软件发展趋势

- 软件项目规模日益扩大
- 软件在整个系统中的比重日益增加

软件的四大本质特征

- 复杂性
- 不可见性
- 易变性
- 非一致性

软件危机

- 落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题

软件工程

- 研究用工程化方法构建和维护有效的、实用的和高质量的软件
- 将系统、规范、可度量的方法应用于软件的开发、运行、维护，及对这些问题的研究

软件项目管理

- **三大关键要素**
    - 目标
    - 状态
    - 纠偏
- **三大目标**
    - 成本
    - 质量
    - 工期
- **定义**
    - 是应用方法、工具、技术以及人员能力来完成软件项目，实现项目目标的过程
- **内容**
    - 估算、计划、跟踪、风险管理、项目管理、人员管理、沟通管理等

软件过程

- **定义**
  - 软件过程是为了实现一个或多个事先定义的目标而建立起来的一组实践的集合，这组实践之间往往有一定的先后顺序，作为一个整体来实现事先定义的一个或者多个目标
- **生命周期模型**
  - 对软件过程的一种人为的粗粒度划分，往往不包括技术实践
- **理论基石**
  - 软件产品和服务的质量，很大程度上取决于生产和维护该软件或者服务的过程的质量
- **广义软件过程 ≈ 软件开发方法（软件开发过程）**
  - 包括狭义软件过程和技术、人员
  - 举例：SCRUM方法是软件开发方法，是广义的软件过程
- **生命周期模型**
  - 生命周期模型是对一个软件开发过程的人为划分，是软件开发过程的主框架

- CMM/CMMI 不是软件开发方法，不是项目管理方法

软件过程管理

- **对象**
  - 管理软件过程
- **目的**
  - 让软件过程在开发效率、质量等方面有着更好性能绩效

- 以下说法是否正确？为什么？
  - A) 软件过程管理是软件项目管理应该要实现目标。
  - B) “在公司导入敏捷过程是我们今年过程改进的主要目标。”
  - C) XP与CMM/CMMI是对立的两种软件开发方法。
  - D) CMM/CMMI不适合当今互联网环境的项目管理需求。
  - E) PDCA和IDEAL不适合在敏捷环境中使用。
  - F) 不同的软件开发过程应该使用不同的生命周期模型，反之亦如此。

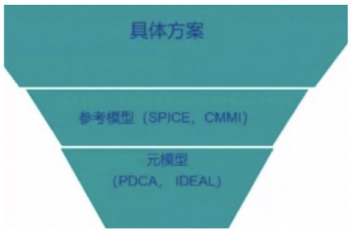
A：过程管理和项目管理完全是两码事，项目过程管理应当有专人去做（说法错误）
B：在概念上来说是合适的，但在操作的时候可能存在问题（说法正确）
C：XP极限编程。CMM/CMMI不是软件开发方法！（说法错误）

- 当然，XP的观念确实与严格的CMM/CMMI是对立的
- XP是软件开发方法，敏捷的
- 但也有人认为XP和CMM/CMMI都是严格的

D：用于过程改造，从这方面说，该方法是正确的（说法正确）

- 担心的情况：部分同学关注的重点可能放在“互联网”上，认为该时代应该使用敏捷的方法，从而认为D为正确，但这样理解是不对的
- 再次强调：CMM/CMMI不是项目管理方法！

E：使用（包括指导等方面），PDCA和IDEAL在任何环境中都可以使用，因为是过程管理中的原模型（说法错误）



- 只要是能用过程去描述的事情，就可以使用PDCA

F：生命周期模型是“人为划分”的需要（粗粒度的划分），为了传递某种观念。从软件开发的过程来看，就算是一样的开发过程，也有可能被划分成不同的生命周期（说法错误）

- 比如说，详细设计，有时被划分成实现阶段，有时不是
- 生命周期模型：像一个框架Scheme

软件过程发展史

- **软硬件一体化（50-70）**
  - **软件完全依附于硬件**
    - 应用特征
      - 软件支持硬件完成计算任务
      - 功能单一
      - 复杂度有限
      - 几乎不需要需求变更
    - 开发特征
      - 硬件太贵
      - 团队以硬件工程师和数学家为主
    - 典型过程和实践
      - SAGE
      - measure twice，cut once
  - **软件作坊**
    - 应用特征
      - 功能简单
      - 规模小
    - 开发特征
      - 非专业领域人员参与
      - 高级程序语言出现
      - 质疑权威文化盛行
    - 典型过程和实践
      - code and fix

- **软件成为独立的产品（70-90）**
    - 应用特征
        - 摆脱硬件束缚
        - 功能强大
        - 规模和复杂度剧增
        - 个人电脑出现
        - 来自市场的压力
    - 典型过程和实践
        - 形式化方法
        - 结构化程序设计和瀑布模型
    - 问题和不足
        - 形式化方法在扩展性和可用性方面存在不足
        - 瀑布模型成为一个重文档，慢节奏的过程
    - 成熟度模型
        - CMM
        - CMMI
- **网络化和服务化（90--）**
    - 应用特征
        - 功能更复杂、规模更大
        - 用户数量急剧增加
        - 快速演化和需求不确定
        - 分发方式的变化
    - 典型过程和实践
        - 迭代式：大型软件过程的开发过程也是一个逐步学习和交流的过程，软件系统的交付不是一次完成，而是通过多个迭代周期，逐步来完成交付
        - 敏捷开发
            - 原则
                - 个体和互动胜过流程和工具
                - 可以工作的软件胜过详尽的文档
                - 客户合作胜过合同谈判
                - 响应变化胜过遵循计划
        - 开源软件开发方法
            - 基于并行开发模式的软件开发的组织与管理方式
            - Linus定律
- **当前**
    - 应用特征
        - 进一步服务化和网络化
        - 用户需求多样性进一步凸显
        - 软件产品和服务的地位变化
        - 错综复杂的部署环境
        - 近乎苛刻的用户期望
    - 开发特征
        - 空前强大的开发和部署环境
        - 盛行共享和开源
        - 潜在支撑获得了长足进步
    - 典型实践和方法

        - DevOps

团队动力学

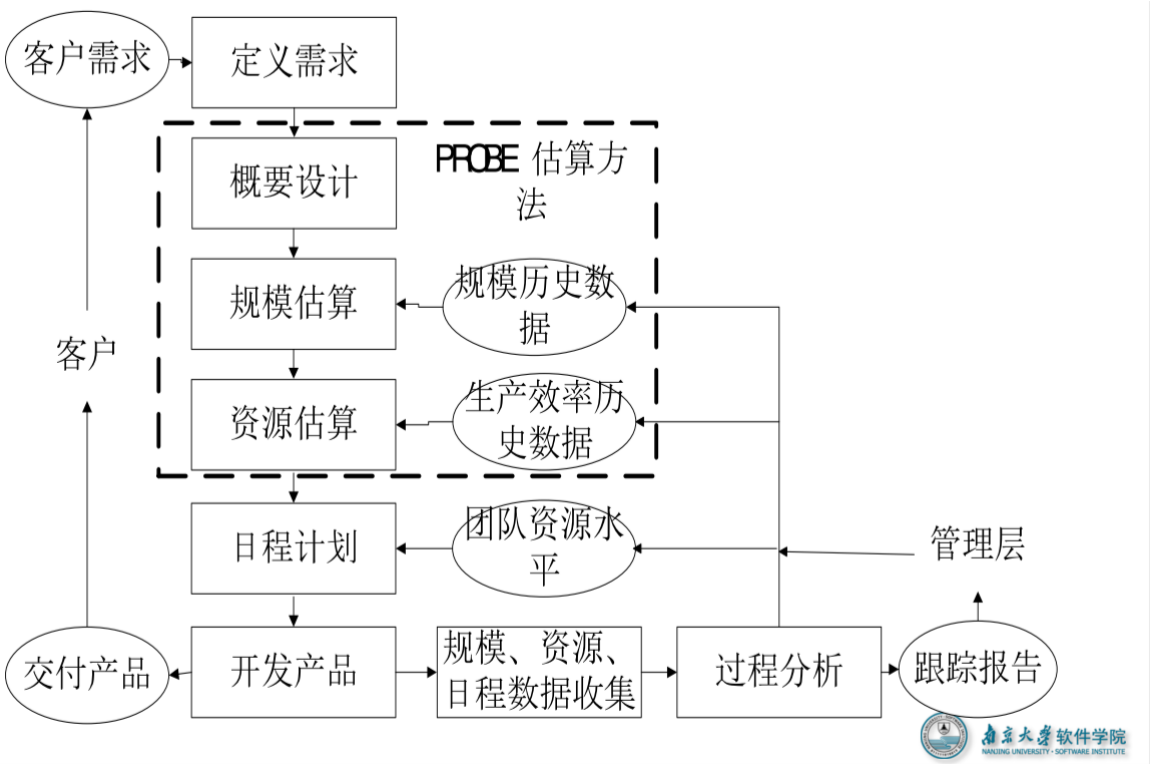
项目估算

- **目的**
    - 给各类计划提供决策依据
- **对象**
    - 规模
    - 时间
    - 日程
- **PROBE**
    - PROxy Based Estimation（基于代理的度量）
    - 缓解精确度量和早期规划之间的矛盾
    - 流程
        - 概要设计
            - 与已有产品/组件相关联
            - 定义能够产生期望功能的产品元素
            - 估算计划构造之物的规模
        - 代理识别
        - 估算并调整程序规模
            - 线性回归
        - 估算并调整资源
            - 线性回归
        - 计算预测区间
            - 分布
- 估算要点一：尽可能划分详细
    - 多个部件的总误差会比各个部件的误差和小
        - 误差趋于抵消
        - 假设没有共同的误差
- **SCRUM估算**
    - 度量一个Story需要付出的工作量
        - 工作量：抽象的，混合了对于开发特性所要付出的努力、开发复杂度、个中风险以及类似东西
        - 估算：
            - 设定标准，并考虑其它特性与标准之间的相对大小关系，单位为Story point
- 估算要点二：建立对结果的信心
- 估算要点三：依赖数据
- 估算要点四：估算要的是过程，而非结果

- **历史数据的处理**
    - 简单方法：均分点
    - 正态分布：均值±标准差点
    - 对数正态分布：ln均值±标准差点
- **估算质量**
    - 相关性：历史数据相关性r≥0.7
    - 显著性：历史数据显著性s≤0.05
    - 极端数据

项目计划

- **工作分解结构**
    - 定义
        - 以可交付成果为导向的对满足项目目标和开发叫副产物的项目相关工作进行的分解
    - 作用
        - 范围基线
        - 提供整体观
        - 不遗漏可交付物
        - 明确各个角色的责任
        - 工作包定义
        - 估算和计划的基础
        - 理解工作，分析风险
    - 创建
        - 识别和分析可交付成果及相关工作
        - 确定工作分解结构的结构与编排方法
        - 自上而下逐层细化分解
        - 为工作分解结构组成部分制定和分配标志编码
        - 核实工作分解的程度是必要且充分的
- **范围管理**
    - 定义
        - 包括确保项目做且只成功完成项目所需的全部工作的各过程
    - 以WBS为基准
        - 收集需求
        - 定义范围
        - 创建WBS
        - 核实范围
        - 控制范围变更
- **开发策略与计划**
    - 在产品组件需求基础之上，明确每个产品组件的获得方式与顺序，从而在项目团队内部建立起大家都理解的产品开发策略
- **通用计划框架**

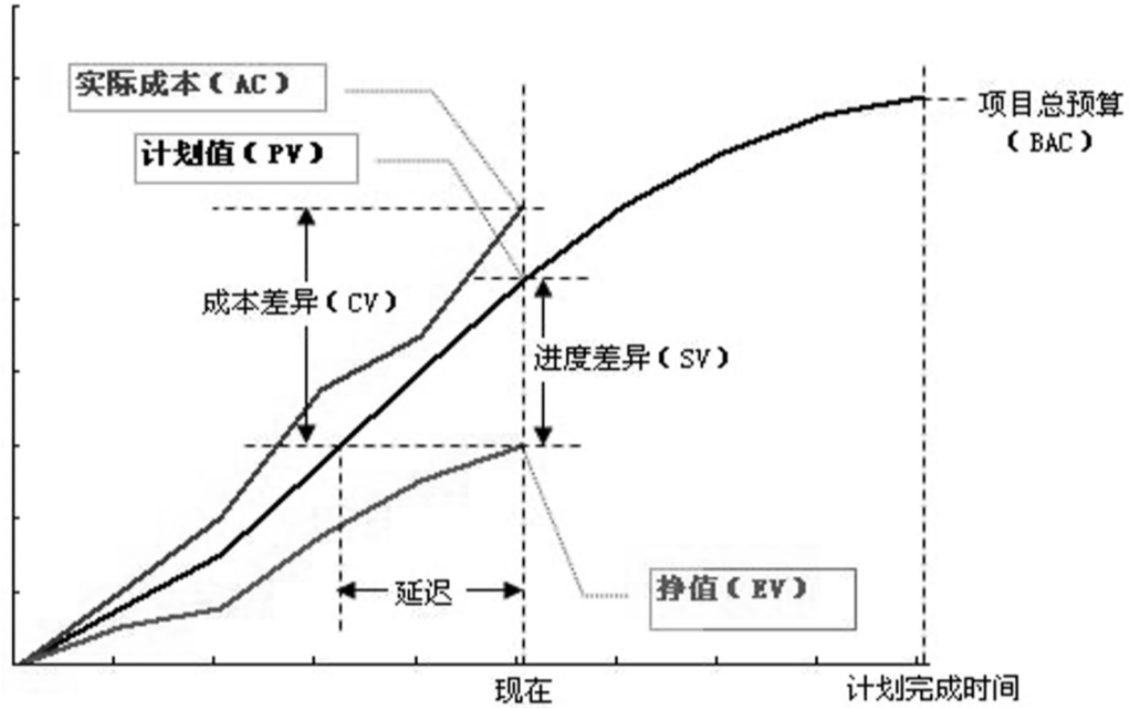


- 日程计划原理和方法
- 质量计划原理和方法
- 风险计划
  - 风险识别
    - 识别与成本、进度及绩效相关的风险
    - 审查可能影响项目的环境因素
    - 审查工作分解结构的所有组件，作为风险识别的一部分，以协助确保所有的工作投入均已考虑
    - 记录风险的内容、条件及可能的结果
    - 识别每一风险相关的干系人
    - 利用已定义的风险参数，评估已识别的风险
    - 依照定义的风险类别，将风险分类并分组
    - 排列降低风险的优先级
  - 风险应对
    - 风险转嫁
    - 风险解决
    - 风险缓解

项目跟踪

- 意义
  - 在项目进展过程中开展跟踪活动的目的在于了解项目进度，以便在项目实际进展与计划产生严重偏离时，可采取适当的纠正措施
  - 以项目计划为进度是否之后的参照物
  - 管理针对偏差而采取的纠偏措施

- **EVM（挣值管理方法）**
  - 定义
    - 每项任务实现附以一定价值
    - 100%完成该项任务，就获得相应价



- **里程碑评审**
  - 定义
    - 往往是指某个时间点，用以标记某项工作的完成或者阶段的结束
  - 内容
    - 项目相关的承诺，如日期、规格、质量等
    - 项目各项计划的执行状况
    - 项目当前的状态讨论
    - 项目面临的风险讨论等

项目总结

- 意义
  - 提供一个系统化的方式来总结经验教训、防止犯同样的错误、评估项目团队绩效、积累过程数据等。提供给项目团队成员持续学习和改进的机会
- 步骤
  - 准备阶段
  - 总结阶段
  - 报告阶段
- **PMBOK总结（Project Management Body Of Knowledge，项目管理知识体系）**
  - 范围管理
    - 关注项目的需求开发过程与变更管理中的得失，对需求管理实际执行情况的差距进行原因分析，找到改进的机会
  - 时间管理

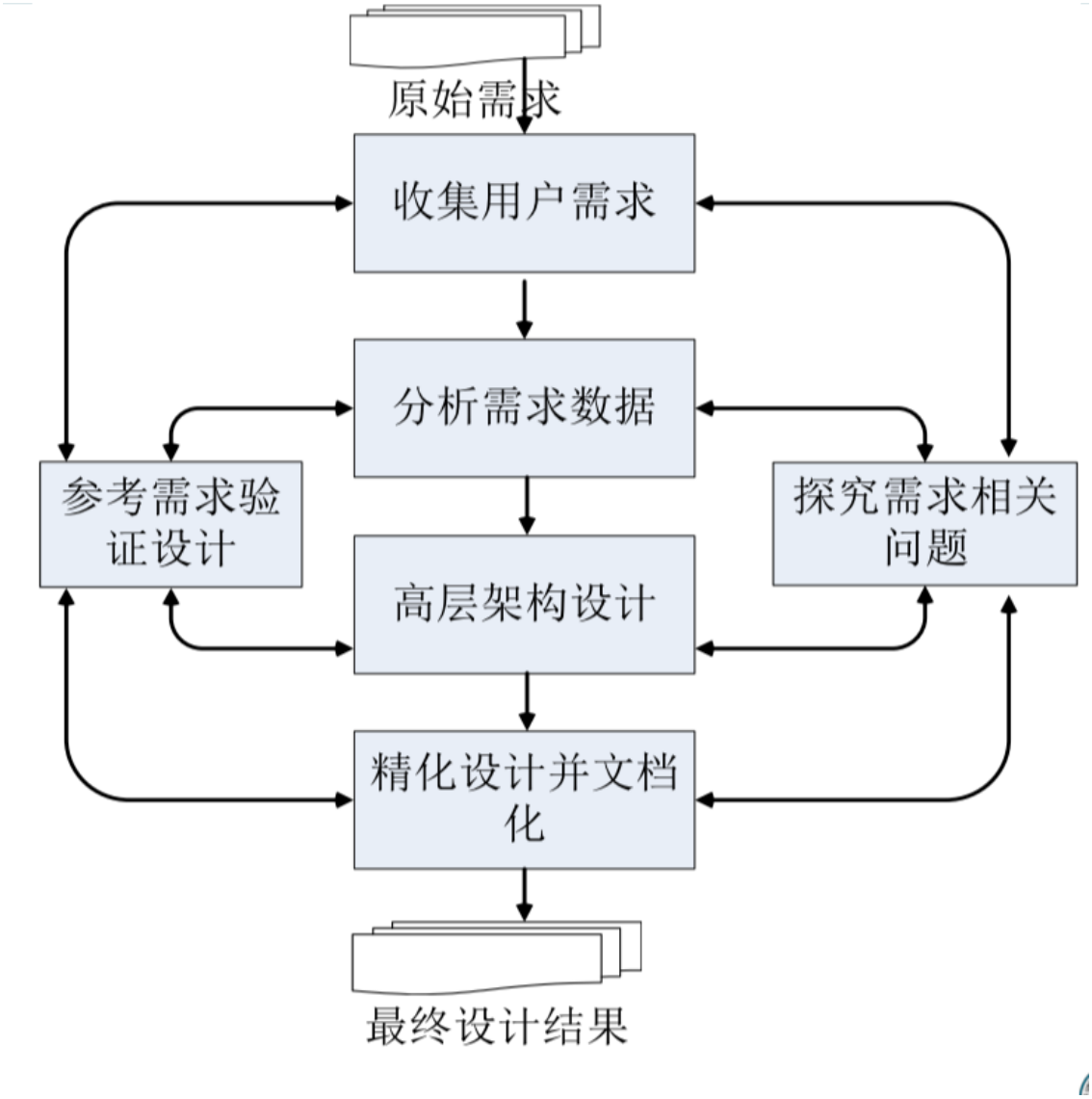
        - 关注项目的日程计划以及对日程计划的跟踪和管理状况，考察计划的准确程度以及各个里程碑的偏差情况
    - 成本管理
        - 包括对成本进行估算、预算和控制的各个过程，从而确保项目在批准的预算内完工
    - 质量管理
        - 包括执行组织确定质量政策、目标与职责的各过程和活动，从而使项目满足其预定的需求
    - 人力资源管理
        - 包括组织、管理与领导项目团队的各个过程
    - 项目沟通管理
        - 包括为确保项目信息及时且恰当地生成、发布、存储、调用并最终处置所需的各个过程
    - 项目风险管理
        - 包括风险管理规划、风险识别、风险分析、风险应对规划和风险监控的等各个过程
    - 采购管理
        - 包括从项目组织外部采购或获得所需产品、服务或成果的各个过程
    - 整合管理
        - 包括识别、定义、组合、统一与协调项目管理过程组的各过程及项目管理活动而进行的各种过程和活动
- **TSP总结**
    - 准备阶段
    - 过程数据评审阶段
        - 分析过程数据，识别过程改进机会
    - 人员角色评价阶段
        - 项目组长
            - 关注领导力、团队激励、团队承诺、会议组织情况
        - 计划经理
            - 关注估算、生产效率、里程碑
        - 开发经理
            - 关注开发内容和开发策略
        - 质量经理
            - 关注从项目整体质量情况出发，总结质量目标的实现过程，并找出改进机会
        - 过程经理
            - 关注团队遵循过程的程度和过程改进方案
        - 支持经理
            - 关注配置管理状况、问题和风险跟踪机制以及复用策略的支持等
        - 工程师
            - 关注个人的绩效（生产效率、质量水平等）
    - 总结报告撰写阶段

软件质量

- 定义：
  - 与软件产品满足规定的和隐含的需求能力有关的特征或者特征的全体
  - PSP：质量是满足用户需求的程度
- PSP质量策略
  - 用缺陷管理替代质量管理
  - 高质量产品意味着要求组成软件产品的各个组件基本无缺陷
  - 各个组件的高质量是通过高质量评审来实现的
- PSP质量指标
  - **Yield**
    - 用以度量每个阶段在消除缺陷方面的效率
    - Phase Yield = $\frac{100*\text{某阶段发现的缺陷个数}}{\text{某阶段注入的缺陷个数}+\text{进入该阶段前遗留的缺陷个数}}$
    - Process Yield = $\frac{100*\text{第一次编译前发现的缺陷个数}}{\text{第一次编译前注入的缺陷个数}}$
  - **A/FR（Appraisal to Failure Ratio，质检失效比）**
    - A/FR = $\frac{PSP\text{质检成本}=\text{设计评审时间}+\text{代码评审时间}}{PSP\text{失效成本}=\text{编译时间}+\text{单元测试时间}}$
    - PSP中A/FR期望值为2.0
  - **PQI（Process Quality Index，过程质量指标）**
    - PQI = $A*B*C*D*E$
    - A = 设计质量：设计的时间应该大于编码的时间
    - B = 设计评审质量：设计评审的时间应该大于设计时间的50%
    - C = 代码评审质量：代码评审时间应该大于编码时间的50%
    - D = 代码质量：代码的编译缺陷密度应当小于10个每千行
    - E = 程序质量：代码单元测试缺陷密度应当小于5个每千行
    - PSP中PQI期望值为大于0.4
  - **评审速度（Review Rate）**
    - PSP中评审速度期望值为代码评审速度小于 200 LOC每小时，文档评审速度小于 4 Page每小时
  - **DRL（缺陷消除效率比）**
    - 以某个测试阶段（一般为单元测试）每小时发现的缺陷数为基础，其他阶段每小时发现缺陷数与该测试阶段每小时发现的缺陷的比值

软件设计

- 设计与质量的关系
  - 低劣的设计是导致在软件开打过程中反攻、不易维护以及用户不满的主要原因
  - 成分设计开源显著减少最终程序的规模，提升质量
  - 设计本身也是一种排错的过程
- PSP设计过程



- 设计的内容
  - 目标程序在整个应用系统中的位置
  - 目标程序的使用方式
  - 目标程序与其他组件以及模块之间的关系
  - 目标程序外部可见的变量和方法
  - 目标程序内部运作机制
  - 目标程序内部静态逻辑
- PSP设计模板
  - 操作规格模板（Operation Specification Template，OST）
    - 外部动态信息
    - 描述系统与外界的交互，即“用户”与待设计系统的正常情况和异常情况下的交互
    - 可以用来定义测试场景和测试用例，也可以作为和系统用户讨论需求的基础
  - 功能规格模板（Functional Specification Template，FST）
    - 外部信息
    - 描述系统对外的接口，如类和继承关系、外部可见的属性和外部可见的方法等
  - 状态规格模板（State Specification Template，SST）
    - 内部动态信息
    - 精确定义程序的所有的状态，状态之间的转换以及伴随每次状态转换的动作
      - 所有状态的名称
      - 所有状态的简要描述
      - 在SST中需要使用的参数和方法的名称与描述

        - 状态转换的条件
        - 状态转换时发生的动作
    - 逻辑规格模板（Logical Specification Template，LST）
        - 内部静态信息
        - 精确描述系统的内部静态逻辑
            - 关键方法的静态逻辑
            - 方法的调用
            - 外部引用
            - 关键数据的类型和定义
- UML图
    - 用例图
    - 时序图
    - 类图
    - 状态机图

设计验证

- 状态机验证（正确的状态机是完整且正交的）
    - 检验状态机，消除死循环和陷阱状态
    - 检查状态转换，验证完整性和正交性
    - 评价状态机，检验是否体现设计意图
- 符号化执行验证
    - 识别伪码程序中的关键变量
    - 使用代数符号表示并重写伪码程序
    - 分析伪码程序的行为
- 执行表验证
    - 识别伪码程序的关键变量
    - 构建表格。表格左侧填入主要程序步骤，右侧填入关键变量
    - 初始化被选定的变量
    - 跟踪被选择的关键变量的变化情况，从而判断程序行为
- 跟踪表验证
    - 识别伪码程序的关键变量
    - 构建表格，表格左侧填入主要程序步骤，右侧填入关键变量
    - 初始化被选定的变量
    - 识别将伪码程序符号化的机会，并加以符号化
    - 定义并且优化用例组合
    - 跟踪被选择的关键变量的变化情况，从而判断程序行为
- 正确性验证
    - 分析和识别用例
    - 对于复杂伪码程序的结构，应用正确性检验的标准问题逐项加以验证
    - 对于不能明确判断的复杂程序结构，使用跟踪表等辅助验证

团队工程开发

- **需求开发**
  - 需求是一切工程活动的基础
  - 需求类别
    - 客户需求
    - 产品需求
    - 产品组件需求
  - 步骤
    - 需求获取
    - 需求汇总
      - 整理各种来源的信息，识别缺失的信息
      - 解决冲突的需求
      - 需求的整理和转化
      - 推导未显式描述的需求内容
    - 需求验证
      - 建立和维护操作概念和相关的场景
      - 分析需求
      - 确认需求
    - 需求文档制作
- **团队设计**
- **实现策略**
  - 评审的考虑
  - 复用策略
  - 可测试性考虑
- **集成策略**
  - 大爆炸集成策略
  - 逐一添加集成策略
  - 集簇集成策略
  - 扁平化集成策略
- **验证与确认**
  - 验证的目的是确保选定的工作产品与事先指定给该工作产品的需求一致
  - 确认的目的是确保开发完成的产品或者产品租价再将要使用该产品的产品或者产品组件的环境中工作正确
  - 步骤
    - 环境准备
    - 对象选择
    - 活动实施
    - 结果分析

配置管理

- **配置管理**
  - 配置项
  - 基线
  - 目的
    - 建立与维护工作产品的完整性
- **度量与分析**
  - 意义
    - 支持客观的估计与计划
    - 支持根据建立的计划和目标，跟踪实际进展
    - 识别与解决过程改进相关议题
    - 提供将度量结果纳入未来其他过程的基础
  - GQM（目标-问题-度量）方法
- **决策分析**
  - 活动
    - 建立评估备选方案的准则
    - 识别备选解决方案
    - 选择评估备选方案的方法
    - 使用已建立的准则与方法，评估备选解决方案
    - 依据评估准则，从备选方案中选择建议方案
  - 根因分析（鱼骨图、石川图）

定量管理

- 基本范式
  - 构建定量模型
    - 子过程能力基线
    - 过程模型
  - 应用模型
    - 监控影响子过程的关键因素
- 基本概念
  - （子）过程性能
    - 遵循某个特定（子）过程的之后产生结果的量化描述，既包括（子）过程度量Xi（例如，时间、缺陷消除效率、工时等），也包括产物度量（例如，缺陷密度，相应时间等）
  - （子）过程性能基线
    - 上述过程性能的一个定量化的刻画，一般包括均值和范围。通常用作过程性能的benchmark
  - （子）过程性能模型
    - 依据子过程的逻辑关系构建相应的数学模型，描述子过程性能基线和整体过程有意义的性能输出（例如，质量、生产效率、成本等）之间关系。例如Process Yield和Phase Yield
- 常用技术
  - 直方图

        - 描述
            - 过程属性的频率
            - 例如缺陷修复的时间
            - 每次测试发现的缺陷的个数
            - 每天无故障工作的时间
            - 产品的不合格率
            - 等
        - 用于
            - 判断过程是否正常
            - 过程能力是否满足需要
            - 不良产品是否发生
            - 产品质量问题的原因
    - 帕累托图
        - 按照由高到低排列的直方图
    - 因果图
    - 散点图
    - 控制图
    - 回归分析
    - 假设检验
    - 方差分析

仿真建模

- 模型
    - 模型是对真实或概念复杂系统的抽象。一个模型被设计用来显示人人们想要研究、预测、修改或控制的系统的显著特征和特性
- 仿真模型
    - 计算机化的模型，代表某种动态系统或现象，当被建模系统的复杂性超出静态模型或其他技术所能有效表示的范围时，通常采用仿真解决实际系统中遇到的复杂性：系统不确定性和随机性、动态行为、反馈机制
- 软件过程仿真模型
    - 软件过程仿真模型关注于特定的软件开发/维护/演化过程。它可以表示当前已实施（按现状）或计划在未来实施的过程
- 使用原因
    - 战略管理
    - 规划
    - 控制和运营管理
    - 流程改进和技术采用
    - 理解
    - 培训和学习
- 步骤
    - 确定建模的范围
    - 定义结果变量
    - 过程抽象
    - 选择和定义输入参数
- 技术

- 蒙特卡罗仿真
- 离散时间仿真
- 系统动力学防战
- 混合仿真