软工基-Review

RandomStar in 2020.08

0. 关于考试

- 据说考试主要由选择+判断+设计题组成，其中选择判断题是有题库的

0.1 画图

- 设计题根据对历年卷的观察，主要是画各种图为主，包括：
  - 数据流图(data flow diagram)
  - CRC卡
  - 类的状态图(state diagram)
  - 软件系统架构图——调用返回架构
  - 简要介绍测试方法
  - NSU 图
  - RMMM计划(这个不知道软工基有没有？)

0.2 一些容易错的题目

第二章

- In general software only succeeds if its behavior is consistent with the objectives of its designers. 这句话是错的
- Which of these are the 5 generic software engineering framework activities?
  - 答案是： communication, planning, modeling, construction, deployment

第三章

- The communication activity is best handled for small projects using six distinct actions (inception, elicitation, elaboration, negotiation, specification, validation).
  - 对小型的软件项目来说沟通最好用六个不同的动作来处理(先启、引出、精化、协商、规范、确认)
  - 这句话是错的，但是PPT上没有

第四章

- 关于瀑布模型：本题答案是A，瀑布模型是一个在需求被定义的很好时，可以接受的实践方式

```
1. The waterfall model of software development is
A. A reasonable approach when requirements are well defined.
B. A good approach when a working program is required quickly.
C. The best approach to use for projects with large development teams.
D. An old fashioned model that is rarely used any more.
```

- The incremental model of software development is
  - 答案是在核心产品需要快速开发的时候非常好的方法
- 原型模型适合在需求不明确的时候使用
- 关于螺旋模型：本题选C。包含风险评估环节，但是AB是错的

```
5. The spiral model of software development
A. Ends with the delivery of the software product.
B. Is more chaotic than the incremental model.
C. Includes project risks evaluation during each iteration.
D. All of the above.
```

- 关于并行开发模型；本题选E
  - 并行开发模型也叫并行工程，会定义一系列的trigger engineering activity state transitions
  - 一般用于C/S体系的软件开发

```
6. The concurrent development model is
A. Another name for concurrent engineering.
B. Defines events that trigger engineering activity state transitions.
C. Only used for development of parallel or distributed systems.
D. Used whenever a large number of change requests are anticipated.
E. Both a and b
```

- 基于组件的开发模型需要依赖于object技术支持，有成本效益cost
- Unified Process统一过程中定义的步骤有：
  - Inception，Elaboration，construction，transition和production，没有validation
- 个人软件开发过程：不需要执行者的仔细监督，更依赖于个人的力量
- It is generally accepted that one cannot have weak software processes and create high quality end products.
  - 人们普遍认为一个人不能通过一个弱的软件过程来创造出一个高质量的软件

第八章

- Requirements engineering is a generic process that does not vary from one software project to another.
  - 软件需求分析是一个通用的过程，不会因为项目不同而不同，这句话是对的
- A stakeholder is anyone who will purchase the completed software system under development.
  - 利益相关者是购买软件的人，这句话是错的
  - 利益相关者还有开发，维护人员和项目经理等人
- It is relatively common for different customers to propose conflicting requirements, each arguing that his or her version is the right one.
  - 不同的客户提出相互冲突的需求是比较常见的，每个人都认为自己的版本是正确的
- 质量函数部署QFD中需要的需求：
  - 普通的需求，期望的需求，令人兴奋的需求
- User stories are complete descriptions the user needs and include the non-functional requirements for a software increment.
  - 用户需求描述是对用户需求的完整描述，包括了软件增量的非功能性的需求
- Analysis patterns facilitate the transformation of the analysis model into a design model by suggesting reliable solutions to common problems.
  - 分析模式通过为常见问题提供可靠的解决方案，促进了**分析模型到设计模型**的转换
- In agile process models requirements engineering and design activities are interleaved.
  - 敏捷模型模型中需求工程和设计是交叉的
- In requirements validation the requirements model is reviewed to ensure its technical feasibility.
  - 这句话是错的，具体原因未知

第九章

- Which of the following is not an objective for building a requirements model?、
  - 答案是 develop an abbreviated solution for the problem，这个属于设计的过程
- In structured analysis models focus on the structure of the classes defined for a system along with their interactions.
  - 这句话是错的，原因未知，PPT上没找到
- It is important to consider alternative actor interactions when creating a preliminary use case.
  - 错的，用例设计不能有可替代的交互
- In many cases there is no need to create a graphical representation of a usage scenario.
  - 这句话居然是对的，我也不是很懂为什么
- UML swimnlane diagrams allow you to represent the flow of activities by showing the actors having responsibility for **creating each data element.**
  - 错的，错的原因是 creating each data element，这里应该是use-case

第十章

- Attributes are chosen for an object by examining the problem statement and identifying the entities that appear to be related.
  - 这句话是错的，原因不明，看起来挺有道理的
- A stereotype is the basis for class reuse in UML modeling.
  - 中文翻译是：原型是UML中重用类的基础，但这句话是错的

第十一章

- For purposes of behavior modeling an event occurs whenever
  A. a state and process exchange information.
  B. the system and an actor exchange information.
  C. two actors exchange information.
  D. two objects exchange information.
  - 这题选B，行为的定义：系统和actor之间发生信息交互
- For purposes of behavior modeling a state is any
  A. consumer or producer of data.
  B. data object hierarchy.
  C. observable mode of behavior.
  D. well defined process.
  - 这题选C，状态是可以观察到的行为模式，其他几个都不对
- The UML sequence diagram shows the order in which system events are processed.
  - 错的，序列图描述的是不同成分之间的沟通
- What are the elements of a WebApp interaction model?
  - 肯定要有用例图和状态图，然后再看剩下的，感觉接口原型也很重要，这题就选C，反正PPT上没有找到
- Configuration analysis focuses on the architecture of the user's web browsing environment.
  - 错的，配置分为服务端的配置和浏览器端的配置，这里只有一部分

第十二章

- Which of the following is not a characteristic common to all design methods?
  - 答案是配置管理，凭感觉猜的，但好像真的是对的
- Cohesion is a qualitative indication of the degree to which a module
  A. can be written more compactly.
  B. focuses on just one thing.
  C. is able to complete its function in a timely manner.
  D. is connected to other modules and the outside world.

    - 选B，内聚力的定义，D是耦合度的定义
- Which of the following is not one of the five design class types
  A. Business domain classes
  B. Entity classes
  C. Process classes
  D. User interface classes
    - 答案是B，B是OO-Design中的三种类(实体类，边界类，控制类)之一
- Which design model elements are used to depict a model of information represented from the user's view?
  A. Architectural design elements
  B. Component-level design elements
  C. Data design elements
  D. Interface design elements
    - 答案是C，数据设计元素是从用户角度来描述的
    - 组件设计是floor of a house
    - 接口设计是draw of external house

第十三章

- The best representation of system architecture is an operational software prototype.
    - 错的，原因不明
- An architectural description is often documented using an architecture template.
    - 错的，没有提到用架构模板，PPT里说的是**a collection of products** to document an architecture
- An architectural decision is often documented using an architecture decision description template.
    - 到了架构决策这里就是对的了
- 架构的style重视组件，connector和约束条件，semantic模型
- To determine the architectural style or combination of styles that best fits the proposed system, requirements engineering is used to uncover
  A. algorithmic complexity
  B. characteristics and constraints
  C. control and data
  D. design patterns
    - 选B，原因不明，记住就可以
- Which of the following is not an example of infrastructure components that may need to be integrated into the software architecture?
  A. Communications components
  B. Database components
  C. Interface components
  D. Memory management components
    - 选C，PPT上没有，选的答案是最外层的
- Static architectural conformance checking assesses whether or not the source code matches the user visible requirements.
    - 错的，静态代码审查不检查用户需求是否匹配，根据SP学过的，感觉应该是检查代码的逻辑错误的

第十四章

- In the context of object-oriented software engineering a component contains
  A. attributes and operations
  B. instances of each class
  C. roles for each actor (device or user)

D. set of collaborating classes

- 选D，原因不明，ppt上也就这一句话

- Which of the following is not one of the four principles used to guide component-level design?
  A. Dependency Inversion Principle
  B. Interface Segregation Principle
  C. Open-Closed Principle
  D. Parsimonious Complexity Principle
  - 选D，开闭原则，依赖倒置原则和接口独立原则都是有的，D的简约复杂性原则没听说过
- WebApp content design at the component level focuses on content objects and the manner in which they interact.
  - 组件级的web应用设计的内容设计不关注交互而是关注package的方式
- Which of the following factors would not be considered during component qualification?
  A. application programming interface (API)
  B. development and integration tools required
  C. exception handling
  D. testing equipment required
  - 不需要考虑测试设备，选D
- Which is the following is a technique used for component wrapping?
  - 答案是干净盒子包装，其他三个选项都是黑盒白盒灰盒，听着像是软件测试的东西
- Which of the following is not one of the issues that form a basis for design for reuse?
  A. object-oriented programming
  B. program templates
  C. standard data
  D. standard interface protocols
  - 这题居然是选A，我人傻了
- In a reuse environment, library queries are often characterized using the ___ element of the 3C Model.
  - 三个C都要选，概念，内容，上下文

第十五章

- Which of the following interface design principles does not allow the user to remain in control of the interaction with a computer?
  A. allow interaction to interruptible
  B. allow interaction to be undoable
  C. hide technical internals from casual users
  D. only provide one rigidly defined method for accomplishing a task
  - 选D，只定义一个严格的方法来完成一项任务
- The reason for reducing the user's memory load is make his or her interaction with the computer quicker to complete.
  - 这个居然是错的，看了一下PPT没有提出原因
- If past interactive models have created certain user expectations it is not generally good to make changes to the model.
  - 如果过去的交互模型已经创建了某些用户期望，那么对模型进行更改通常是不好的。
  - 这句话是对的
- Which model depicts the image of a system that an end user creates in his or her head?
  - 答案是system perception
  - 如果问的是终端用户的配置文件，那就是user model
  - 描述接口设计的是implement model

- Which of these framework activities is not normally associated with the user interface design processes?
  A. cost estimation
  B. interface construction
  C. interface validation
  D. user and task analysis
    - 答案是cost estimation
- The computer's display capabilities are the primary determinant of the order in which user interface design activities are completed.
    - 这句话是错的

第十七章

- Which of the following characteristics should not be used to assess the quality of a WebApp?
  A. aesthetics
  B. reliability
  C. maintainability
  D. usability
    - 选A，美学
- With WebApps content is everything, a poorly defined user interface will be quickly overlooked by frequent users.
    - 糟糕的UI设计依然不会被忽略
- Screen layout design has several widely accepted standards based on human factors research.
    - 这个居然是错的，PPT上没有找到，只能记住
- Content objects are not normally chunked into Web pages until the implementation activities begin
    - 错的，原因不明
- Which of the following is not one of the content architectural structures used by web engineers?
    - 选parallel并行
- To allow the user to feel in control of a WebApp, it is a good idea to mix both horizontal and vertical navigation mechanisms on the same page.
    - 不是好主意，这个需要记住
- Which of these is not one of the design activities associated with object-oriented hypermedia design?
    - 选内容设计，另外三个都是OO超媒体设计的要素

第二十章

- Defect amplification models can be used to illustrate the costs associated with using software from its initial deployment to its retirement.
    - 翻译：缺陷放大模型可以用来说明软件从最初部署到退役的成本
    - 这句话是错的，应该是从最初的设计开始到系统测试期间
- A review summary report answers which three questions?
  A. terminate project, replace producer, request a time extension
  B. what defects were found, what caused defects, who was responsible
  C. what was reviewed, who reviewed it, what were the findings
  D. none of the above
    - 本题选C，记住即可，审核总结报告的三个问题：审核了什么，谁审核的，发现了什么

第二十二章

- The best reason for using Independent software test teams is that
  A. software developers do not need to do any testing
  B. strangers will test the software mercilessly
  C. testers do not get involved with the project until testing begins
  D. the conflicts of interest between developers and testers is reduced
    - 这个居然选D，只能记住，测试团队独立出来的原因是为了减少开发和测试之间的利益冲突
- Bottom-up integration testing has as it's major advantage(s) that
  A. major decision points are tested early
  B. no drivers need to be written
  C. no stubs need to be written
  D. regression testing is not required
    - 自底向上测试不需要stub，同样的自顶向下测试不需要driver
- Smoke testing might best be described as
  A. bulletproofing shrink-wrapped software
  B. rolling integration testing
  C. testing that hides implementation errors
  D. unit testing for small programs
    - 最好的描述是B，滚动的集成测试
- When testing object-oriented software it is important to test each class operation separately as part of the unit testing process.
    - 不是separately的，需要通过usage scenario来测试
- The OO testing integration strategy involves testing
  A. groups of classes that collaborate or communicate in some way
  B. single operations as they are added to the evolving class implementation
  C. operator programs derived from use-case scenarios
  D. none of the above
    - 本题的答案是A
- The focus of validation testing is to uncover places that s user will be able to observe failure of the software to conform to its requirements.
    - 对的
- Software validation is achieved through a series of tests performed by the user once the software is deployed in his or her work environment.
    - 错的，原因不明，怀疑是因为“用户部署”
- Acceptance tests are normally conducted by the 最终用户
- Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that software is able to continue execution without interruption.
    - 看的不是能否在不中断的情况下继续执行，而是看是否可以恢复正确的操作
- Stress testing examines the pressures placed on the user during system use in extreme environments.
    - 测试的不是用户的压力，而是软件在极端环境下的压力

第二十三章

- The testing technique that requires devising test cases to demonstrate that each program function is operational is called
  A. black-box testing
  B. glass-box testing
  C. grey-box testing
  D. white-box testing
    - 是黑盒测试的定义

    - 如果是 requires devising test cases to exercise the internal logic of a software module，涉及到内部逻辑的测试就是白盒测试，黑盒测试是只能从外面看的
- The cyclomatic complexity metric provides the designer with information regarding the number of
  A. cycles in the program
  B. errors in the program
  C. independent logic paths in the program
  D. statements in the program
    - 这个不知道是哪里来的，答案是C
- The cyclomatic complexity of a program can be computed directly from a PDL representation of an algorithm without drawing a program flow graph.
    - 这个是对的，不需要绘制流程图，可以直接计算出来
- 控制结构的测试：
    - condition test：关键词——内部的逻辑
    - data flow test：关键词——location of definition， use of variable
    - loop test：关键词——loop
- Real-time applications add a new and potentially difficult element to the testing mix
  A. performance
  B. reliability
  C. security
  D. time
    - 这题选D，实时应用就选时间

第二十四章

- It is not possible to test object-oriented software without including error discovery techniques applied to the system OOA and OOD models.
    - 对的，必须依赖OOA和OOD
- The correctness of the OOA and OOD model is accomplished using formal technical reviews by the software quality assurance team.
    - 错的，是基于现实问题和建模的一致性
- Test case design for OO software is driven by the algorithmic detail of the individual operations.
    - 居然是对的
- Validation of object-oriented software focuses on user visible actions and outputs from the system.
    - 这个也是对的，PPT上没有找到
- Scenario-based testing
  A. concentrates on actor and software interaction
  B. misses errors in specifications
  C. misses errors in subsystem interactions
  D. both a and b
    - 这个不能选D，只有一个A

第二十五章

- Which of the following is not one of the objectives of WebApp content testing?
  A. Find organizational or structure errors
  B. Identify linking errors
  C. Uncover semantic errors
  D. Uncover syntactic errors
    - 选B，这个不是内容测试中的环节

- Which of the following is not a WebApp interface mechanism?
  A. Browser
  B. Cookies
  C. Forms
  D. Links
  - 答案是C，不太懂
- WebApp compatibility testing is conducted to be sure that the user model for usage scenario matched the user category assigned to a given user.
  - 错的，兼容性测试的目的是测试客户端
- The purpose of WebApp navigation syntactic testing is to ensure the correct appearance of each navigation mechanism.
  - 错的，不是测试导航栏的外观
- Which of following is not one of the elements that need to be considered when constructing WebApp server-side configuration tests?
  A. Browser compatibility
  B. Database software integration
  C. Operating system compatibility
  D. System security measures
  - 服务端，不需要管浏览器
- Which of the following is not a testable WebApp security element?
  A. Authentication
  B. Encryption
  C. Firewalls
  D. Penetration
  - 选D，渗透测试
- WebApp performance tests are designed to
  A. asses WebApp usability
  B. evaluate page loading times
  C. simulate real-world loading situations
  D. test network connectivity
  - 选C，性能测试需要模拟真实的加载情况

第二十九章

- Which of the following is not considered one of the four important elements that should exist when a configuration management system is developed?
  - 这个要记住，SCM四个元素分别是：组件元素，过程元素，构建元素和人类元素
- Many data repository requirements are the same as those for a typical database application.
  - 这个是对的
- Which of the following tasks is not part of software configuration management?
  A. change control
  B. reporting
  C. statistical quality control
  D. version control
  - 答案是C统计质量控制
- A basic configuration object is a __ created by a software engineer during some phase of the software development process.
  A. program data structure
  B. hardware driver
  C. unit of information
  D. all of the above
  - 选C，PPT上没找到

- Version control systems establish a change set as part of their primary functionality.
  - 错的，没听说过这change set
- When software configuration management is a formal activity the software configuration audit is conducted by the
  A. development team
  B. quality assurance group
  C. senior managers
  D. testing specialists
  - 答案居然是B，人晕了，根本没有提到过的概念，但是记住就好
- Content management establishes a process by which Web content is rendered on the user's display screen.
  - 错的，不太懂这题

第三十一章

- The best project team organizational model to use when tackling extremely complex problems is the
  A. closed paradigm
  B. open paradigm
  C. random paradigm
  D. synchronous paradigm
  - 复杂问题用开放式架构
- One of the best ways to avoid frustration during the software development process is to
  A. give team members more control over process and technical decisions.
  B. give team members less control over process and technical decisions.
  C. hide bad news from the project team members until things improve.
  D. reward programmers based on their productivity.
  - 避免混乱的办法是加强控制
- Which of these software characteristics is not a factor contributing to project coordination difficulties?
  A. interoperability
  B. performance
  C. scale
  D. uncertainty
  - 不是导致协作困难的因素：是软件的性能
- Which of these software characteristics are used to determine the scope of a software project?
  A. context, lines of code, function
  B. context, function, communication requirements
  C. information objectives, function, performance
  D. communications requirements, performance, information objectives
  - 本题答案是C，除了C还有context
- When can selected common process framework activities be omitted during process decomposition?
  A. when the project is extremely small in size
  B. any time the software is mission critical
  C. rapid prototyping does not require their use
  D. never the activities are invariant
  - 选D，从来没有什么东西是不变的

第三十四章

- Which of the following is not one of the guiding principles of software project scheduling:
  A. compartmentalization
  B. market assessment
  C. time allocation
  D. effort validation
    - 选B，其他几个都是

第三十五章

- Proactive risk management is sometimes described as fire fighting.
    - 这句话是错的，救火是被动的风险管理
- Software risk always involves two characteristics
  A. fire fighting and crisis management
  B. known and unknown risks
  C. uncertainty and loss
  D. staffing and budget
    - 选C
- Three categories of risks are
  A. business risks, personnel risks, budget risks
  B. project risks, technical risks, business risks
  C. planning risks, technical risks, personnel risks
  D. management risks, technical risks, design risks
    - 这题的答案是B，PPT上提到了项目风险和技术风险
- The reason for refining risks is to break them into smaller units having different consequences.
    - 错的
- Hazard analysis focuses on the identification and assessment of potential hazards that can cause
  A. project termination
  B. schedule slippage
  C. cost overruns
  D. an entire system to fail
    - 选D，没找到

第一部分：软件工程的基本概念

1. Introduce

1.1 The Nature of Software 软件的本质

- 软件是什么？
    - 软件是一系列对象或者项目组成的结构，包含了
        - 程序指令：提供执行时需要的函数和功能
        - 数据结构：使得程序可以充分操作信息
        - 文档：描述程序如何操作和使用
    - 软件不会wear out 但是会 deteriorate
    - 虽然现在软件开发已经转向基于组件的集成，但大多数软件依然是定制的
- 软件应用的种类
    - 系统软件
    - 应用软件
    - 工程/科学软件

    - 嵌入式软件
    - 产品线软件
    - web应用
    - 人工智能软件
- 为什么旧版的软件需要更新
    - 必须对软件进行调整以满足新的计算环境或技术的需求
    - 必须对软件进行增强以实现新的业务要求
    - 必须对其进行扩展以使其能够与其他更现代的系统或数据库进行互操作
    - 必须对其进行重新架构以使其在网络环境中可行

2. Software engineering 软件工程

2.1 Definition

- IEEE给出的定义：
    - 在软件的开发，操作和维护中应用系统，规范，可量化的方法；即工程学在软件中的应用
    - 以及与之相关的研究

2.2 The Software Process

- 通用的软件开发流程框架
    - 沟通（客户协作和需求收集）
    - 计划（建立工程工作计划，描述技术风险，列出资源需求，生产的工作产品，并定义工作时间表）
    - 建模（创建模型以帮助开发人员和客户理解需求和软件设计）
    - 构建（代码生成和测试）
    - 部署（交付的软件用于客户评估和反馈）

2.3 软件工程的实践

- 实践的本质 The Essence of Practice
    - 理解问题(交流和分析)
    - 制定解决方案
    - 执行计划
    - 检测结果的正确性
- 总体原则：
    - 存在的全部价值——给使用者提供价值
    - KISS——保持简单和容易操作
    - 保持开放的眼界
    - What you produce, others will consume
    - 面向未来
    - 计划好复用
    - think

2.4 软件开发的故事

- 讲了几个很无聊的故事

3. 软件过程结构

3.1 总体过程模型

- Process flow 工艺流程：分为四种
    - 线性流程
    - 迭代流程
    - 进化流程：环形，有出口和入口

- 并行流程：多个过程并行

3.4 过程模式

- 流程模式定义了一组活动，动作，工作任务，工作产品和相关行为
  - 使用模板来定义一个模式
  - 通用的软件模式元素
    - 模式的名称
    - 模式的目标
    - 种类
    - Initial context：介绍应用场景和先决条件
    - solution：介绍如何解决
    - Resulting context
    - 相关模式
    - 例子和已知的应用

3.5 过程评估

- CMMI：The Capability Maturity Model Integration 能力成熟度模型集成
  - 级别0：未完成（未执行流程或未达到为此级别定义的所有目标）
  - 级别1：已执行（正在执行生产所需工作产品所需的工作任务）
  - 级别2：受管理（从事工作的人员可以访问足够的资源来完成工作，积极参与利益相关者的工作，监视，审查和评估工作任务和产品，以确保与过程描述保持一致
  - 级别3：已定义（将管理和工程过程记录，标准化并集成到组织范围的软件中）流程）
  - 级别4：量化管理（使用详细措施对软件过程和产品进行量化理解和控制）
  - 级别5：优化（通过对过程进行定量反馈并测试创新思想，可以实现持续的过程改进）

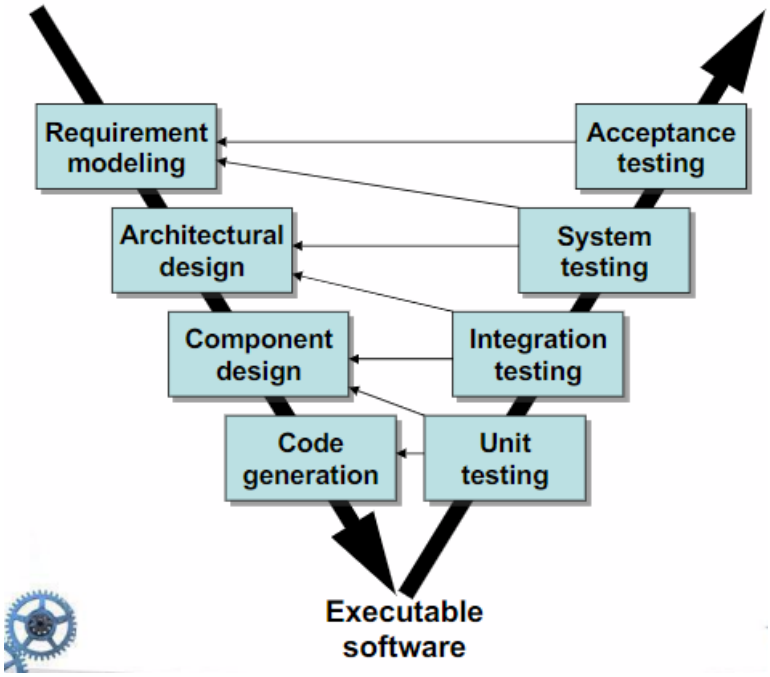
4. 过程模型

4.1 规范化的模型

- 规范化的模型主张有顺序地进行软件工程
  - Prescriptive process models advocate an orderly approachto software engineering

瀑布模型

- 分为交流，计划，建模，构建，部署等步骤
  - 真实的项目中一般不会按照这样的线性流程
  - 顾客不能精确提出所有的需求
  - 需要比较长的时间跨度来能获得一个可以使用的版本
- V-model



增量模型(渐进式模型)

- 先做出核心产品，在此基础之上不断添加新的功能和特征
- RAD模型(快速应用开发模型)
  - 需要足够多的人力资源
  - 需要开发人员和客户做出快速启动的承诺（？
  - 不能work的情况
    - 系统不能很好地模块化
    - 需要调整接口的时候
    - 技术要求非常高的项目

进化模型

- 在交流——快速计划/建模设计——建立原型——部署之间循环
  - 适合在客户有一定需求但是不知道细节的时候迈出第一步
  - 做出来的原型需要被弃用
- 螺旋模型
  - 模型图呈螺旋上升的形状
- 并行开发模型
  - 定义一系列事件，将触发每个活动，动作或任务从状态到状态的转换
  - 对于C/S体系应用开发特别好
  - 定义一系列活动而不是一串线性的事件
  - 注重高质量

4.2 专用过程模型

- 基于组件的开发-当重用是开发目标时要应用的过程
- 形式化方法-强调需求的数学规范
- 面向方面的软件开发-提供一种用于定义，指定，设计和构造方面的过程和方法

4.3 UML

- UML(统一建模语言)：用于描述用例驱动的，架构为核心的，可迭代的和增长的软件工程过程，分为如下几个过程
  - 开始阶段 inception

    - 细化阶段 elaboration
    - 构建阶段 construction
    - 过渡阶段 transition

4.4 个人和团队过程模型

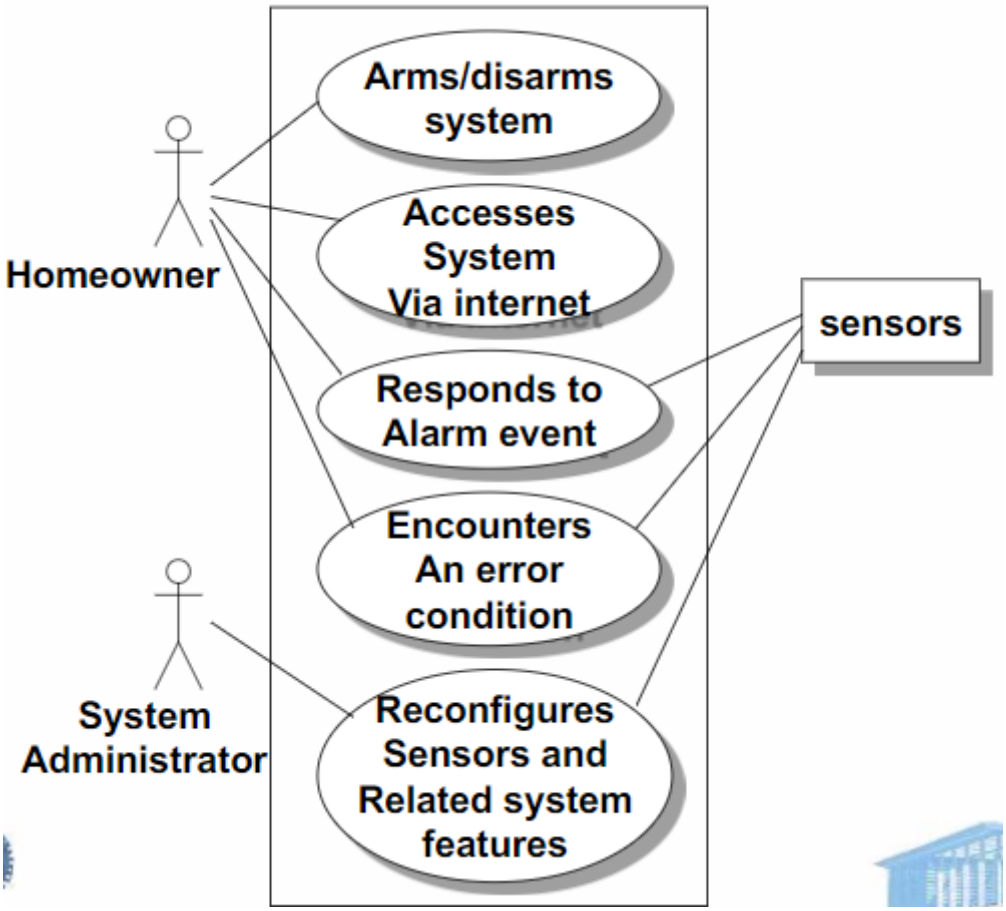
- 个人软件过程 PSP
    - 规划
    - 高级设计
    - 高级设计的审查
    - 开发
    - 事后维护
- 团队软件过程 TSP
    - 每个项目都是使用“脚本”“启动”的，该脚本定义了要完成的任务
    - 团队是自我指导的
    - 鼓励进行度量
    - 分析度量是为了改善团队流程

第二部分：需求分析和软件设计

8. 理解需求(很重要，有一堆图)

- 需求工程：分为如下几个部分
    - 初始–提出一系列问题以建立
        - 对问题的基本理解–想要解决方案的人
        - 所需解决方案的性质，以及
        - 客户与开发人员之间初步沟通和协作的有效性
    - 启发-从所有利益相关者那里获得需求
    - 精心设计-创建一个可识别数据，功能和行为要求的分析模型
    - 谈判-达成对开发人员和客户切合实际的可交付系统
    - 规范-可以是以下各项中的任何一个（或多个）
        - 书面文件
        - 模型集
        - 形式化的数学符号
        - 用户场景（用例）的集合
        - 原型
    - 验证–寻找审查机制 validation
        - 内容或解释错误
        - 可能需要澄清的区域
        - 信息缺失
        - 不一致（设计大型产品或系统时的主要问题）
        - 矛盾或不现实（无法实现）的需求。
    - 需求管理
- 质量函数的部署
    - 函数部署：确定系统需要用到的每个函数的值
    - 信息部署：明确数据对象和事件
    - 任务部署：明确系统的各个表现
    - 值分析：确定各个需求的优先级
    - 确定三种需求：普通需求，期望的需求和令人excited的需求
- 非功能性的需求

    - 质量属性，性能属性，安全性属性或常规系统约束
    - 用两个阶段来确定哪些非功能性需求是兼容的
        - 第一阶段是简历非功能性需求位列，系统SE guidelines为行的矩阵
        - 第二阶段是团队对非功能性的需求进行优先级排序，解决冲突
- 用例use-case
    - 包含了一系列描述系统使用的线程的用户场景
    - 每个场景都是从“参与者”的角度来描述的，参与者是通过某种方式与软件交互的**人或设备**
    - **用例图**：参与者+事件——连线

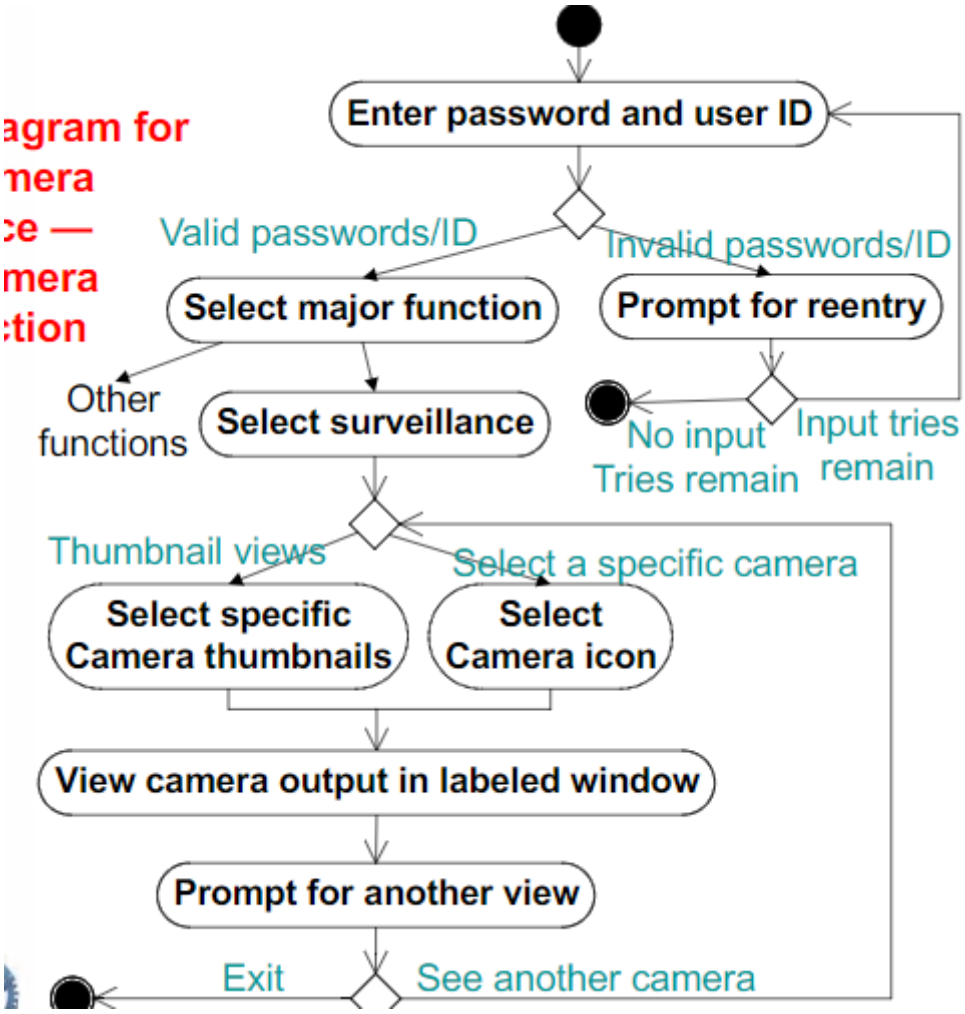


- 分析模型中几种很常见的图
    - use-case图：上面提过了，参与者+事件+连线
    - 类图：定义一个类和类内的变量/方法
    - **状态图**：由单个状态图组合而成一个大的状态图
        - 表示系统中各种操作对应的过程和状态
        - 单个状态图包含如下内容
            - 状态的名称
            - 状态的变量：system status表示系统的状态，可以用Display给各个变量赋值
            - 状态的事件：表明该状态下的各种事件，用entry表示进入的状态，用do表示这个状态下需要做的事情

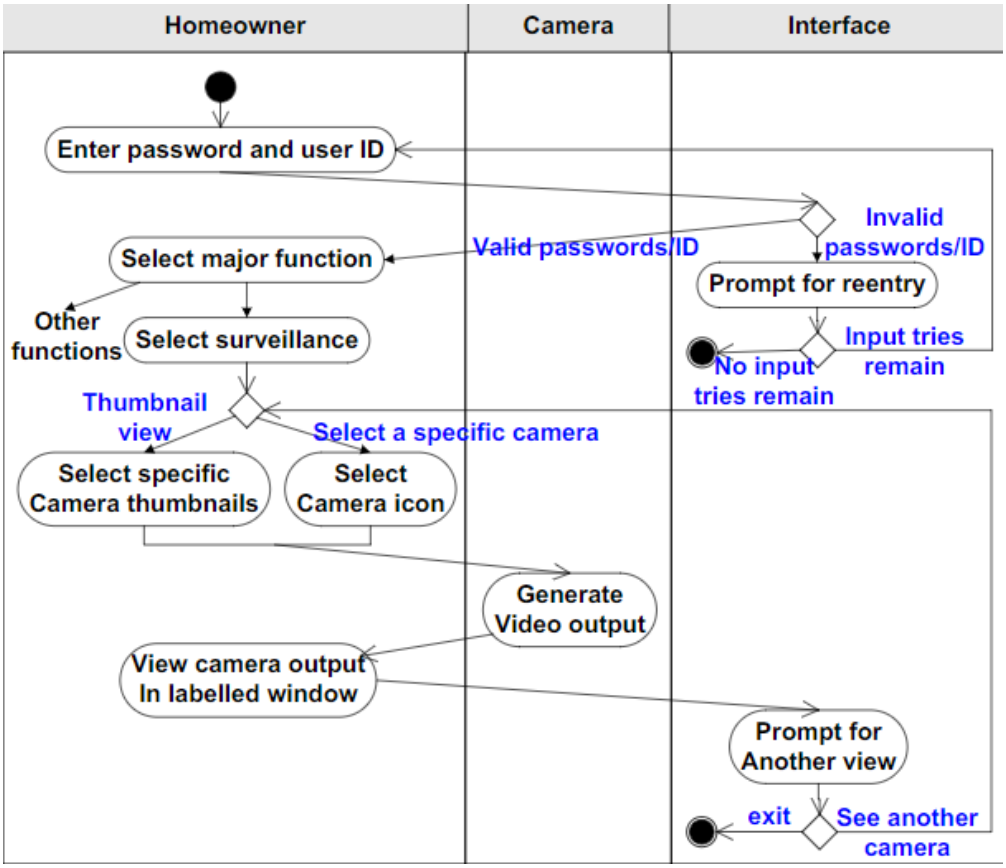
9. 需求建模：基于场景的方法

- 需求分析可以让软件工程师：
    - 详细阐述在早期需求工程任务中建立的基本需求
    - 建立描述用户场景，功能活动，问题类别及其关系，系统和类别行为，转换后的数据流以及软件必须满足的约束的模型
- 需求建模的经验法则

    - 抽象级别应该要相对较高
    - 分析模型的每个元素都应加深对软件需求的总体了解，并提供对系统的信息域，功能和行为的洞察力
    - 在设计之前对基础架构和其他非功能模型进行延迟考虑
    - 最小化系统的耦合性(coupling)
    - 模型尽可能简单有价值
- 域分析：目标是识别，分析和指定来自特定应用程序领域的通用需求，通常会被用于同类型项目中进行复用
- 软件需求建模的方法
    - 基于场景的建模
        - 用例，用例图
        - 事件图：通过提供过程流的图形表示来补充用例
            - 类似于算法的流程图，详细叙述了一个用户使用过程中会发生的各种事件和各种选择对应的结果



        - 泳道图：表示用例描述的活动流，并同时指示哪个参与者（如果在特定用例中涉及多个参与者）或分析类负责由活动矩形描述的动作
            - 在流程图的基础上划分了参与者和系统的接口，看起来就像泳池中的泳道



- 面向流的建模
  - 数据流图
  - 控制流图
  - 过程中叙述
- 基于类的建模
  - 类图
  - 分析包
  - CRC模型
  - 协作图
- 基于表现的模型
  - 状态图
  - 线型图

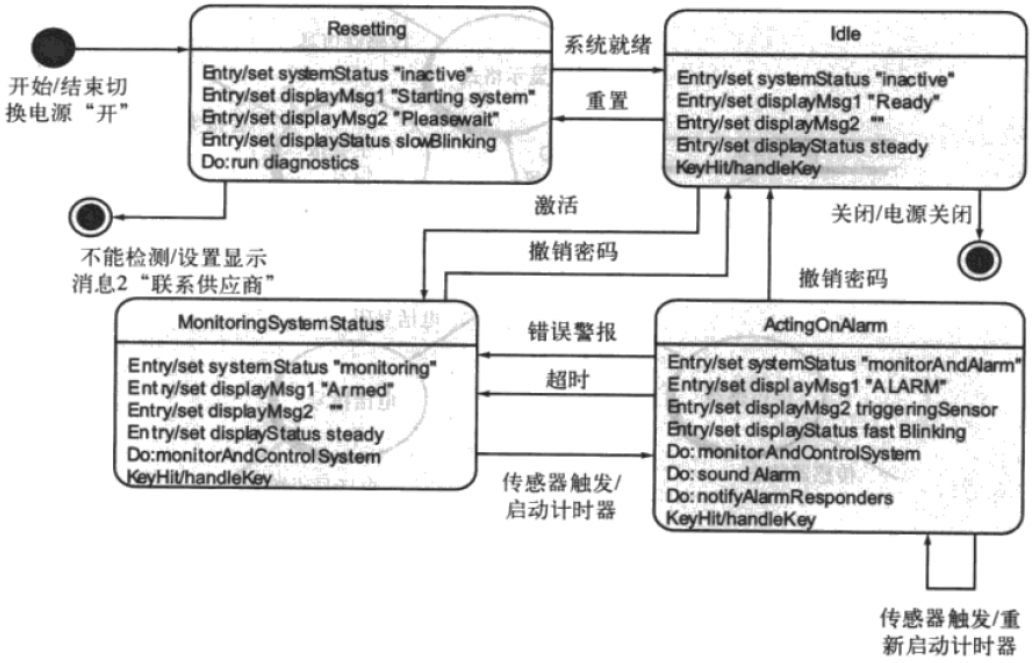
10. 基于类的建模

- 基于类的建模需要需要包含以下内容：
  - 系统需要操作的各个对象
  - 每个对象上面可以执行的各类操作
  - 对象之间的关系
  - 类之间的合作关系
- CRC建模
  - “责任”是类封装的属性和方法
  - “合作者”是需要提供给其他类信息来完成“责任”的类
  - 类的种类：
    - 实体类：也称为模型或业务类，直接从问题陈述中提取
    - 边界类：用来创造用户接口
    - 控制类：用来控制一些特定的工作，包括
      - 类和对象的创建

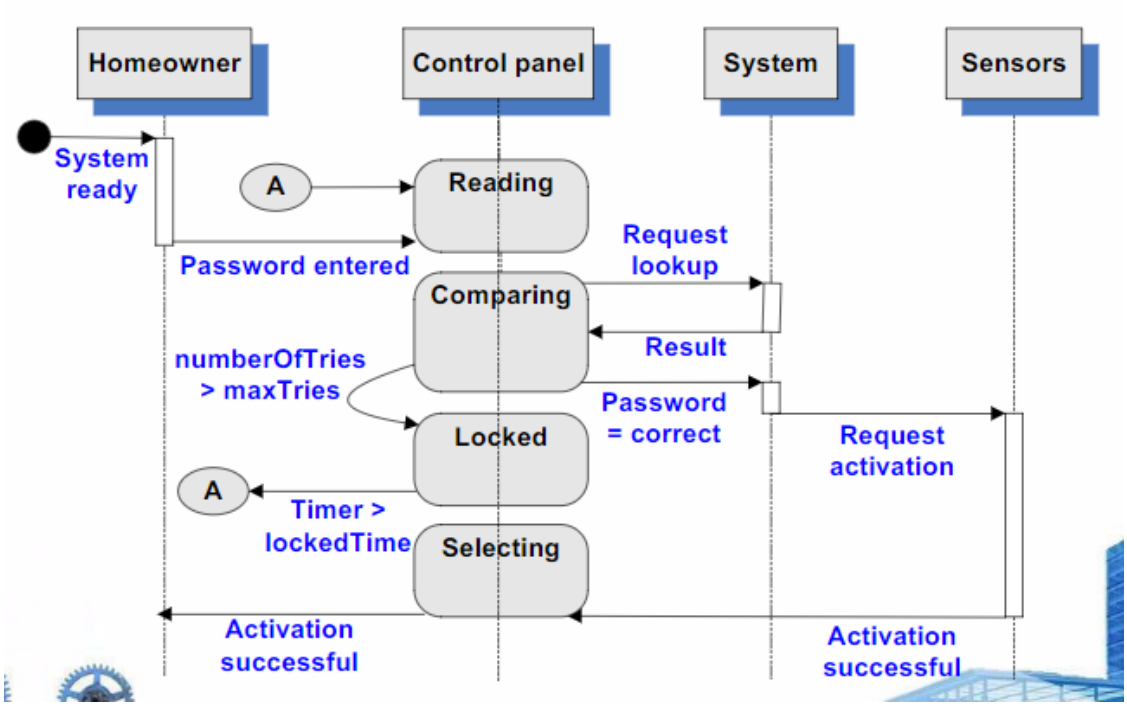
        - 边界类的实例化和信息交互
        - 对象之间的复杂通信
        - 用户和应用之间的数据交流和识别
    - **怎么画CRC卡？**
        - CRC卡是一个标准索引卡的集合，每一张卡片表示一个类
        - 类名写在上方，类的责任写在左侧，合作者写在右侧
        - 用描述性的语言写出类的所有责任，合作者写别的具体的类

11. 基于行为模式的建模

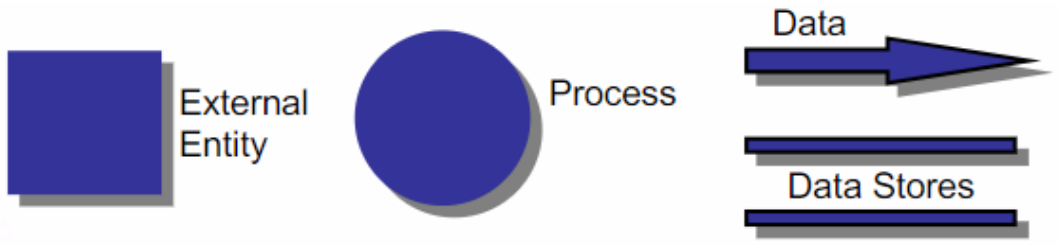
- 行为模型表明软件将如何响应外部事件或刺激。要创建模型，分析人员必须执行以下步骤：
    - 评估所有的用例，充分了解交互的顺序
    - 确定驱动交互序列的事件
    - 为每个用例创建一个序列
    - 为每个用例建立状态图
    - 查看行为模型验证一致性和正确性
- UML：类状态图的绘制
    - 描述系统的状态和导致状态间转换的各种事件，可以有外部输入的一些信息
    - 用一个黑点表示状态的开始，用一个有环的黑点表示状态结束
    - 圆角矩形表示系统的各个状态，每个状态可以包含对执行动作的一个简要描述(写在do后面
    - 带标签的箭头表示驱动从一个状态转换到另一个状态的过程



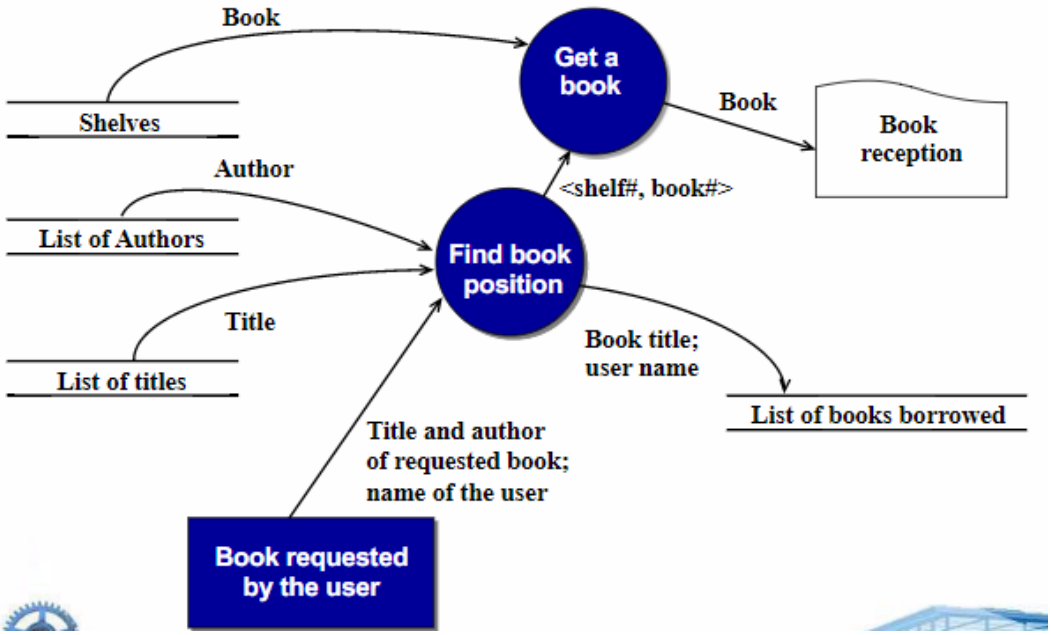
    - 线性图的画法



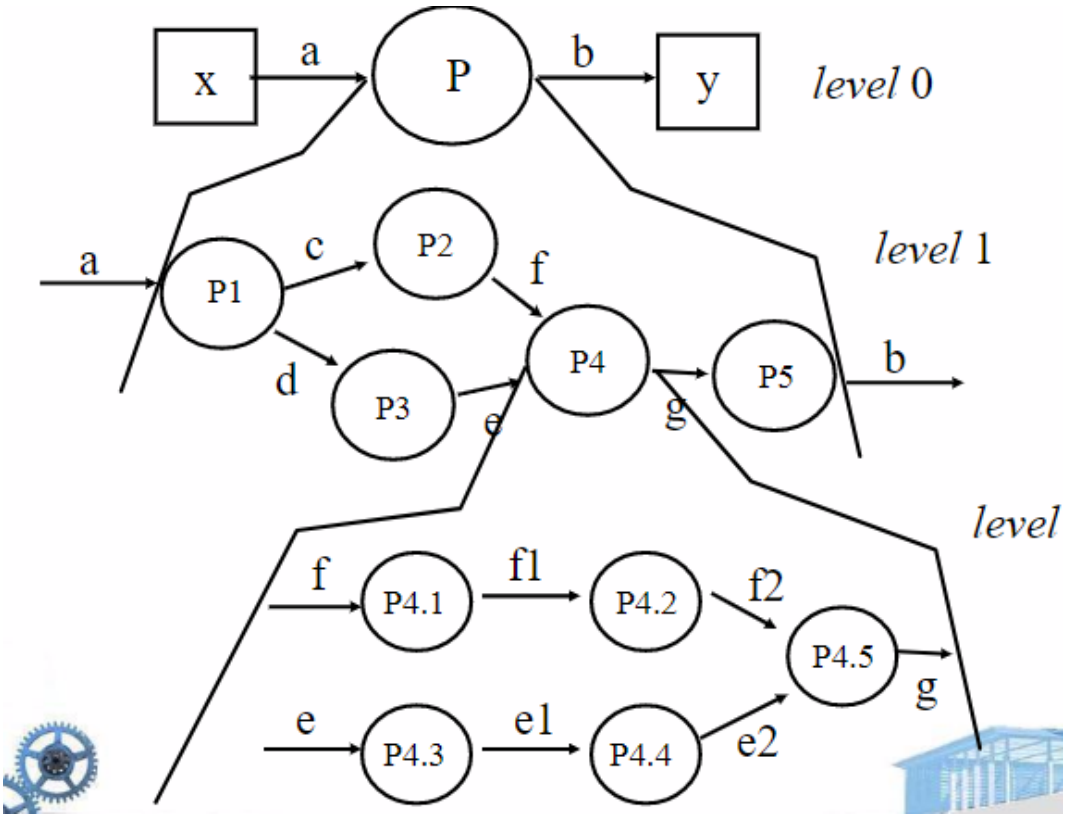
- 面向流的建模
  - 数据流图中出现的各类元素



  - 具体的画法：
    - 外部实体衍生出一系列的过程，存储的数据在过程中流向各个过程



    - 数据流的架构



- Web应用的建模
  - 内容分析
  - 交互分析
  - 功能缝隙
  - 配置分析
    - 配置模型：分为服务端和客户端
  - 导航分析

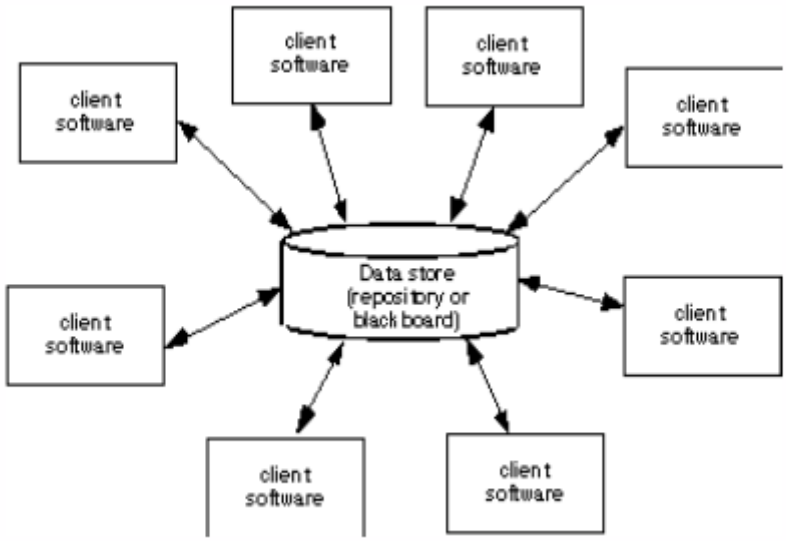
12. 设计理念

- 好的软件设计需要包含：Firmness牢固度，Commodity，Delight
- 软件工程的设计
  - 数据和类的设计
  - 架构设计
  - 接口设计
  - 组件级别的设计
- 质量准则设计应展示一种架构，该架构是
  - 使用可识别的建筑风格或样式创建的；具有良好设计特性的组件组成；可以以进化的方式实施
  - 设计应模块化也就是说，软件应在逻辑上划分为元素或子系统。
  - 设计应包含数据，体系结构，接口和组件的不同表示形式。
  - 设计应导致适合于要实现的类并绘制的数据结构来自可识别的数据模式。
  - 设计应导致组件表现出独立的功能特征。
  - 设计应导致接口降低组件之间以及与外部环境之间连接的复杂性。
  - 设计应使用可重复的方法得出取决于软件需求分析过程中获得的信息。
  - 设计应使用能有效传达其含义的符号来表示。
- 基本的设计理念
  - 抽象：数据抽象，过程抽象，控制抽象
  - 架构：软件的整体结构
    - 结构特性

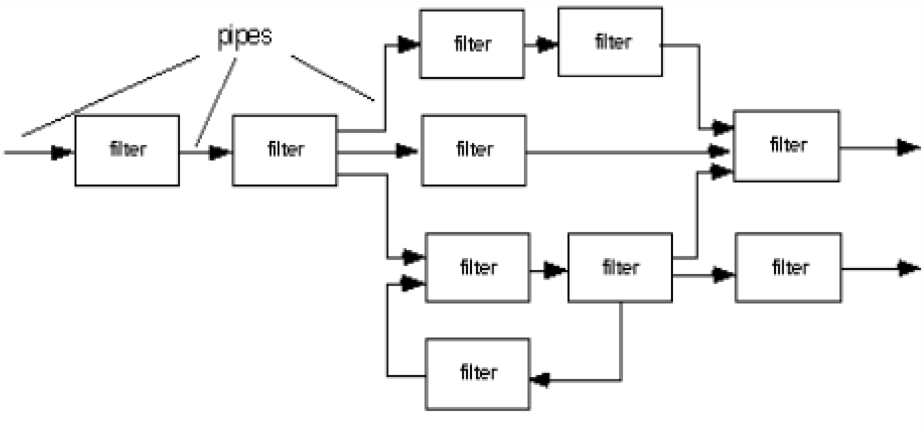
        - 附加功能属性
        - 相关系统
    - 模式：设计各类模板
    - 关注点的分离
        - 将复杂的问题细分为若干可以独立解决的子问题
    - 模块化：将软件模块化来降低成本
    - 隐藏
    - 功能独立：通过开发功能专一而不和别的模块过度互动的模块来实现
        - cohesion 内聚力
        - coupling 耦合度
    - 细化
    - aspect
    - 重构
    - 面向对象设计：继承封装多态的特性
    - 设计类别特征
        - 完整性
        - 原始性
        - 高凝聚力
        - 低耦合度

13. 架构设计

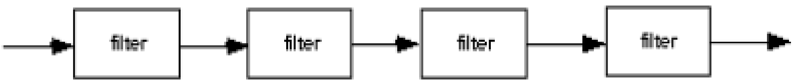
- 架构的作用
    - 分析设计是否满足其规定要求
    - 在进行设计更改仍然相对容易的阶段考虑架构替代方案
    - 降低开发过程中的风险
- 架构图的种类
    - 数据为中心的架构



    - 数据流架构

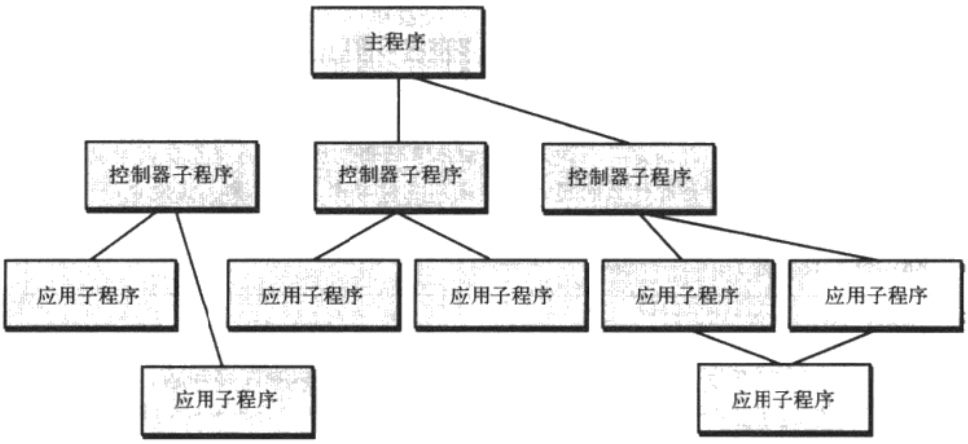


(a) pipes and filters

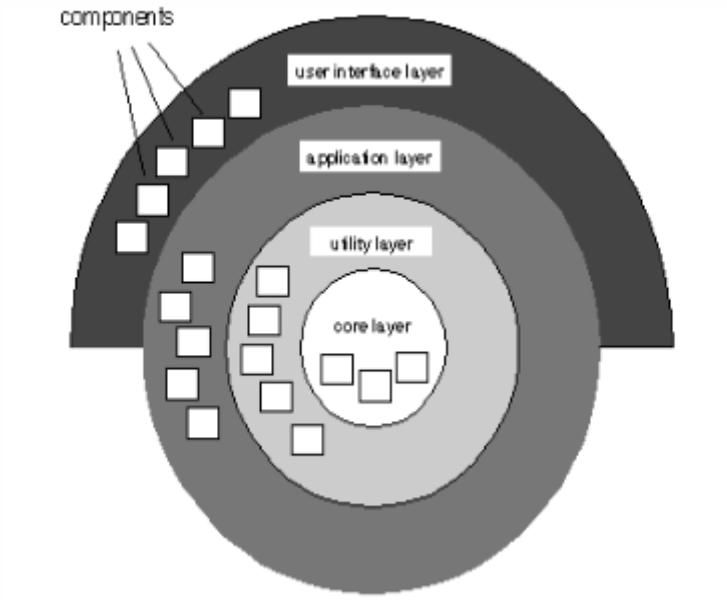


(b) batch sequential

- 调用和返回架构
  - 可以设计出一个相对易于修改和扩展的程序结构
  - 主程序/子程序架构：将功能分解为一个控制层次，主程序调用一系列程序构建，程序构件又去调用其他的构件



- 面向对象的架构
- 分层的架构



- 架构需要考虑的因素
  - 经济性
  - 能见度
  - 间距：是否划分的足够合理
  - 对称性
  - 紧急情况处理
- 架构的复杂性
  - 主要由组件和架构之间的依赖性决定
  - 共享依赖关系：表示使用相同资源的消费者或为相同消费者生产的生产者之间的依赖关系
  - 流依赖关系：表示资源生产者和消费者之间的依赖关系
  - 约束依赖关系：关系表示一组活动之间相对控制流的约束

14. 组件设计

- 设计准则
  - 组件–应该为在架构模型中指定的组件建立命名约定，然后在组件级模型的一部分中进行完善和细化
  - 接口-接口提供有关通信和协作的重要信息（以及帮助我们实现OPC）
  - 依赖关系和继承–从左到右建模依赖关系，从底部（派生类）到顶部（基类）建模继承
- 组件设计的各类原则：
  - The Open-Closed Principle (OCP)
  - The Liskov Substitution Principle (LSP)
  - Dependency Inversion Principle (DIP)
  - The Interface Segregation Principle (ISP)
  - The Release Reuse Equivalency Principle (REP)
  - The Common Closure Principle (CCP)
  - The Common Reuse Principle (CRP)
- 各类基于组件的软件开发
  - 几种标准
    - OMG/CORBA
    - Microsoft COM
    - Sun JavaBeans
  - 几个过程
    - Component qualification
    - Component adaptation

        - Component composition
        - Component update
        - CBSE Activities
- 可复用环境 Reuse Environment
    - 组件数据库能够存储软件组件和检索它们所需的分类信息
    - 一种软件组件检索系统（例如，对象请求代理），使客户端应用程序能够从库服务器中检索组件和服务
    - 支持将重用组件集成到新设计或实现中的CBSE工具

15. 用户接口设计

- 三条golden rules
    - Place the user in control
    - Reduce the user's memory load
        - Reduce demand on short-term memory.
        - Establish meaningful defaults.
        - Define shortcuts that are intuitive.
        - The visual layout of the interface should be based on a real world metaphor.
        - Disclose information in a progressive fashion.
    - Make the interface consistent
- 用户接口设计模型
    - 用户模型：系统所有最终用户的配置文件
    - 设计模型：用户模型的设计实现
    - 心理模型：用户心中想象接口的画面（？
    - 实现模型：界面“外观”以及描述界面语法和语义的支持信息
- 接口设计实现
    - 设计的过程：
        - 分析建模，设计，构建，验证的循环往复
    - 接口分析：需要搞清楚以下内容
        - 谁来用——用户分析：分析一堆
        - 接口需要完成的任务
        - 接口上展现的内容：显示的文本之类的东西
        - 接口的工作环境
    - **有效的APP/Web接口设计**
        - 视觉上明显且forgiving
        - 使用户不必担心系统的内部工作原理
        - 最大程度地完成工作

17. web应用设计

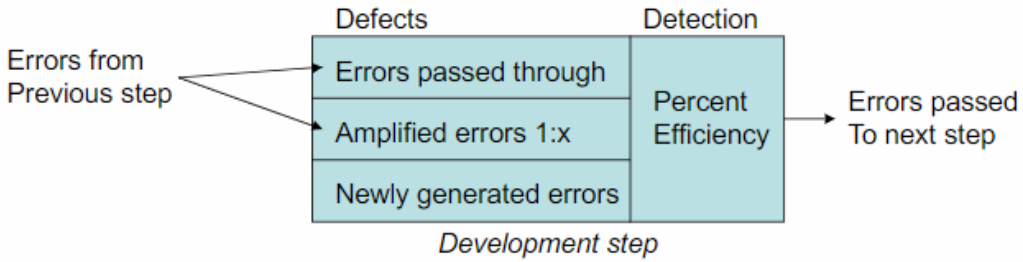
- web应用开发注重的：安全性，可用性，扩展性，上市时间
- 对于用户而言注重的内容：时效性，结构性，内容优质度，精确性和一致性，响应时间和延迟，表现
- web应用设计的目标：同一性，鲁棒性，适航性，视觉吸引力，兼容性
    - 美术设计：各种奇怪的要求，肯定记不住，看直觉
    - 内容设计：content-object 表示实例化彼此之间的关系所需的机制
    - 架构设计：内容的架构，整个应用的架构
        - MVC 架构：
            - model 包含内容对象和运行逻辑
            - view 是展示给外部的视图和接口

        - controller 控制model和view之间的数据传输和连接
    - **NSU(navigation semantic units)导航语义单元**
        - 是一系列的信息和导航结构，用于满足一部分相关用户的需求
        - 由navigation node和navigation links连接而成
        - 看了一张例图，感觉就是描述用户进入一个网站之后不同的操作会进入什么样的页面和使用什么不同的功能，以及这些操作之间的各种复杂的关系
        - 比较像“地图”，所以叫做navigation
        - 构成要素
            - Individual navigation link 有文字说明的线，表示网页中的链接，图标，按钮等引起网页变化的控键
            - Horizontal navigation bar水平导航栏， 写出了网页的合新功能和内容
            - Vertical navigation column垂直导航栏，列出了主要内容和功能，列出了web应用的所有内容对象
            - tabs 将内容或功能类别表示为需要链接时选择的选项卡
            - Site maps 提供了一个包含所有内容的选项卡，用于导航到WebApp中包含的所有内容对象和功能

第三部分：软件的审核和测试

20. 项目审核

- 审核的内容：软件系统存在的错误和缺陷
    - 错误是软件发布之前就存在的问题，也就是能直接找出来的bug
    - 缺陷是软件发布之后才发现的质量问题
    - 分为正式的审核(FTR)和非正式的审核
        - SDRs 案例驱动的审核
- 缺陷放大模型
    - 图示



    - 假设在设计过程中发现错误，将花费1.5单位来纠正。相对于此成本，测试开始之前发现的相同错误将花费6.5个单位；在测试中，15个单位；发行后介于67到100个单位之间，也就是发现的越早，纠正错误需要的成本越小
    - 设计活动会导致软件过程中所有错误的50%-65%。然而，事实证明，正式的评审技术可以有效地发现设计缺陷的75%
- 审核效果的评估和计算
    - $E_{review} = E_p + E_a + E_r$
        - Preparation effort 准备时花费的努力
        - Assessment effort 评估时的努力
        - Rework effort 重新工作的努力
    - $Err_{total} = Err_{minor} + Err_{major}$
        - 主要错误+次要错误

- 缺陷密度 Defect Density = Err_{total}/WPS 表示每个单位的产品找到的错误个数
    - WPS 工作产品的size

22. 软件测试的策略

- 基本的测试策略
    - 从小的开始测试到大的
    - 对于常规的软件：一开始的焦点是模块，然后是模块的集成
    - 对于面向对象的软件：一开始的焦点的面向对象类中的包含的各个属性和操作
    - 测试中要注意的问题：都是一些很正确的废话
- 不同级别的测试
    - 单元测试
    - 集成测试
    - 验证测试
    - 系统测试
- 单元测试
    - 用测试样本去测试待测试的模块，需要测试的内容包括：
        - 接口
        - 局部的数据结构
        - 边界情况
        - 独立路径
        - 错误处理路径
- 集成测试策略
    - 选择：
        - big bang方法
        - 增量构建的策略
    - 集成的种类：
        - 自顶向下的集成
        - 自底向上的集成
    - 回归测试
        - 对已经执行的某些测试子集的重新执行，确保所做的更改不会传播意料之外的副作用
        - 每当更改软件的时候，软件的某些配置就会被修改
        - 有助于防止对软件的更改引起以外的行为或者其他错误
        - 回归测试可以通过重新执行所有测试用例的自己或者使用自动捕捉/回放的相关工具来手动进行
    - 烟雾测试
        - 已转换为代码的软件组件将集成到“内部版本”中
        - 设计一系列测试来暴露错误，这些错误将使构建无法正确执行其功能
        - 该版本与其他版本集成在一起，整个产品每天都要测试
- 通用测试标准
    - 接口的完整性：内部和外部的接口模块都需要测试
    - 功能的有效性：寻找功能缺陷
    - 内容的正确性：局部数据结构和变量的正确性
    - 表现：验证特定性能范围是否已测试
- 面向对象的测试
    - 从评估分析模型和设计模型的正确性/一致性开始
    - 产生的一些改变
        - 单元测试的概念由于面向对象的封装而扩大了
        - 集成测试关注类及其在“线程”中或在使用场景下的执行
        - 验证使用常规的黑盒方法

    - 测试CRC模型
        - 重新访问CRC模型和对象关系模型。
        - 检查每个CRC索引卡的描述，以确定委托的责任是否属于协作者的定义。
        - 反转连接以确保每个被要求提供服务的协作者都从合理的来源接收请求。
        - 使用步骤3中检查的反向连接，确定是否可能需要其他类别，或者在这些类别之间是否适当地划分了责任。
        - 确定是否可以将广泛要求的职责合并为一个职责。
        - 将步骤1到5迭代地应用于每个类，并通过分析模型的每次演变进行
    - 类测试和单元测试等价
        - 测试了类中的操作
        - 检查了类的状态行为
    - 集成应用了三种不同的策略
        - 基于线程的测试
        - 基于用例的测试
        - 集群测试
- Web应用的测试
    - 用户体验测试–确保应用程序符合利益相关者的可用性和可访问性期望
    - 设备兼容性测试–在多个设备上进行测试
    - 性能测试–测试非功能性要求
    - 连通性测试–应用能够可靠连接的测试能力
    - 安全测试–确保应用符合利益相关者的安全期望
    - 野外测试–在实际用户环境中的用户设备上测试应用程序
    - 认证测试–应用程序符合发行标准
- 狗屁不通讲了半天还是要debug

23. 常规应用的测试

- 可以测试的内容：可操作性，可观察性，可控制性，简单性，稳定性，可理解性
    - 外部测试和内部视角测试
- 测试数据的设计：bug经常出现在边界条件上
    - 详尽的测试 Exhaustive Testing
    - 选择性的测试：Selective Testing
    - 白盒测试：所有的情况都执行一次
        - 派生的测试用例
            - 以设计或代码为基础，绘制相应的流程图。
            - 确定所得流程图的圈复杂度。
            - 确定线性独立路径的基础集。
            - 准备测试用例，以强制执行基础中的每个路径组
        - 图矩阵：基于程序的流程图建立
        - 控制结构的测试
            - 情况测试：基于运行逻辑设计测试用例
            - 数据流测试：选择各种各样的测试路径
    - 黑盒测试：
        - 基于图的测试
        - 等价类的分离
        - 比较测试
        - 正交阵列测试：当输入参数的数量很少且每个参数可能取的值有明确界限时使用
        - 基于模型的测试：

24. 面向对象应用的测试

- 三件必须做的事
  - 测试的定义必须扩大到包括适用于面向对象的分析和设计模型的错误发现技术
  - 单元和集成测试的策略必须发生重大变化
  - 测试用例的设计必须考虑OO软件的独特特征
- 面向对象测试的策略
  - 单元测试
    - 单元更改的概念
    - 最小的可测试单元是封装的类
    - 单个操作不再可以孤立地进行测试（常规的单元测试视图），而是作为类的一部分进行测试
  - 集成测试
    - 基于线程的测试
    - 基于用例的测试
    - 集群测试
  - OOT方法
    - 基于故障的测试-测试人员查找可能的故障（即可能导致缺陷的系统实施方面）。为了确定这些故障是否存在，设计了测试用例以执行设计或代码。
    - 类测试和类层次结构–继承不会消除对所有派生类进行全面测试的需要。实际上，它实际上会使测试过程复杂化。
    - 基于场景的测试设计–基于场景的测试着重于用户的行为，而不是产品的行为。这意味着（通过用例）捕获用户必须执行的任务，然后将其及其变体作为测试应用
    - 随机测试：生成各种随机（但有效）的测试序列
    - 分离测试：减少测试类所需的测试用例数量，其方式与传统软件的等效分区相同
      - 基于状态的分离
      - 基于属性的分离
      - 基于类的分离
    - 表现测试

25. web应用的测试

- 新增的测试内容：
  - 数据库的测试
  - 用户接口测试
  - 易用性测试
  - 兼容性测试
  - 组件级别的设计
  - 导航测试
  - 配置测试：服务端和客户端
  - 安全性测试
  - 压力测试
  - 加载测试
  - 绩效测试

第四部分：软件项目的管理和部署

29. 软件的配置管理

- 第一准则：无论您在系统生命周期中处于何处，系统都将发生变化，并且在整个生命周期中始终存在着对其进行更改的渴望

    - 商业的变化
    - 技术的变化
    - 用户需求的变化
- SCM Repository
    - Repository Feature 仓库的功能
        - 版本控制
        - 依赖性跟踪和变更管理
        - 需求跟踪
        - 配置管理
        - 审计跟踪
    - SCM元素：组件，过程，构建，人
        - 过程中包含：版本控制，变更控制，标识，配置审计，报告
    - 对于web和手机应用：
        - 内容，人，可扩展性，Politics
    - 妈的讲的什么玩意儿

31. 项目管理

- 4个P：人，产品，过程，项目
- 利益相关者：高级经理，项目经理，实践者，顾客，终端使用者
- 软件开发团队
    - 团队领导：MOI模型——motivation，organization，ideas/innovation
    - 讲了一堆要怎么决定一个开发团队的原则，感觉接近于常识
    - 组成范式：
        - 封闭范式：用传统的结构组成一个团队
        - 随即范式：松散组成，依赖团队成员的个人主动性
        - 开放范式：试图半封闭半随机
        - 同步范式：依靠对问题的自然划分，并组织团队成员来解决问题，而彼此之间**很少进行主动沟通**
    - 避免团队的toxicity：主要讲了团队怎么处理各种失败，属于常识
    - 敏捷的团队
        - 互相信任
        - 技能的分配必须适合这个问题
        - 为了保持团队的凝聚力可能要将标新立异的人排除
        - 队伍是自我组织的
    - 团队的协作和交流
        - 正式的，非个人化的方法
        - 正式的人际关系程序
        - 非正式的人际程序
        - 电子通讯
        - 人际关系网络
- 一个项目的基本过程
    - 一个好的开始
    - 保持动力
    - 追踪进度
    - 做出好的决策
    - 事后分析

34. 项目日程安排

- 日程安排的原则

    - compartmentalization 分割成独立的小任务
    - interdependency 相互依存
    - effort validation
    - 规定每个人的责任
        - 定义一个task网络
    - 规定应该有的结果
    - 规定项目中的里程碑
- 面向对象项目的进展：几个技术的里程碑
    - OO分析完成
    - OO设计完成
    - OO编程完成
    - OO测试
- 获得价值分析(EVA)
    - 什么是获得价值(Earned Value)
        - 是进度的一种衡量标准
        - 使我们能够使用定量分析而不是凭直觉来评估项目的“完成度”
        - 可以将早至15%的绩效准确，可靠地读入项目
    - budgeted cost of work scheduled (BCWS) 计划的预算工作成本
    - budget at completion, BAC 完成预算，是预算工作成本之和
    - budgeted cost of work performed (BCWP) 某个时间点已经预期消耗的成本
    - Schedule performance index 进度绩效指数 SPI=BCWP/BCWS
    - Schedule variance 进度差异：SV=BCWP-BCWS
    - Percent scheduled for completion 计划完成的百分比 BCWS/BAC
    - 完成百分比 = BCWP/BAC
    - 实际工作花费 ACWP

35. 风险分析

- 主动和被动的风险管理
    - 被动(Reactive) 风险管理
        - 项目团队在风险发生时对风险做出反应
        - 分为缓解，修复故障，危机管理等
    - 主动的风险管理
        - 进行正规的风险分析
        - 组织纠正风险的根本原因
    - 风险管理的七条基本原则
        - 保持全局的视野
        - 做出前瞻性的看法
        - 鼓励开放交流
        - 整合
        - 重视过程的持续性
        - 开发一个共享的产品版本
        - 鼓励团队合作
- 风险的分类
    - 产品的大小
    - 业务的影响
    - 客户的特征
    - 流程的定义

    - 开发环境
    - 所用技术栈
    - 人员规模和经验
- 风险的评估
    - 风险的组成要素
        - 绩效风险
        - 成本风险
        - 日程进度风险
        - 支持风险
    - 风险投影：评估风险发生的可能性
    - 风险表：风险内容+概率+可能发生的影响+RMMM
        - 影响用1-5的大小评估
    - RMMM：Risk Mitigation Monitoring& Management
        - mitigation：怎么样才能避免这个风险
        - monitoring：可以跟踪哪些因素，从而使我们能够确定风险的可能性越来越大
        - management：如果风险变为现实，我们有什么应急计划
    - RE = P * C 概率乘以成本