软件需求

- 对系统进行描述

目标

- 用户需求和系统需求的概念
- 功能需求和非功能需求
- 描述系统需求的两个技术
- 软件需求如何在需求文档中组织

内容

- 功能和非功能需求
- 用户需求
- 系统需求
- 软件需求文档

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

什么是需求?

只要用户规定的，就是需求

- 范围很广，高到服务和约束的高层抽象描述，低到具体的数学形式化的功能描述
- 需求具有双重功能
  - 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

合同必须全是自然语言，不能用图表（形象但不谨）

一个功能描述有可能既是需求又是设计（如产品的颜色），视具体情况而定

需求提取(Davis)

如果一家公司要与另一机构签订一个大型软件开发项目合同，那这家公司就要尽量概要地定义对该项目的要求，而且在这样的描述中不应限制解决方案。这时候，就需要一个文本形式的需求以便多个承包商竞标。一旦签订了合同，承包商就要为客户写出更详细的系统定义，要让用户能看懂并要在此确认系统到底需要提供哪些服务。这两种文件都被称为需求文档。

需求类型

- 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。
- 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。
- 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。

定义和描述

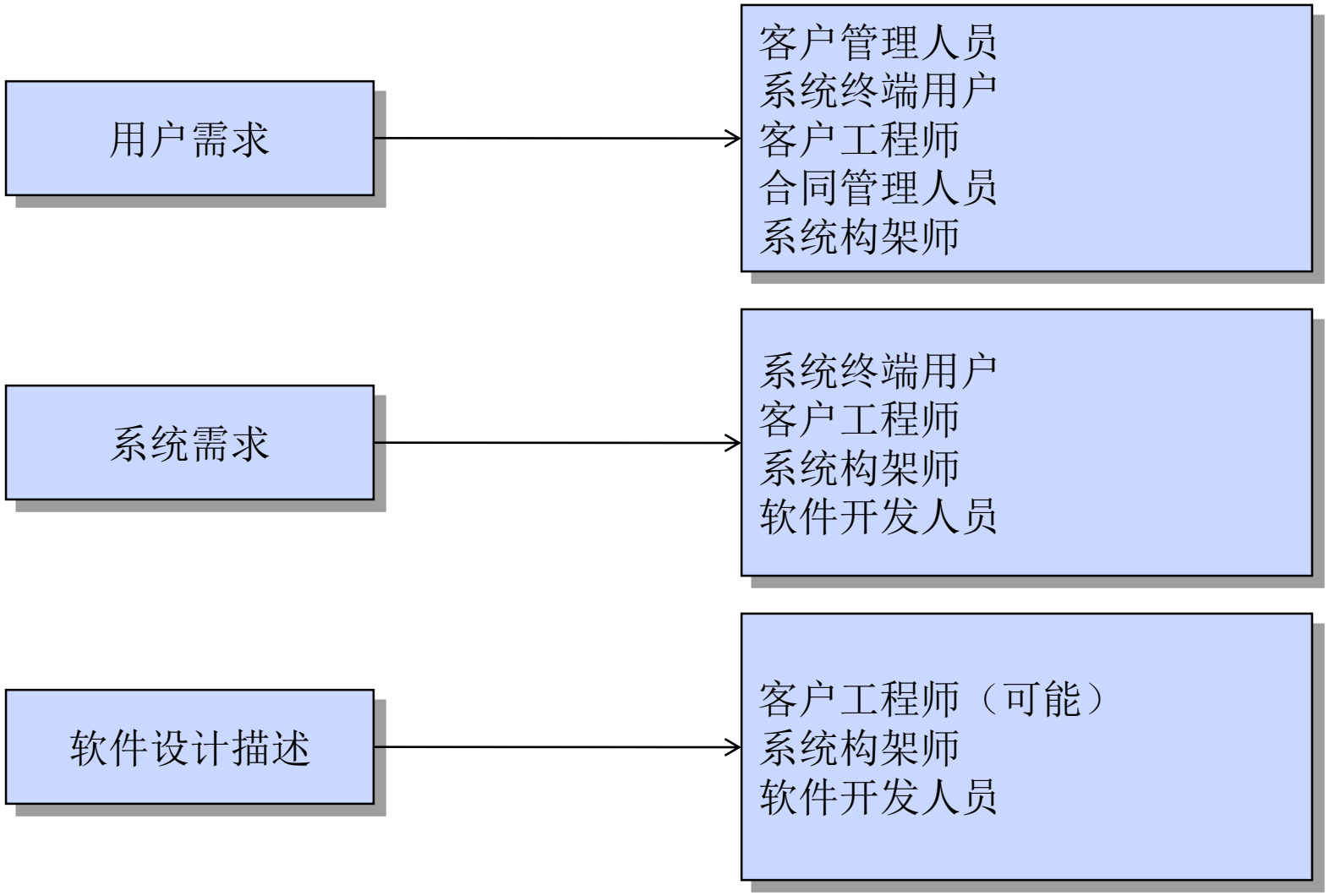
用户需求定义

1.该软件应该提供显示和访问由别的工具产生的外部文件的功能

系统需求描述

1.1 用户应该能定义外部文件的类型
1.2 每种外部文件类型应有一个关联的工具
1.3 每种外部文件类型都能在用户的显示器上显示
1.4 用户能够为外部文件类型定义图标
1.5 当用户选择了一个图标，相当于执行相对应的工具

需求阅读者


用户需求
客户管理人员
系统终端用户
客户工程师
合同管理人员
系统构架师
系统需求
系统终端用户
客户工程师
系统构架师
软件开发人员
软件设计描述
客户工程师（可能）
系统构架师
软件开发人员

功能和非功能需求

- 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。

功能需求

- 描述功能或系统服务
- 依赖于软件的类型，预期的用户和软件所应用的系统的类型
- 功能性用户需求可能是高层的表述，关于系统应该做什么，但是功能性系统需求则应该详细描述系统的服务

功能需求的例子

- 用户应该能够从总的数据库中查询或者是选择其中的一个子集并从中查询。
- 系统应该提供适当的浏览器供用户阅读馆藏文献。
- 每次借阅能对应一个独特的识别符（ORDER_ID），可拷贝到用户账户的常备储存区内。

需求不精确

- 当需求没有被精确定义时，会带来问题
- 模糊的需求可能开发者和用户有不同的解释
- 例如 ‘适当的浏览器’
  - 用户的目的 – 各个不同的文档类型有特殊用途的浏览器
  - 开发者解释 – 提供一个可以显示文档内容的文本浏览器

需求的完整性和一致性

- 理论上，需求应该既完整又一致
- 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
- 实际上, 不可能产生一个完全完整的、没有不一致的需求文档

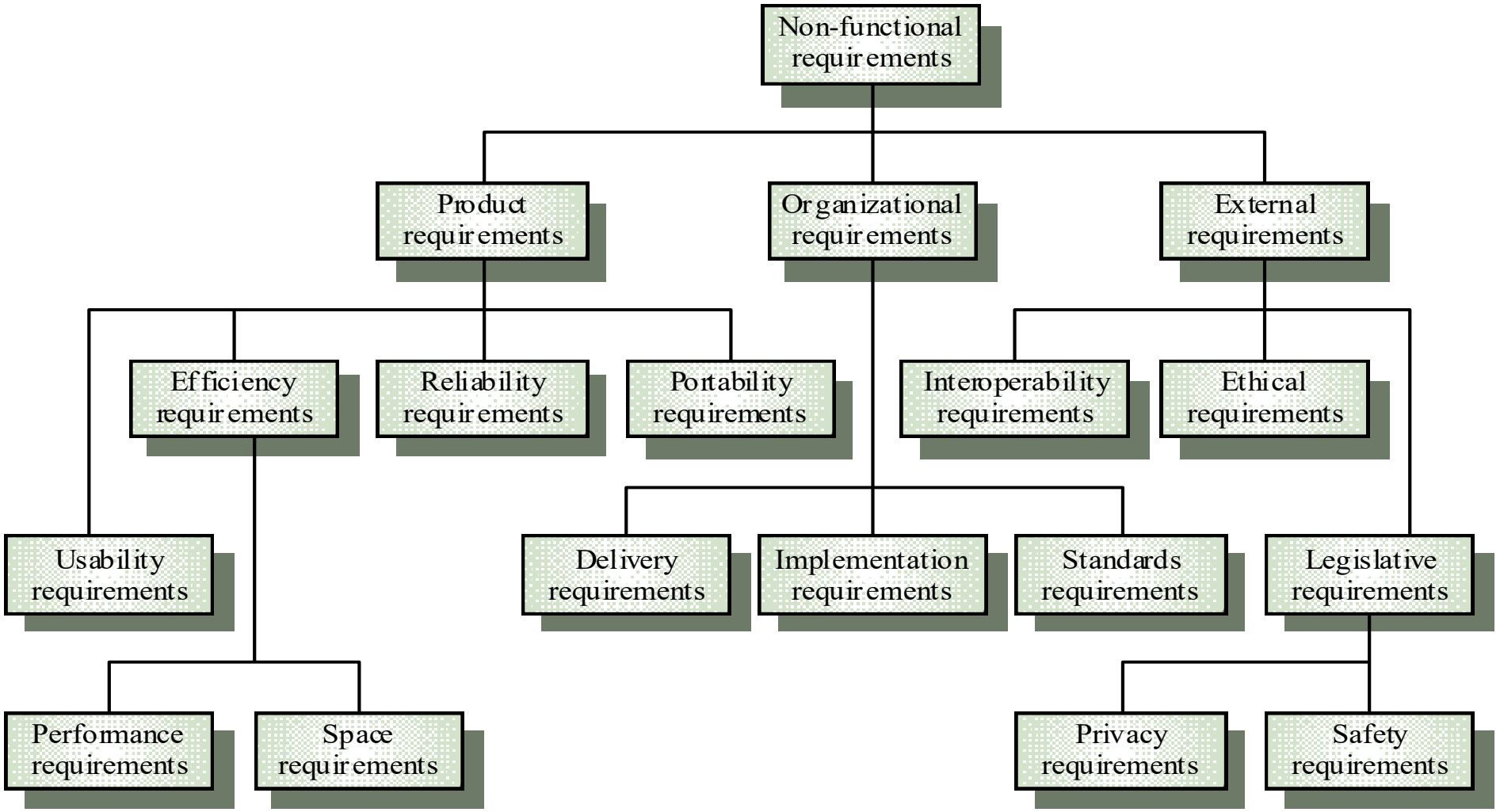
非功能需求

- 定义系统特性和约束，例如可靠性、响应时间、存储需求。约束是I/O设备容量、系统显示方式等。
- 过程需求可能也会规定CASE工具、编程语言或者开发方法
- 非功能需求可能比功能需求更关键。如果这些没有满足，系统是没用的。

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

非功能需求类型


Non-functional requirements
Product requirements
Organizational requirements
External requirements
Efficiency requirements
Reliability requirements
Portability requirements
Interoperability requirements
Ethical requirements
Usability requirements
Delivery requirements
Implementation requirements
Standards requirements
Legislative requirements
Performance requirements
Space requirements
Privacy requirements
Safety requirements

非功能需求例子

- 产品需求
  - 4.C.8 它应该能将所有APSE和用户之间的必需的通信用标准的Ada字符集表达
- 机构需求
  - 9.3.2 系统开发过程和可交付的文档将遵照XYZCo-SP-STAN-95中的相关定义
- 外部需求
  - 7.6.5 系统不应该对系统的操作人员公开客户除名字和索引代码之外的任何个人信息

目标和需求

- 非功能需求可能很难精确表述，而模糊的需求可能难于检验
- 目标
  - 用户的总体目的，例如易用性
- 能检验的非功能需求
  - 能够被客观地度量的描述
- 目标有助于开发者理解系统使用者的意图

举例

- **系统目标**
  - 对有经验的管理员来说，系统应该容易使用，其构成应该使得用户的错误最小
- **可检验的非功能需求**
  - 经过2小时的培训，有经验的管理员应该能使用所有的系统功能。经过培训后，有经验的用户的平均错误数量每天应不超过2个。

需求度量

属性	度量
速度	每秒处理的事务数 用户/事件响应时间 屏幕刷新时间
规模	K字节 RAM芯片数量
易用性	培训时间 帮助的页数
可靠性	平均无故障时间 不可用的概率 故障发生的概率 可用性
鲁棒性	故障后的重启时间 引起故障的事件的百分比 故障时数据损坏的可能性
可移植性	目标依赖条件的比例 目标系统的数量

需求的相互作用

- 在复杂的系统中，不同的非功能需求经常会冲突
- 太空船系统
  - 为了使重量最小，系统中独立的芯片数量应该最小
  - 为了使功耗最小，应该使用低功耗芯片
  - 然而，使用低功耗芯片可能意味着使用更多的芯片，哪个是更关键的需求呢？

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

图书馆系统领域要求

- 应有一个基于Z39.50标准的数据库用户接口
- 因为版权限制，一些文档看完必需立即删除。根据用户的需求，这些文档可以本地打印或者打印到网络打印机。

列车保护系统

- 列车的减速应如下计算:
  - $D_{train} = D_{control} + D_{gradient}$

  这里 $D_{gradient}$ is $9.81 m/s^2$ * 补偿坡度/alpha，而$9.81 m/s^2$/alpha是不同列车所特有的值

领域需求问题

- 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

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

数据库要求

需求合并

4.A.5 数据库应该支持配置对象的生成和控制；也就是说，数据库中对象本身是由其它的对象组合而成。配置控制工具应该允许对一个版本集合中的对象根据其不完全的名字访问它们。

带栅格编辑器的用户需求

需求合并

2.6 栅格工具 为协助在一个图表中定位实体，用户可能通过面板上的选择项打开一个以公分或英寸为单位的栅格，最初栅格处于关闭状态。在编辑期间可以随时切换栅格的打开和关闭，并可以随时在英寸和公分单位之间转换。栅格选择项以调节方式提供，当图形较小时栅格线的数目将会减少，以避免图表被栅格填满。

需求的问题

- 数据库需求包括概念上的和具体的信息
  - 描述配置控制工具的概念
  - 包含了使用非完全名称访问时的细节，应该分开
- 栅格需求混合了三种不同的需求
  - 概念上的功能需求 (栅格的必要性)
  - 非功能需求 (栅格单位)
  - 非功能的用户界面需求 (栅格打开和关闭)

结构化表述

名称（唯一性、确定性）、描述（介绍性、清晰易懂）、输入及来源、输出及目的地、前置条件后置条件、决定者（负责人）、变更记录等等

2.6 栅格工具

2.6.1 编辑器应该提供栅格工具，包含一个由水平和垂直线阵列作为编辑器窗口的背景。这个栅格是被动的，即实体的对准是用户的任务。栅格是被动的，即实体的对准是用户的任务。

基本原理：一个栅格帮助用户产生一个具有间隔的整齐的图表。虽然一个主动栅格能自动将实体与某栅格线对齐，但这种定位也是不精确的。用户是决定实体应该在哪里放置的最好的决定者

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6

编号方便查找、需求跟踪、讨论时的称谓；
需求删除后编号一并删除，新增的需求编号不能占用已删除的编号（防止其他需求对已删除的需求的引用导致需求混乱）

详细用户需求

3.5.1 在设计中添加节点

3.5.1.1 **编辑器必须为提供用户工具，把一个特别类型的节点加入到设计中**

3.5.1.2 应该按下列步骤增加一个节点:

1. 用户选择要添加的节点的类型
2. 用户将光标移动到图形中要添加节点的恰当位置，然后指出要在该处添加节点
3. 然后用户将节点符号拖放到它上面指定的位置

基本原理: 用户是决定该在哪里放置节点的最佳人选。这个方法对用户选择节点类型和定位提供了直接控制

Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1

撰写需求的指南

- 设计一个标准的格式，用它来撰写所有的需求
- 使用一致的语言。要区别强制性和希望性的需求。
- 对文本加亮（如黑体、斜体）来突出显示关键性的需求
- 尽量避免使用计算机专业术语

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
- 系统需求可以用系统模型（第7章讨论）表达

需求和设计

- 原则上, 需求应该规定系统应该做什么，设计则描述系统如何做。
- 实际上, 需求和设计是不能分离的
  - 系统体系结构可能用来构成需求描述
  - 系统和其它系统存在交互操作的约束，这也产生设计需求
  - 使用特别的设计可能是一个领域需求，例如采用特殊编程环境以提高可靠性

自然语言描述的问题

- 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

结构化语言描述

- 格式限制的自然语言用来表达需求
- 这排除了二义性、随意性带来的问题，为描述带来一定程度的规范
- 通常采用一个以格式模板为基础的方法

基于格式的描述

- 功能或实体的定义
- 输入和来源的描述
- 输出和目的地的描述
- 其它被引用实体的索引
- 前条件和后条件
- 对操作的副作用 (如果有的话)

基于格式的节点描述

ECLIPSE/Workstation/Tools/DE/FS/3.5.1

Function Add node

Description Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.

Inputs Node type, Node position, Design identifier.

Source Node type and Node position are input by the user, Design identifier from the database

Outputs Design identifier.

Destination The design database. The design is committed to the database on completion of the operation.

Requires Design graph rooted at input design identifier.

Pre-condition The design is open and displayed on the user's screen.

Post-condition The design is unchanged apart from the addition of a node of the specified type at the given position.

Side-effects None

Definition: ECLIPSE/Workstation/Tools/DE/RD/3.5.1

基于PDL的需求定义

- 需求用一种类似程序设计语言来描述但是具有更大的表达能力
- 两种情况下最合适
  - 当操作可分解为一系列动作且顺序非常关键的情况
  - 当硬件和软件的接口必须被定义的时候
- 缺点
  - 定义领域概念时表达能力不够
  - 这种描述更多被看成是设计描述

ATM操作的PDL描述

```
class ATM {
        // declarations here
        public static void main (String args[]) throws InvalidCard {
                try {
                        thisCard.read () ;  // may throw InvalidCard exception
                        pin = KeyPad.readPin () ; attempts = 1 ;
                        while ( !thisCard.pin.equals (pin) & attempts < 4 )
                                {       pin = KeyPad.readPin () ;  attempts = attempts + 1 ;
                                }
                                if (!thisCard.pin.equals (pin))
                                        throw new InvalidCard ("Bad PIN");
                        thisBalance = thisCard.getBalance () ;
                        do {  Screen.prompt (" Please select a service ") ;
                                service = Screen.touchKey () ;
                                switch (service) {
                                        case Services.withdrawalWithReceipt:
                                                receiptRequired = true ;
```

PDL的缺点

- 用一种易懂的方式来表达系统功能，PDL的表达能力是不够的
- 符号只有懂得编程语言的人员才能搞懂
- 需求可能作为设计描述，而不是帮助人们理解系统的一个模型

接口描述

- 多数系统和其它系统有交互，交互接口必须作为需求的一部分被描述
- 三种类型的接口应该定义
  - 程序接口
  - 交换数据的数据结构
  - 数据的表示
- 格式化的符号是接口描述的有效技术

打印服务器接口的PDL 描述

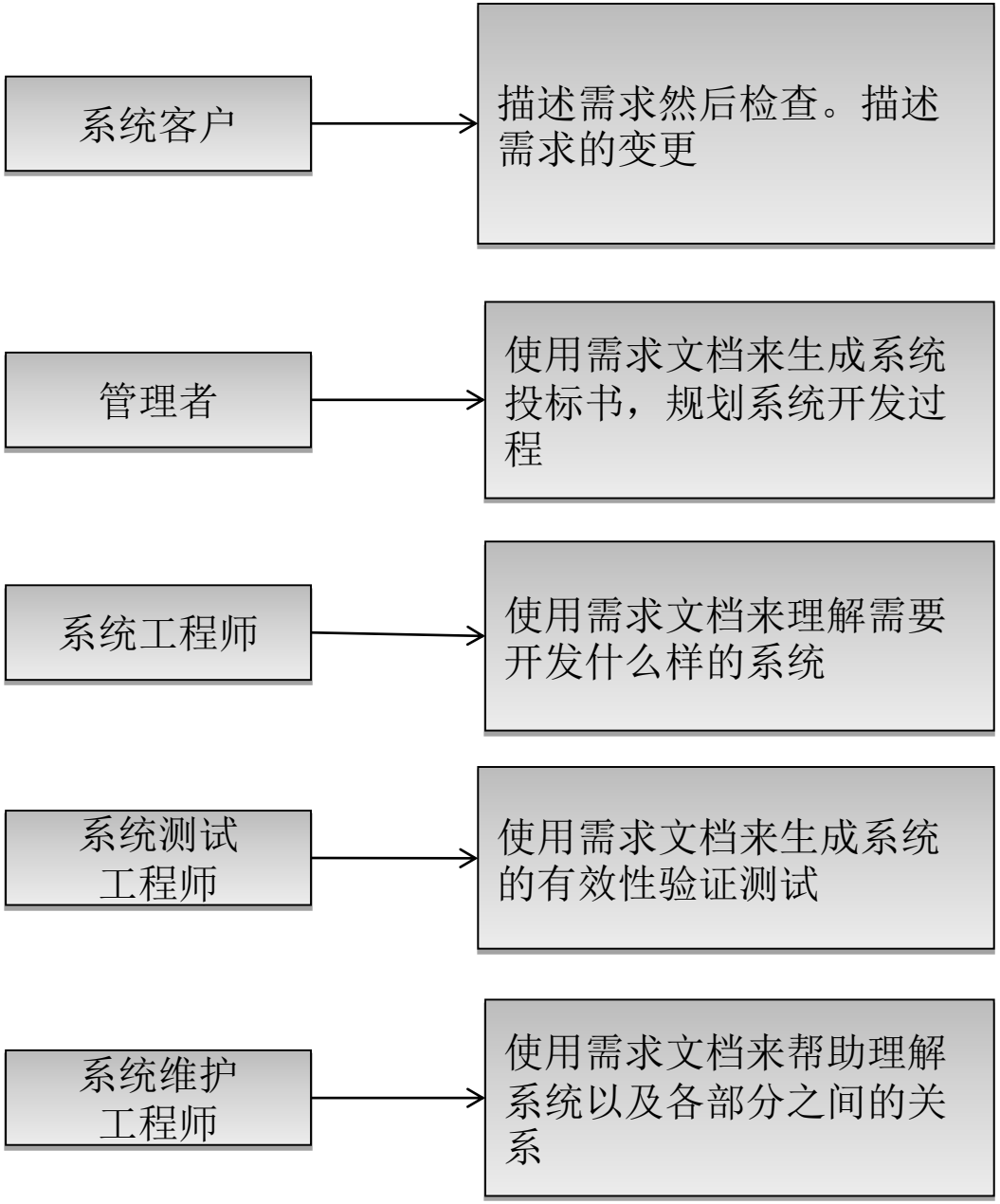
```
interface PrintServer {

// defines an abstract printer server
// requires:          interface Printer, interface PrintDoc
// provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter

        void initialize ( Printer p ) ;
        void print ( Printer p, PrintDoc d ) ;
        void displayPrintQueue ( Printer p ) ;
        void cancelPrintJob (Printer p, PrintDoc d) ;
        void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;
} //PrintServer
```

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。


系统客户
描述需求然后检查。描述需求的变更
管理者
使用需求文档来生成系统投标书，规划系统开发过程
系统工程师
使用需求文档来理解需要开发什么样的系统
系统测试工程师
使用需求文档来生成系统的有效性验证测试
系统维护工程师
使用需求文档来帮助理解系统以及各部分之间的关系

需求文档的使用者

需求文档的要求

- 描述系统外部行为
- 描述实现上的约束
- 容易改变 结构化、去耦合
- 成为系统维护人员的参考工具
- 记录系统的整个生命周期，即预测变更
- 对意外事件作出可接受的反应

主要指变更

IEEE 需求标准

- 引言
- 一般描述
- 专门需求
- 附录
- 索引
- 这是应该实例化到一个特定系统的通用结构

需求文档结构

- 引言
- 术语
- 用户需求定义
- 系统体系结构
- 系统需求描述
- 系统模型
- 系统进化
- 附录
- 索引

要点

- 需求描述了系统应该做什么以及定义系统运行和实现的约束
- 功能需求描述系统应该提供的服务
- 非功能需求包括对系统的约束和系统开发过程的约束
- 用户需求是关于系统应该做什么的高层描述

要点

- 用户需求用自然语言、表、方块图撰写
- 系统需求是为了沟通系统应该提供的功能
- 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
- 软件需求文档是经过认可的系统需求描述