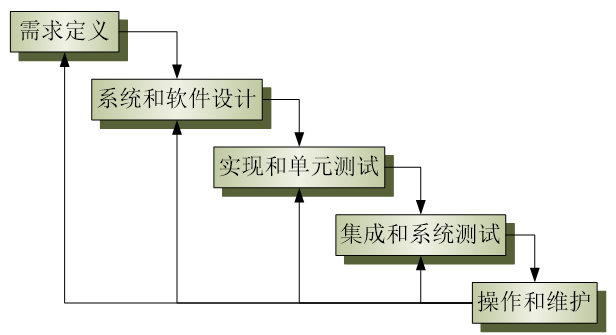
PPT3 软件过程

1. 软件过程模型
2. 瀑布模型：每个过程阶段独立，直到上一个过程完成，下一个过程才启动



其间伴随着过程反复

瀑布模型的缺点是过程进行中对应变更比较困难。

问题：将项目生硬地划分成几个明显不同的阶段，这使得响应客户的需求比较困难，所以这个模型适用于需求能够较好地理解的情况

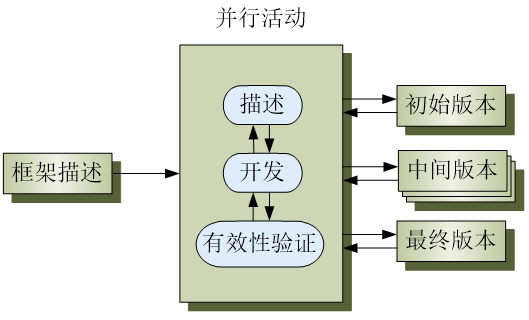
1. 进化式开发：描述与开发交替进行

探索式：其目标是与客户一起工作，从最初的需求草案进化到最后的系统。应该从理解最清楚的需求开始。

抛弃式：目的是理解系统需求。从理解较差的需求开始

问题：缺乏过程可见性；系统结构通常较差；需要特殊的工具和技术 (例如采用快速建立原型的语言)

适用性：中小型交互式系统；大型系统的一部分 (如用户界面)；生命周期短的系统

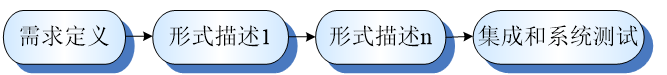


1. 形式化的系统开发

基于用形式化数学转换将系统描述转换成一个可执行程序

转换是保证正确性的，所以程序是符合描述的。

具体实现如IBM的净室(Cleanroom)过程软件开发

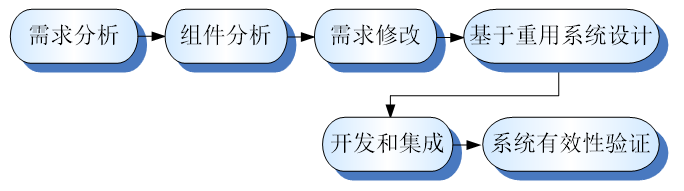


问题：应用这项技术需要专门的技能和训练；系统的某些方面难以形式化描述，例如用户界面

适应性：适合于严格的系统，特别是安全性和保密性要求极高的系统

1. 面向复用的开发

基于已有的组件或者COTS系统的系统重用



1. 过程反复：系统需求在项目进行期间总是进化的，所以对大型系统来说经常是早期阶段反复的过程；反复可以应用于任何通用过程模型

两个混合模型

1. 增量式开发：系统不是一次交付，而是将要求的功能分成多次增量进行开发和交付

用户需求是有优先顺序的。优先级最高的需求包含在最初的增量中。

一旦一个增量的开发开始时，需求就要冻结。虽然后面的增量可以继续进化

优势：客户要求的功能随着每次增量被交付，所以系统功能可以较早看到

较早的增量可以作为原型帮助引出后面增量的需求

项目总体失败的风险降低

最高优先级的系统服务由于是最早增量，得到最多的测试

极限编程：基于非常小的功能增量的开发和交付的一种新的方法；依靠连续的代码改进，用户参与到开发团队一起开发，是敏捷开发的一种

1. 螺旋式开发

将过程表示为螺旋线，而不是用一系列活动和活动间的回溯来表示

螺旋线中的每个回路表示过程中的一个阶段

没有固定的阶段。螺旋线中的回路根据需要选取

风险需要明确评估、然后在过程中解决

1. 软件设计方法
2. 数据流模型：包括数据元素，数据操作，数据流向
3. 实体-关系模型
4. 结构模型
5. 对象模型：将数据与操作封装视为对象，具有一定的属性，各个对象之间连线构成模型

PPT4

1. 项目管理
2. 软件项目管理涉及的行为，是为了确保软件及时并按照进度的要求交付，同时满足开发或者采购该软件的机构的需求 项目管理是需要的，因为软件开发总是受到预算和进度的约束，这些约束是由开发该软件的机构所设置的。
3. 软件管理的特别之处：

产品是无形的

产品是灵活可变的，其它产品中这是比较少见的。

软件工程学科被认为还是不完整的，跟机械、电子工程等学科不同

软件开发过程是非标准化的

许多软件项目是一次性的项目

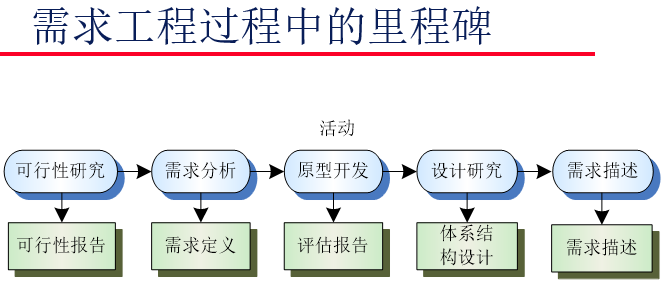
1. 项目计划的类型：

质量计划，有效性验证计划，配置管理计划（支持系统的集成计划，使每个开发人员可以在管理中访问工程代码和文档，超找变更，编译连接组件并生成系统），维护计划，人员发展计划

1. 活动组织

里程碑：是一个过程活动的终点，瀑布过程可以定义明显的进度里程碑

可交付物：是交付给客户的项目结果



1. 项目调度

将项目分解成多个任务，估算完成每个任务需要的时间和资源

并发地组织任务以形成劳动力的优化利用

使得任务间的依赖性最小化以避免一个任务等待另一个任务带来的延迟

依赖于项目管理者的直觉和经验

甘特图：活动的时间条形图

1. 风险管理

风险包括项目风险、产品风险和业务风险

过程：

风险识别：识别项目、产品和业务风险

风险分析：评估这些风险出现的可能性及其后果

风险规划：制订计划说明如何规避风险或降低风险对项目的影响

风险监控：监控整个项目过程中的风险

管理者要扮演多种角色，其中最主要的活动是规划、估算和进度

PPT5

1. 需求工程：

建立用户需求以及使用和开发的约束的过程

需求工程过程中生成的需求本身是系统服务和约束的描述

1. 需求的双重功能：

可能是一个合同标书的基础 – 所以必须公开解释

可能是合同本身的基础 – 所以必须详细定义

这两种情形都可能称为需求

1. 需求的类型：

用户需求：关于系统服务和约束的自然语言加上方块图表述。为客户撰写。较为粗略。

系统需求：一个结构化的文档写出系统的服务。作为客户和承包商之间的合同内容。较为细致。

软件描述：一个详细的软件描述可以作为设计或实现的基础。为开发人员撰写。

1. 功能与非功能需求

功能需求：系统需要提供的服务的表述，系统应该如何响应特定输入，系统在特定的情形下应该如何动作。

非功能需求：系统提供的服务或功能上的约束，例如时间约束、开发过程约束、标准等。

领域需求：需求从系统应用领域中得出，反映了领域的特征。

1. 需求应该具有完整性和一致性

完整性：需要的所有服务都应该给出描述

一致性：在系统服务的描述上应该没有冲突和矛盾

1. 非功能需求分类

产品需求：描述交付产品的行为，如执行速度、可靠性等

机构需求：机构政策和程序的结果，如过程标准、实现要求

外部需求：系统外部因素和开发过程，如互操作性要求、法律要求等

1. 需求的度量：

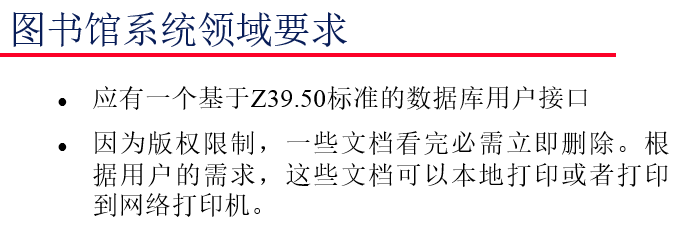


1. 领域需求

从使用领域中得到，描述反映领域的特征和性质

可能是新的功能需求、已有需求的约束或者定义一个特定的计算

如果领域需求不被满足，系统可能无法工作

如：

问题：

易懂性不够：需求使用应用领域中的语言表达；开发系统的软件工程师往往无法理解

不够清晰明了：领域专家能够很好的理解领域知识所以他们不愿把领域需求表达得更加清晰明了

1. 用户需求：

应该描述功能性和非功能性需求，使得没有具体的技术知识的系统用户也能理解

用户需求用自然语言、表和方块图定义

自然语言的问题：

不够清楚：为了使得文档易读，保证精确性是困难的

需求混乱： 功能性和非功能性需求会混在一起

需求合并：几个不同的需求可能放在一起表达

1. 系统需求

比用户需求更详细的描述

作为系统设计的基础

可以作为系统合同的一部分

系统需求可用结构化的自然语言、PDL或者格式化的语言来撰写

1. 需求和设计：

自然语言的问题：

二义性：需求的读者和作者必需对同一词语有同样的解释。自然语言是做到这点比较困难，因为自然语言存在二义性。

随意性太大：同一件事情可能在描述中用好几种不同的方式讲述.

模块化不够：自然语言的结构不足以构建系统需求

改用结构化的语言描述

1. 需求文档

需求文档是对系统开发者要求的正式表述

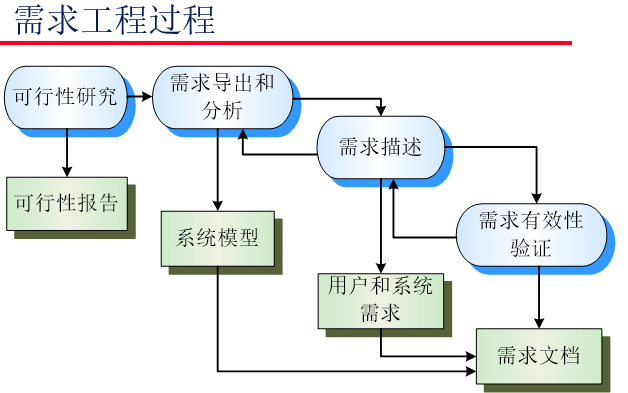
应该包括系统定义和需求描述

不是设计文档，陈述的是系统应该做什么而不是怎么做。

1. IEEE 需求标准

PPT6 需求工程过程

1. 共通的过程：需求导出，需求分析，需求验证，需求管理



1. 可行性研究

决定一个建议的系统是否值得做

1. 导出和分析

也叫需求导出或者需求发现

相关的技术人员和客户一起工作以发现应用领域、系统提供的服务、系统的运行限制

可能涉及终端用户、管理者、维护工程师、领域专家等项目相关人员。

需求分析有三种活动

区分. 识别实体之间的结构关系

提取. 识别实体中的一般特性

发散. 识别看一个问题的不同方式

1. 面向视点的需求导出

项目相关人员不同的问题视点，这种多视点的分析是重要的，因为分析系统需求没有单个正确的方式

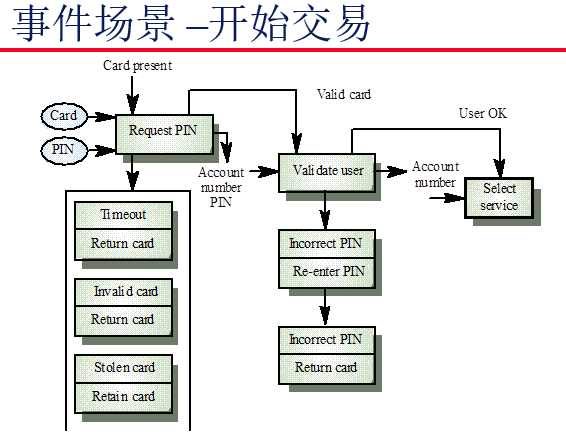
过程模型：

视点识别：发现接收系统服务的视点，以及识别每个视点提供的服务

视点组织：组织相关的视点形成层次结构。通用的放在较高的层次。

视点文档：对被识别的视点和服务描述的精炼。

视点系统映射：将分析转化为面向对象的设计



椭圆. 来自视点和交付给视点的数据

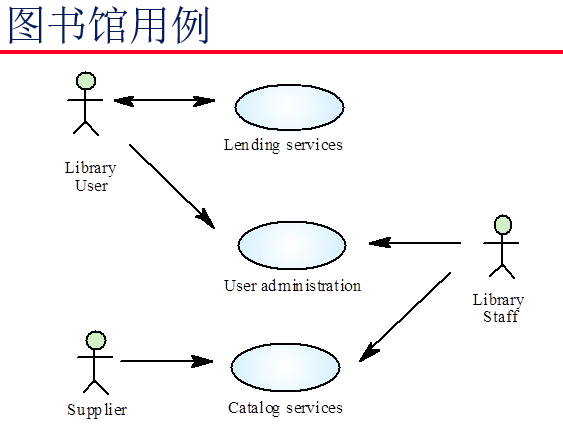
控制信息从每个方框的顶部出入

数据从每个方框的右侧出来

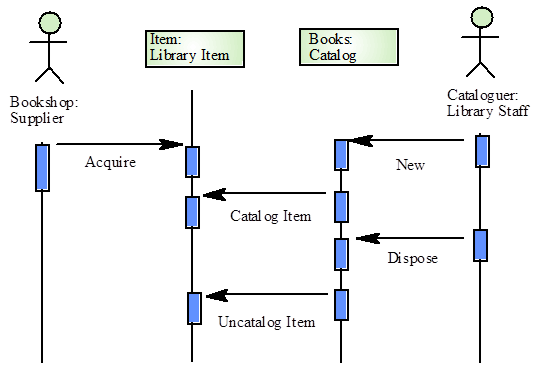
异常显示在方框的底部

场景完成后预期的下一个事件的名称表示在一个粗线框中

1. 用例



1. 序列图：目录管理的序列图



序列图中：方块表示活动，横向箭头表示转移，纵向直线表示时间

1. 需求的有效性验证

证明系统中定义的需求是客户真正想要的

需求错误的代价是很高的，所以有效性验证非常重要

需求有效性验证技术：

需求评审：对需求做系统性的手工分析

原型开发：用一个可执行的系统模型来检查需求。在第8章详述

测试案例生成：检查需求的易测性

自动一致性分析：检查结构化需求描述的一致性

1. 需求管理过程包括规划和变更管理

PPT7 系统模型

一 系统建模

系统建模有助于分析者理解系统功能，模型可用来和客户沟通

不同的模型从不同的角度来描述系统

从外部来看，是对系统上下文或系统环境建模

从行为来看，是对系统行为建模

从结构上看，是对系统的体系结构和系统处理的数据的结构建模

二 系统模型类型

数据处理模型，说明数据在不同的阶段如何被处理的。（数据流图）

组成模型，说明系统中的实体是如何由其它实体组成的。（实体-关系图）

体系结构模型，说明构成整个系统的主要子系统。（体系结构图）

分类模型，说明实体间怎样具有共同特性。（对象类/继承关系图）

激励/响应模型，说明系统对事件的响应。（状态转换图）

1. 上下文建模

上下文模型用来说明系统的边界