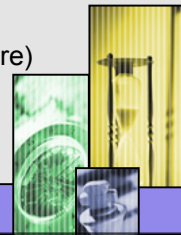
We Are Modelers

- Developing a computer system can be seen, then, as modeling
- What is a model?
 - It's a representation of the relevant aspects of some thing of interest,
 - In a different, more tractable, more convenient medium
- What is our medium?
 - Ultimately it's bits in the memory

The Modeling Gap

Can we model directly in bits?

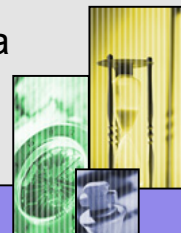
- No – we gave that up more than fifty years ago
- Let's think of that as a painter trying to with individual fragments (or molecules!) of pigment
- The ultimate medium we (users of programming languages) actually perceive today,
 - Is lines of code in a chosen, high-level (3rd generation) language
 - (Akin, perhaps, to using paint we bought from a store)



... The Modeling Gap

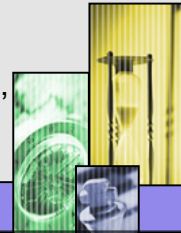
So can we model a subject matter directly in lines of code?

- No – the differences between lines of Java, C++, etc. and a typical subject matter is still too great
- Your money or your life test.
- (All but the bravest painters tend to do something else before they try to paint a fingernail or an eyelash)

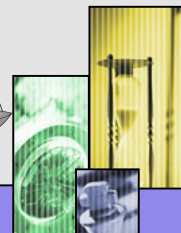
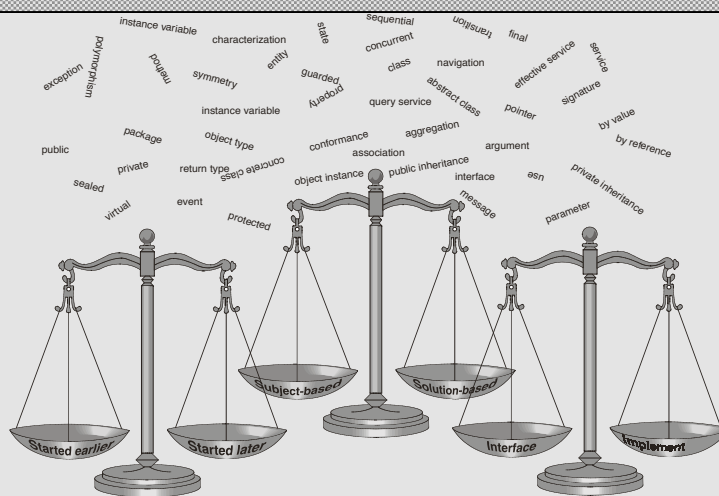


The Modeling Gap

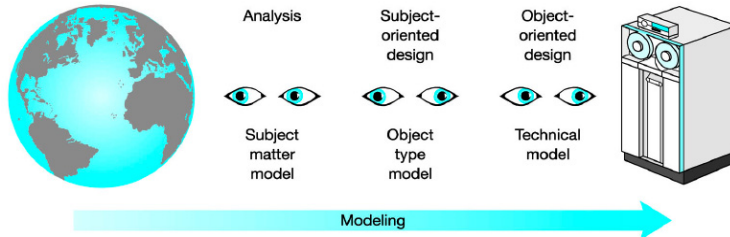
- We need a design, a plan
 - An equivalent of the painter's outline sketch
- OK, can we start there? Can we model a subject matter directly in `classes`, `interfaces`, `pointers`, etc.?
 - No – software has always been much more complex than paint or clay or foamboard etc.
 - There are too many possibilities, too many choices
 - The generality, the openness, the flexibility, the “soft” in software makes design a very difficult business



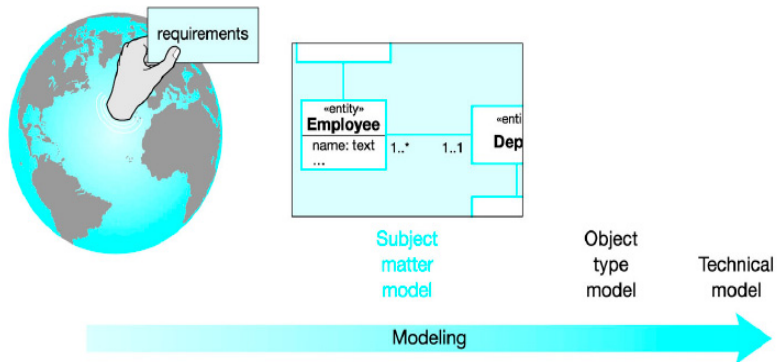
Software Design is Difficult



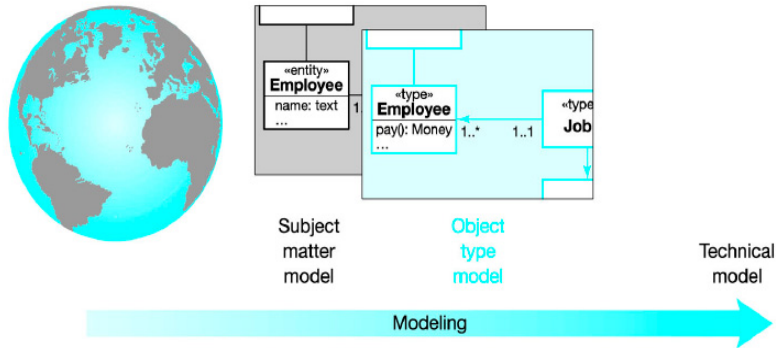
Three Models



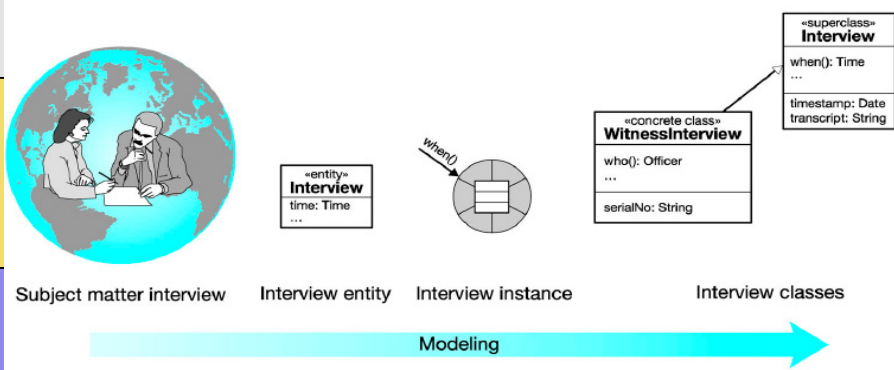
Subject Matter Model



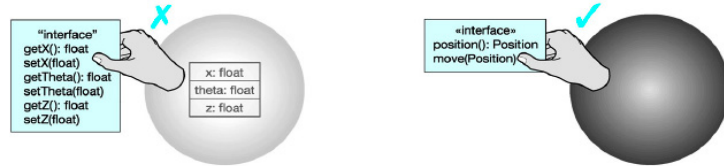
Object Type Model



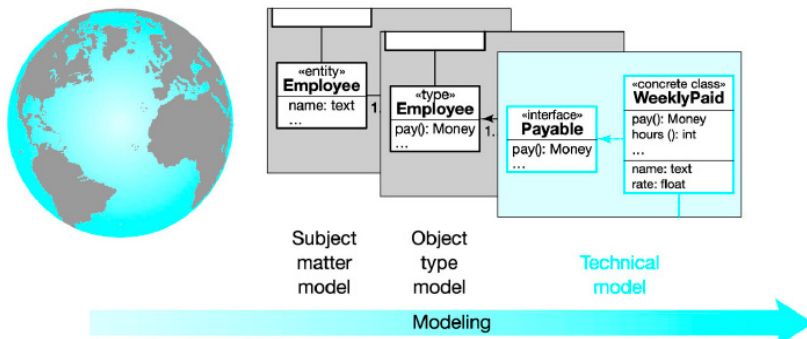
Clients, Types and Classes



Inside Out or Outside In



Technical Model



Devices in the Technical Model

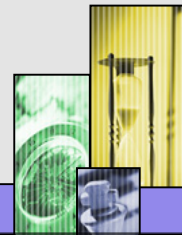
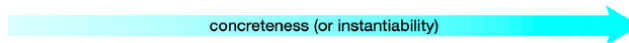
Pure type, e.g.
interface
purely abstract class (pABC)

Abstract class (ABC)

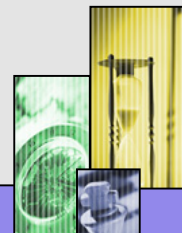
Concrete class

Object instance
The most concrete
thing would be an
instance in memory.)

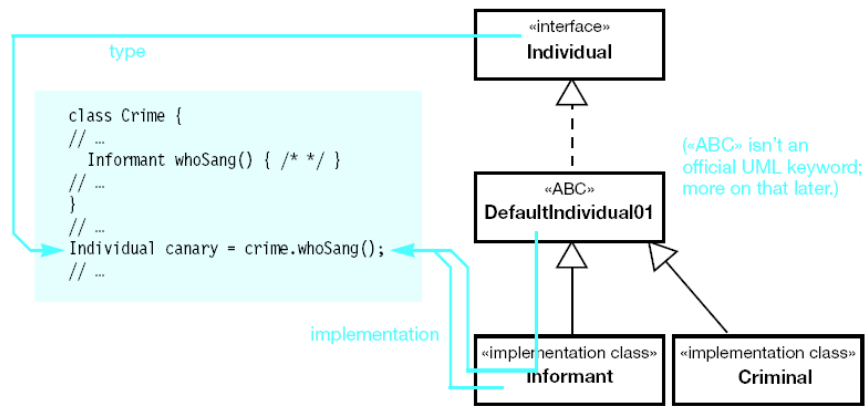
concreteness (or instantiability)



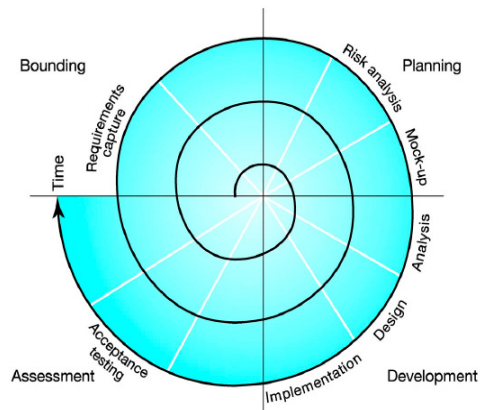
Polymorphism is your friend



Implementation and Type



Iterative Development



Agile Development (our aim)

- Neither too much nor too little modelling
- Subject matter model represents entities, attributes and state
- Object Type model represents methods, message passing, outside-in design
- Technical model has classes and specific OO constructs
- Code a little, test a little

