软件开发方法与技术

东南大学计算机科学与工程系
李 必 信

教学大纲

一、软件开发方法与技术概述
二、软件开发环境
三、基于UML和设计模式的软件开发技术
四、基于构件的软件开发技术
五、基于体系结构的软件开发
六、敏捷软件开发

课程作业

- 基于Internet的软件开发方法与技术
- 基于Internet的软件过程模型
- 基于Internet的软件分析与测试技术
- 基于Internet的软件工程

成绩计算

- 课程成绩＝课程考试（闭卷：50％）＋课程设计（40％）＋出勤率（10％）

参考文献

- 朱三元等，软件工程技术概论，科学出版社，2002。
- Craig Larman著，方梁等译，UML和模式应用（第二版），机械工业出版社，2004。
- 阮俊杰编著，软件开发方法与管理教程，北京希望电子出版社，2003。
- Perdita Stevens and Rob Pooley. Using UML-Software engineering with objects and components, Addison Wesly.2000

一、软件开发方法与技术概述

目录

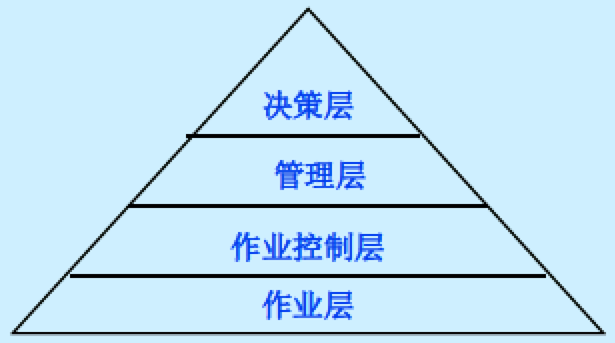
计算机系统＝硬件(略)＋软件

对软件的认识(1)

- 工具观 — 软件是一种代替、强化和延伸人类思维能力的工具。
- 逻辑观 — 计算机的思考方式是形式逻辑、形式逻辑通过建立一种可机械实现的符号演算机制，模拟了人类思维中的一类精确推理过程（如形式化语言、有限状态自动机理论）。
- 计算观 — 计算的实质就是寻找一种可机械执行的算法。例如，著名的计算模型：图灵机（被公认为现代计算机的祖先）。
- 组成观 — 软件可以用两个简单的算式来概括：（1）软件＝程序＋文档（2）程序＝算法＋数据结构
- 模型观 — 从模型论的角度看，整个软件系统是客户业务实现的一个模型，特别是软件的研制过程就是一个建模和模型间的演绎过程。

对软件的认识(2)

- 集成观 — 各种软件、硬件资源的集成
- 结构观 —



决策层是企业应用软件体系的最高层，通过对下面几层软件系统数据的分析、挖掘，通过规划论、决策论、对策论及数理统计的有关模型，对企业的决策层提供辅助支持

管理层的软件系统为企业的管理机构提供服务．如统计报表等

作业控制层的应用系统主要是为作业层提供一个协作和控制的环境

基础层，位于这个层次的软件系统的主要功能代替了传统的手工劳动

对软件的认识(3)

- 工程观 — 软件工程原理
- 对象观 — OOA&D理论认为，现实世界是由对象构成的，对象是具有特殊属性和行为的实体，对象间具有共性、层次性、继承性和关联性，反应了客观世界从普遍到一般、从个性到共性及相互制约的运动规律。从这个角度，软件的开发可视为“类”的抽象及其关联的建模过程，软件的运行是对象的实例化及其状态的演变过程。

对软件的认识(4)

- 生存期 — 根据ISO/IEC12207软件生命周期标准，一个通用的软件生命周期模型：（1）定义阶段（计划、需求分析）；（2）开发阶段（概要设计、详细设计、编码、集成测试、试运行和验收）；（3）运行维护
- 质量观 — 一种三维软件质量衡量：(1）软件运行（正确性、可用性、强壮性、安全性、性能）；（2）软件维护（可理解性、可扩展性、易维护性、灵活性）；（3）软件移植（可移植性、可复用性、互操作性）。

对软件的认识(5)

- 人机工程观—好的系统应该易于学习、理解和掌握；界面应该简洁。

应用系统体系结构

- 基本类型

1.个人单机软件 (桌面操作系统,OFFICE,杀毒软件等)
2.服务器批处理软件(OS自主调度,事件驱动,控制台触发等)
3.联机事务处理()
4.异步事务处理
5.非可靠事务处理

- 基本结构类型

1.终端服务器模式
2.瘦客户端的C/S结构
3.胖客户端的C/S结构
4.三层C/S结构
5.B/S体系结构
6.体系结构的设计要点

- 分布式计算标准

1.DCE:OSF(开放软件基金组织)1991提出分布计算标准
2.CORBA:OMG组织推出的对象管理模型
3.DTP:OPEN GROUP提出的网络计算模式下的一种通用体系结构

- 产品解决方案

1.Windows 2000 DNA
2.Windows.NET
3.J2EE
4.Web Services
5.Enterprise Portal
6.Grid Computing

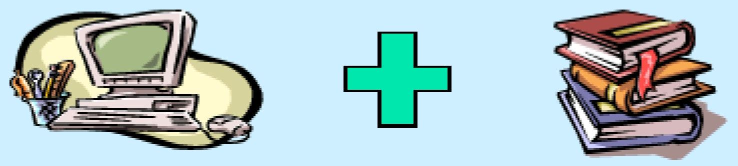
与软件相关的一些基本概念

1.What is software?
2.What is software engineering?
3.What is the difference between software engineering and computer science?
4.What is the difference between software engineering and system engineering?
5.What is a software process?
6.What is a software process model?

7. What are the costs of software engineering?
8. What are software engineering methods?
9. What is CASE (Computer-Aided Software Engineering)
10. What are the attributes of good software?
11. What are the key challenges facing software engineering?

1.What is software?

Computer programs **and** associated documentation



Software products **may be developed for a particular customer or may be developed for a general market**

Software products **may be**

- Generic **- developed to be sold to a range of different customers**
- Bespoke **(custom) - developed for a single customer according to their specification**

What is software (cont.)

Software is a set of items or objects that form a "configuration" that includes

- Programs
- documents
- data ...

2.What is software engineering(1)

Software engineering is an engineering discipline which is concerned with all aspects of software production

Software engineers should

- adopt a systematic and organised approach to their work
- use appropriate tools and techniques depending on
  - the problem to be solved,
  - the development constraints and
  - the resources available

What is software Engineering(2)

IEEE, 1990:

- (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of [quality] software [and software services] that is, the application of engineering to software.
- (2) The study of approaches as in (1)

Why Software Engineering?

1.Software development is hard !

Important to distinguish "easy" systems (one developer, one user, experimental use only) from "hard" systems (multiple developers, multiple users, products)

Experience with "easy" systems is misleading

One person techniques do not scale up

Analogy with bridge building:

Over a stream = easy, one person job

Over River Severn … ? (the techniques do not scale)

Why Software Engineering ?(cont.)

2.The problem is complexity

Many sources, but size is key:

UNIX contains 4 million lines of code

Windows 2000 contains 10^8 lines of code

Software engineering is about managing this complexity.

3.Software Engineering – Computer Science

- CS to SE is as Chemistry to CE.
- CS and programming languages are only tools to SE.
- SE is an art and need to be an engineering.
- Any one can write code to solve a problem but only SE can produce code that is robust and easy to understand and maintain and does it in most efficient way.

4.What is the difference between software engineering and system engineering(1)

System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering

Software engineering is part of this process

System engineers are involved in system specification, architectural design, integration and deployment

What is the difference between software engineering and computer science(2)

Computer Science	Software Engineering
is concerned with	is concerned with
➢ theory ➢ fundamentals	➢ the practicalities of developing ➢ delivering useful software

Computer science theories are currently insufficient to act as a complete underpinning for software engineering

What is a Software Engineering Process ?

❑Defines Who is doing What, When to do it, and How to reach a certain goal.

$$\begin{tikzcd} {} \arrow[r, "\text{New or Change}", "\text{Requirements}"'] & \boxed{\text{Software-Engineering Process}} \arrow[r, "\text{New or Changed}", "\text{System}"'] & {} \end{tikzcd}$$

- **The software engineering process is the process of developing a system from requirements, either new or changed.**
- **It is a controlled approach to a development task which follows a number of predefined activities and milestones and has a means to control and monitor the progress.**

5.What is a software process?

A set of activities whose goal is the development or evolution of software

Generic activities in all software processes are:

- Specification and verification- what the system should do and its development constraints
- Development - production of the software system
- Validation - checking that the software is what the customer wants
- Evolution - changing the software in response to changing demands

6.What is a software process model?

A simplified representation of a software process, presented from a specific perspective

Examples of process perspectives:

Workflow perspective	**represents inputs, outputs and dependencies**
Data-flow perspective	**represents data transformation activities**
Role/action perspective	**represents the roles/activities of the people involved in software process**

Generic process models

- Waterfall
- Evolutionary development
- Formal transformation
- Integration from reusable components

7.What are the costs of software engineering?

Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs

Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability

Distribution of costs depends on the development model that is used

8.What are software engineering methods?

Structured approaches to software development which include system models, notations, rules, design advice and process guidance

Model descriptions (Descriptions of graphical models which should be produced)

Rules (Constraints applied to system models)

Recommendations (Advice on good design practice)

Process guidance (What activities to follow)

9.What is CASE ? (Computer-Aided Software Engineering)

Software systems which are intended to provide automated support for software process activities, such as requirements analysis, system modelling, debugging and testing

Upper-CASE

Tools to support the early process activities of requirements and design

Lower-CASE

Tools to support later activities such as programming, debugging and testing

10. What are the attributes of good software?

The software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

Maintainability: Software must evolve to meet changing needs

Dependability: Software must be trustworthy

Efficiency: Software should not make wasteful use of system resources

Usability: Software must be usable by the users for which it was designed

11.What are the key challenges facing software engineering?

Software engineering in the 21st century faces three key challenges:

Legacy systems: **Old, valuable systems must be maintained and updated**

Heterogeneity: **Systems are distributed and include a mix of hardware and software**

Delivery: **There is increasing pressure for faster delivery of software**

Software Development
Why is it so Challenging (Difficult) ?

- Software development is a highly intellectual activity and so is heavily dependent on human thought.
- No valid theoretical models of software development exist.
- No laws of "software physics" that can define logical constraints and relationships between various variables in the project and product environment.

The Nature of Software (Brooks)

- The software essence
  - Complexity
  - Conformity
  - Changeability
  - Invisibility
- The software accidents
  - Stakeholders
  - Process
  - Modeling language and tools

软件生命周期

- Coarse granularity
  - 1.1 Analysis
  - 1.2 Design
  - 1.3 Implementation
  - 1.4 Maintenance
- Refined granularity
  - 2.1 Application Requirements determination
  - 2.2 Software Requirements specification
  - 2.3 Architectural design
  - 2.4 Detailed design
  - 2.5 Implementation
  - 2.6 Integration
  - 2.7 Maintenance

1 Application Requirements Phase

- Requirement – statement of a system service or constraint
- Service
  - Business rule
  - Computation
- Constraint
- Information gathering
- Requirements document

2 Software Specification Phase

- Application Requirements document → Software Specification document
- Visual modeling
  - Class diagrams
  - Use case models
- Implementation independent

3 Architectural Design

- Solution strategy
  - Client
  - Server
  - Application logic layer
- Modules
- UML:
  - Packages
  - Components
  - Deployment

4 Detailed Design

- User interface (client)
- Database (server)
- Application logic
  - Algorithms
  - Functions
  - Methods

5 Implementation

- Coding
- Testing
- Performance tuning
- DBA
- Installation
- User training

6 Integration

- Incremental integration
- Dependencies between modules (coupling)
  - Stubs
  - Drivers
- Uniform distribution of intelligence in modern OO systems
- Designing OO systems for integration

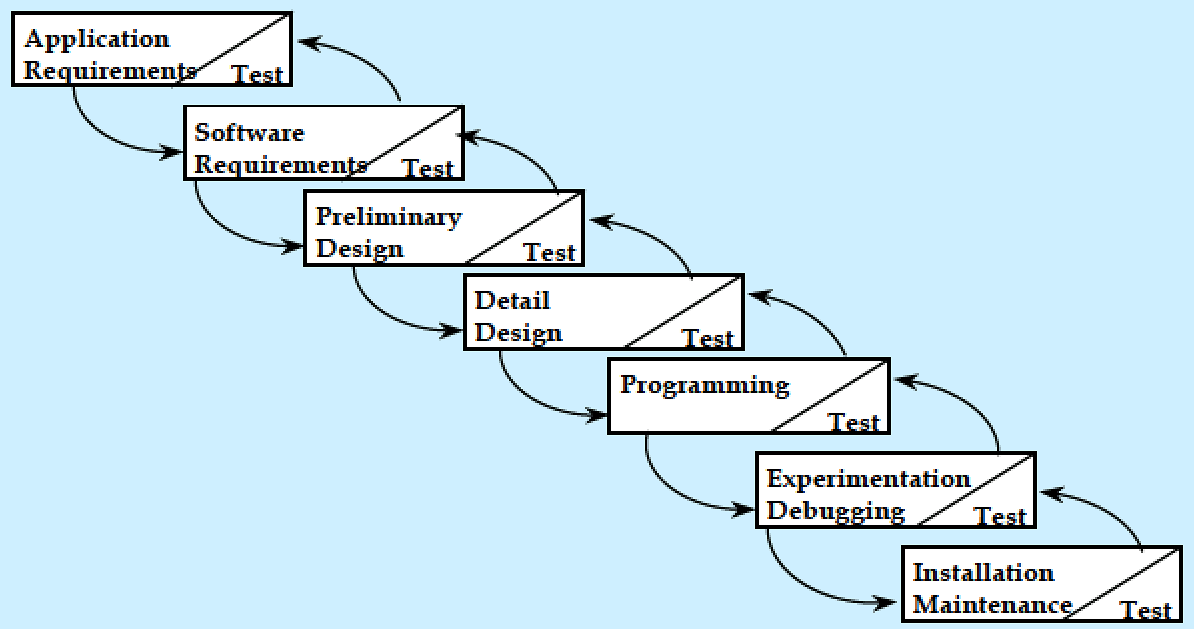
7 Maintenance

- Housekeeping
- Adaptive maintenance
- Perfective maintenance
- Software phasing-out
  - Perfective maintenance cannot help
  - Unpredictable effects of changes
  - Lack of documentation
  - Platform to be replaced

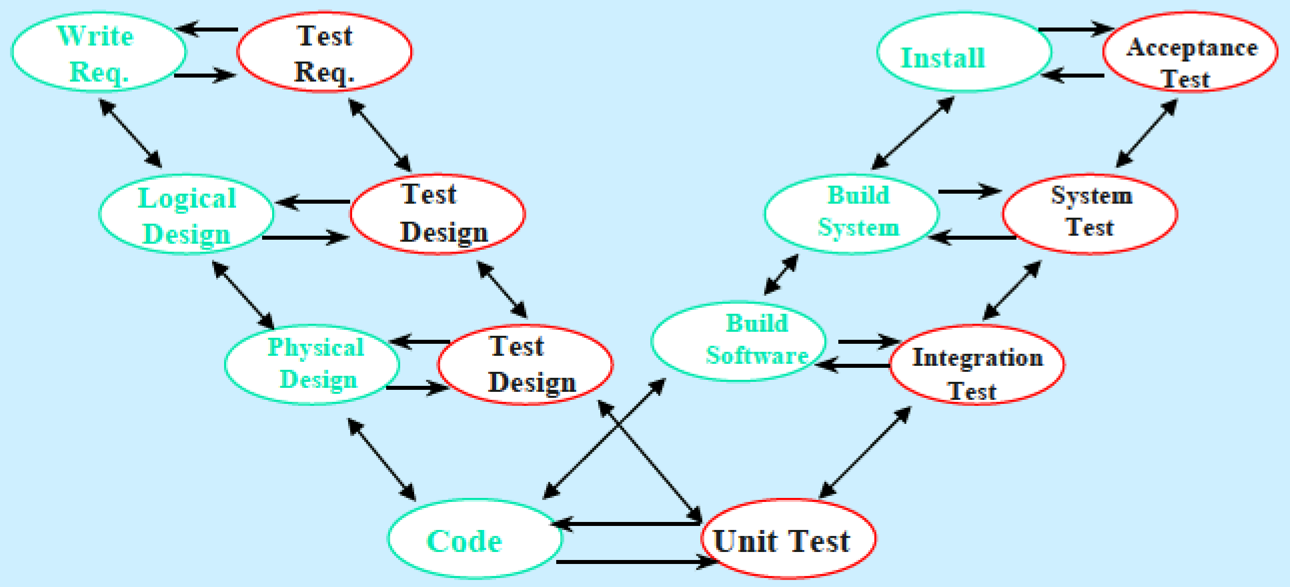
软件过程模型

- Linear
  - **The Waterfall**
  - **The W**
- Spiral

1 The Waterfall Model (B.W. Boehm)



2 W - Whole Life Cycle

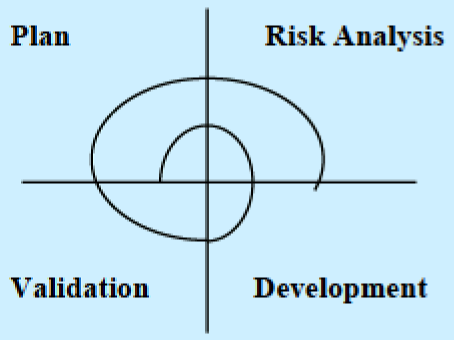


3 Traditional Life Cycle

- Linear, sequential phases
- No overlap between phases
- Assumes stable requirements up-front
- Even if everything goes right, what's wrong ?

4 Spiral (I ** 3) Life Cycle

- Antithesis of Waterfall
- Iterative - many refinements
- Incremental - moving forward:
  - functions, features, quality
- Integrative - continual integration of work products
- Non-linear, evolving prototypes
- More time spent in Analysis & Design
- Less time spent in Development



1:制定计划
2:风险分析
3:实施开发
4:客户评估

The Rational Unified Process(RUP)

1 Characteristics of the Process

- Iterative Process
- Architecture Centric
- Software Modeling
- Use Case Driven
- Supports Object-Oriented Techniques
- Scalable
- Ongoing Quality Control & Risk Management

2 Phases and Iterations (1)

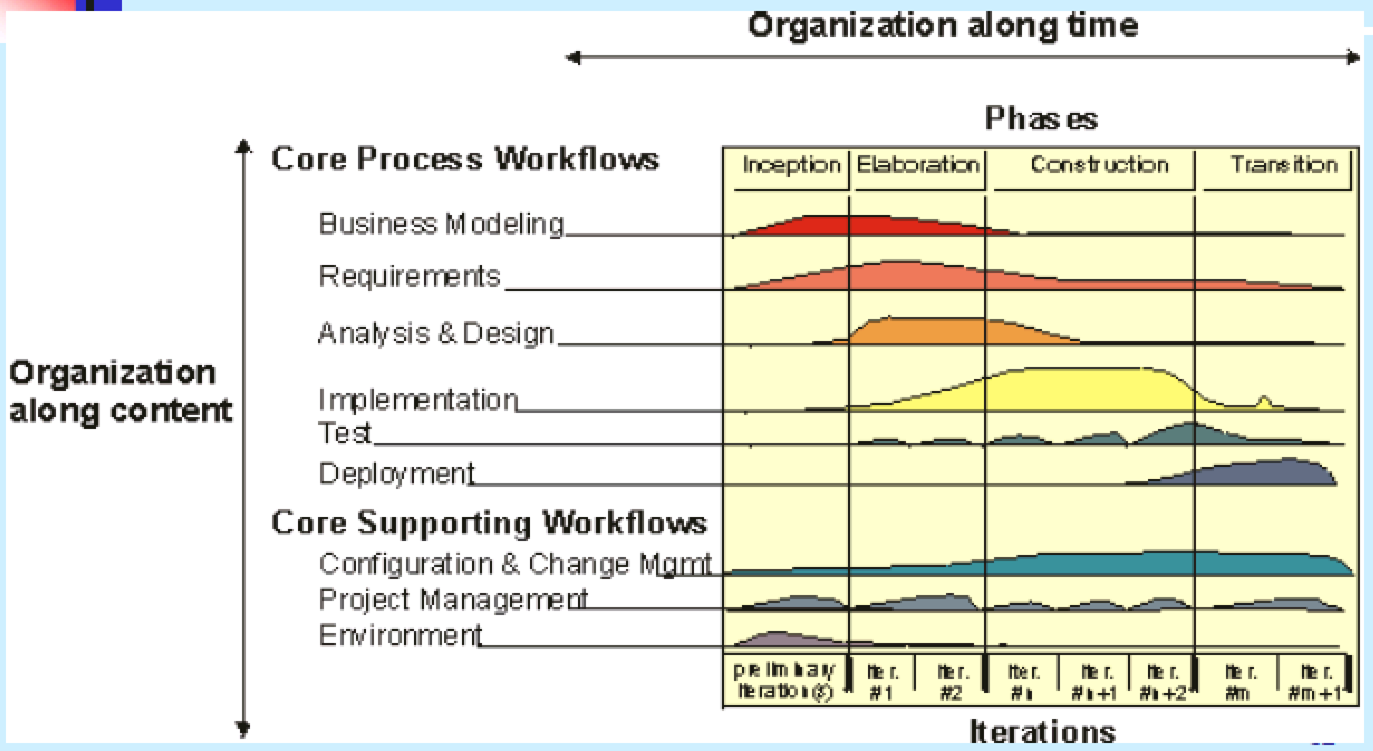
A phase is the span of time between two major milestones of the process in which a well-defined set of objectives are met, artifacts are completed, and decisions are made whether to move into the next phase.

Four phases:

- Inception - Establish the business case for the project
- Elaboration - Establish a project plan and a sound architecture
- Construction - Grow the system
- Transition - Supply the system to its end users

An iteration represents a complete development cycle, from requirements capture in analysis to implementation and testing, that results in the release of an executable project.

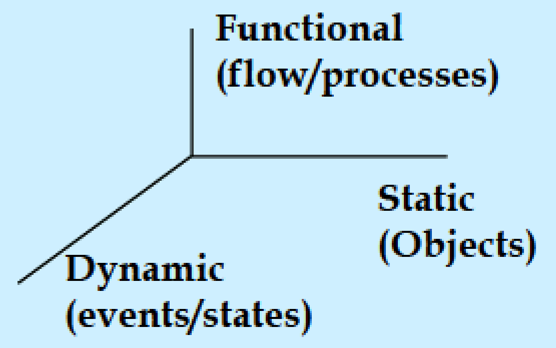
3 Phases and Iterations (2)



Software Development Methodology

1 The Three Dimensions of a System

Static: Represents the static, structural, “data” aspects of a system.
Dynamic: Represents the temporal, behavioral, “control” aspects of a system.
Functional: Represents the transformational, “function” aspects of a system.



2 Software Development Approaches

- The past – **basic/c/pascal/fortran etc.**
  - Procedural programs
  - Deterministic execution
  - Program in control
- The present- **c++/Java/python etc.**
  - Interactive program
  - Event-driven execution
  - Objects
- Structured vs. Object-Oriented

3 The Structured Approach

- **结构**：是指系统内各组成要素之间的相互联系、相互作用的框架。
- **结构化方法**：强调结构的合理性，以及所开发的软件结构合理性，由此提出了一组提高软件结构合理性的准则，如分解和抽象、模块的独立性、信息隐藏等。针对不同的开发活动、它有结构化分析、结构化设计、结构化编程和结构化测试。
- IBM的“输入－加工－输出”（简称IPO）80年代流行的开发方法

3 The Structured Approach(2)

- Modeling Techniques
  - Functional Decomposition
  - Data Flow Diagrams (DFDs)
  - Entity Relationship Diagrams (ERDs)
  - State Diagrams (STDs)
  - Data Dictionaries
  - Structure Charts
- Problems
  - Sequential and transformational
  - Inflexible solutions
  - No reuse

4 The Object Oriented Approach(1)

- 20世纪50年代人工智能语言LISP引入了动态联编（dynamic binding）和交互式开发环境的思想；60年代后期的离散事件模拟语言SIMULA67首次引入类的概念和继承机制；70年代的抽象数据类型，尤以CLU语言、MODULA-2语言和Ada语言影响较大。
- 但真正的面向对象程序设计的兴起，主要应归功于Xeror公司的Palo Alto研究中心（PARC）的Dynabook项目。该项目采用的程序设计语言Smalltalk-72，1981推出Smalltalk-80.
- 面向对象方法起源于面向对象编程语言。自20世纪80年代中期到90年代，面向对象的研究重点已从编程语言转移到程序设计方法学上来。

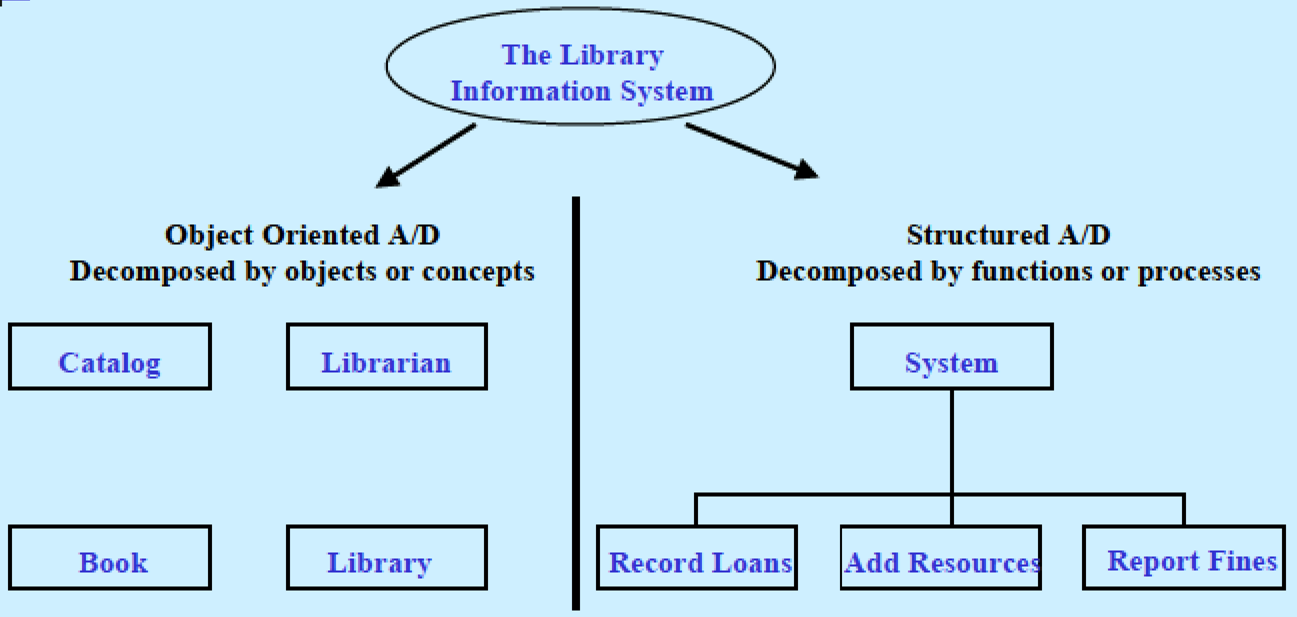
4 The Object Oriented Approach(2)

- The focus is on:
  - Responsibility and Roles
  - Co-operation and Collaboration between objects
- The 3 aspects of a system (Function, Data and Behavior) are melted into one concept and are considered as one unit

5 Structured vs. Object Oriented

- In structured approach the system decomposition (divide-and-conquer) is primarily by function or process, resulting in a hierarchical breakdown of processes composed of sub-processes. This is a solution oriented approach.
- In Object Oriented approach the problem space is decomposed by objects. This is a problem domain oriented approach.

6 Structured vs. OO Example



Software Modeling

1 What is a Model ?

A model is an abstraction of something for the purpose of understanding it before building it.

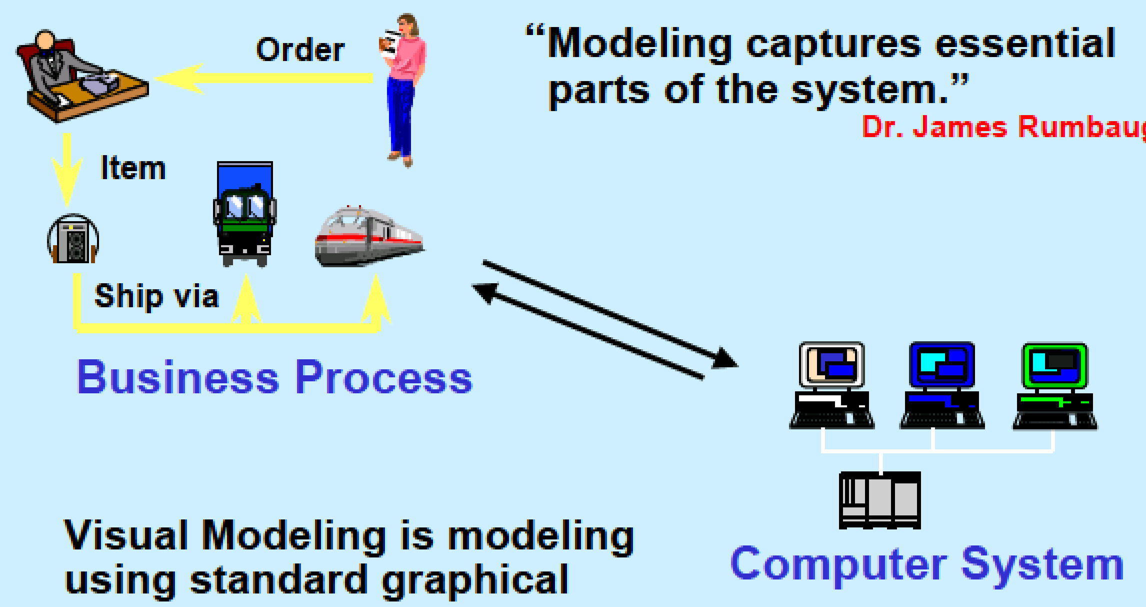
Models serve several purposes:

- Understanding the problem
- Testing a physical entity before building it
- Communication with customers
- Visualization
- Reduction of complexity
- Finding errors and omissions
- Planning the design
- Generating code

2 Why do we model?

- Furnish abstractions to manage complexity.
- Provide structure for problem solving.
- Experiment to explore multiple solutions.
- Reduce time-to-market for business problem solutions.
- Decrease development costs.
- Manage the risk of mistakes.

3 What is Visual Modeling?



"Modeling captures essential parts of the system."
Dr. James Rumbaugh

Visual Modeling is modeling using standard graphical notations

4 Object Modeling Methods

Following is a list of the most common methods that are in use:

✶	OOD	G. Booch	1991
✸	HOOD	HOOD Technical Group	1993
✹	OOA	S. Shlaer and S. Mellor	1988, 1992
❋	OOA/OOD	T. Coad and Y. Yourdan	1991
⌗	OMT	J. Rambaugh, M. Blaha, …	1991
✧	OOSE	I. Jacobson, …	1992
✧	FUSION	D. Coleman, …	1994
⌑	**UML1.0-1.5:**	**Booch, Rambaugh, Jacobson**	**1998-2003**

5 Object Management Group (OMG)

- Was founded in April 1989 by eleven companies, including 3Com, American Airlines, Canon, Data General, HP, Philips, Sun and Unisys Corporation.
- Establishment of industry guidelines and standardized object software modeling to provide a common framework for application development.
- http://www.omg.org/

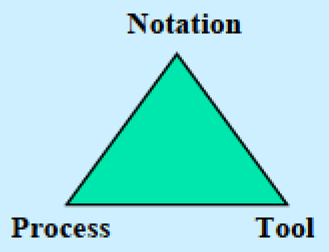
6 What is the UML ?

OMG's Unified Modeling Language™ (UML™)
Helps you specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements.

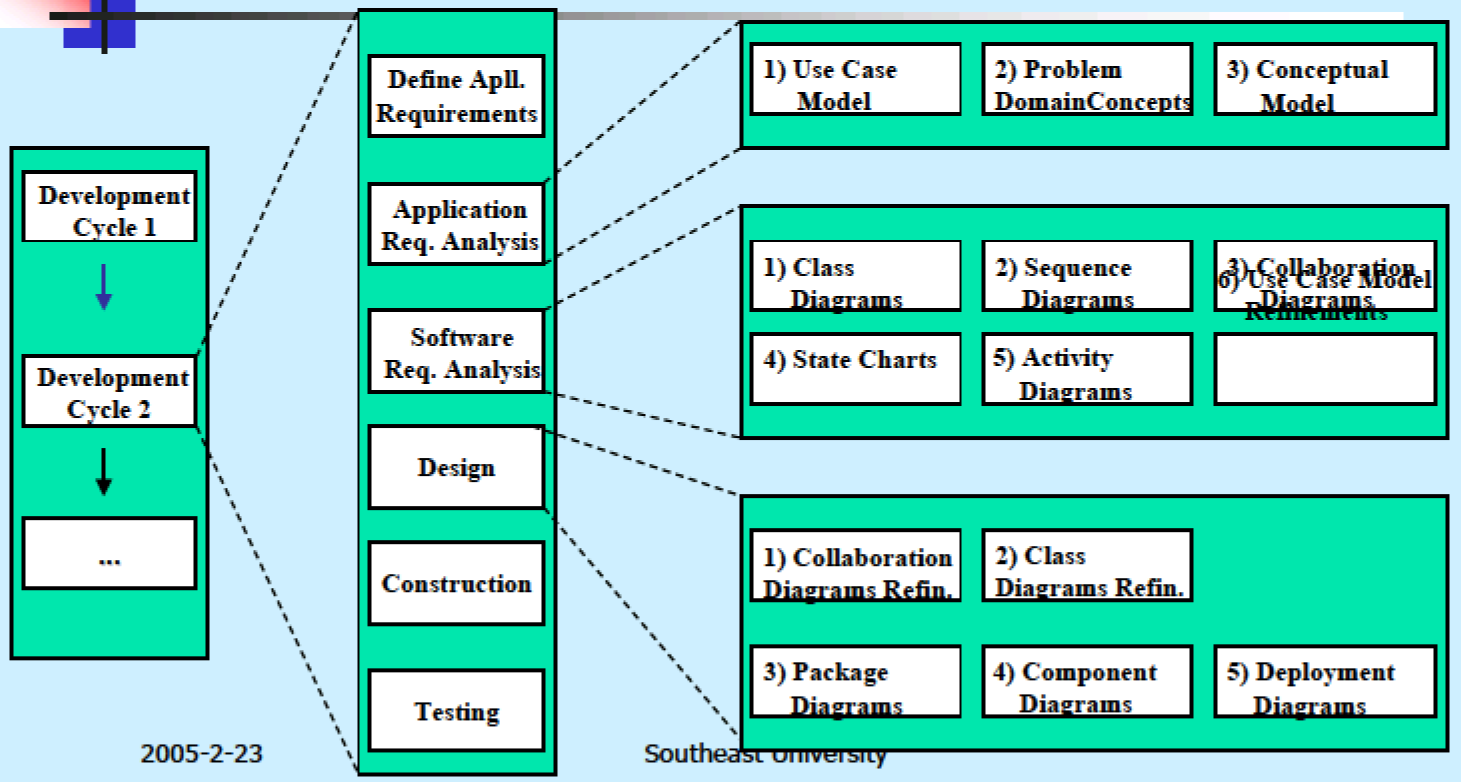
- UML stands for Unified Modeling Language
- The UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
- It can be used with all processes, throughout the development life cycle, and across different implementation technologies.

7 Unified Modeling Language - UML

- The UML is a notational system (including semantics for its notations) aimed at modeling system using object-oriented concepts.
- The UML has been accepted as an industry standard for object-oriented modeling by OMG (Object Management Group, an industry standards body).
- The UML is a language for software modeling; it does not guide a developer in how to do object-oriented analysis and design or what development process to follow, but it provides framework and the communication mechanism in the process of designing OO systems.



8 Development Life Cycle with UML



9 The basic building blocks of UML

- model elements **(classes, interfaces, components, use cases, etc.)**
- relationships **(associations, generalization, dependencies, etc.)**
- diagrams **(class diagrams, use case diagrams, interaction diagrams, etc.)**

Simple building blocks are used to create large, complex structures.

Formal Modeling Methods

- **Mathematics-based**
- **Unambiguous**
- **For specification/requirements**
  - usually including correctness proof rules
  - usually including specific notations
  - usually including process/methodology
  - usually including tools
- **The Z Notation/VDM/SDF+ASF/Refinement Calculus**
- **Specification Description Language - SDL**

其它软件开发

- 基于构件的软件开发(四)
- 基于体系结构的软件开发(五)
- 敏捷软件开发(六)

Summary

- **Nature** of software development – **craft** or even **art**
- **The triangle for success** – stakeholders, process, modeling language and tools
- The software development **lifecycle**
- **Structured** development approach
- **Object-oriented** development approach
- Software Modeling
- Other methods