第10章. 需求规格说明

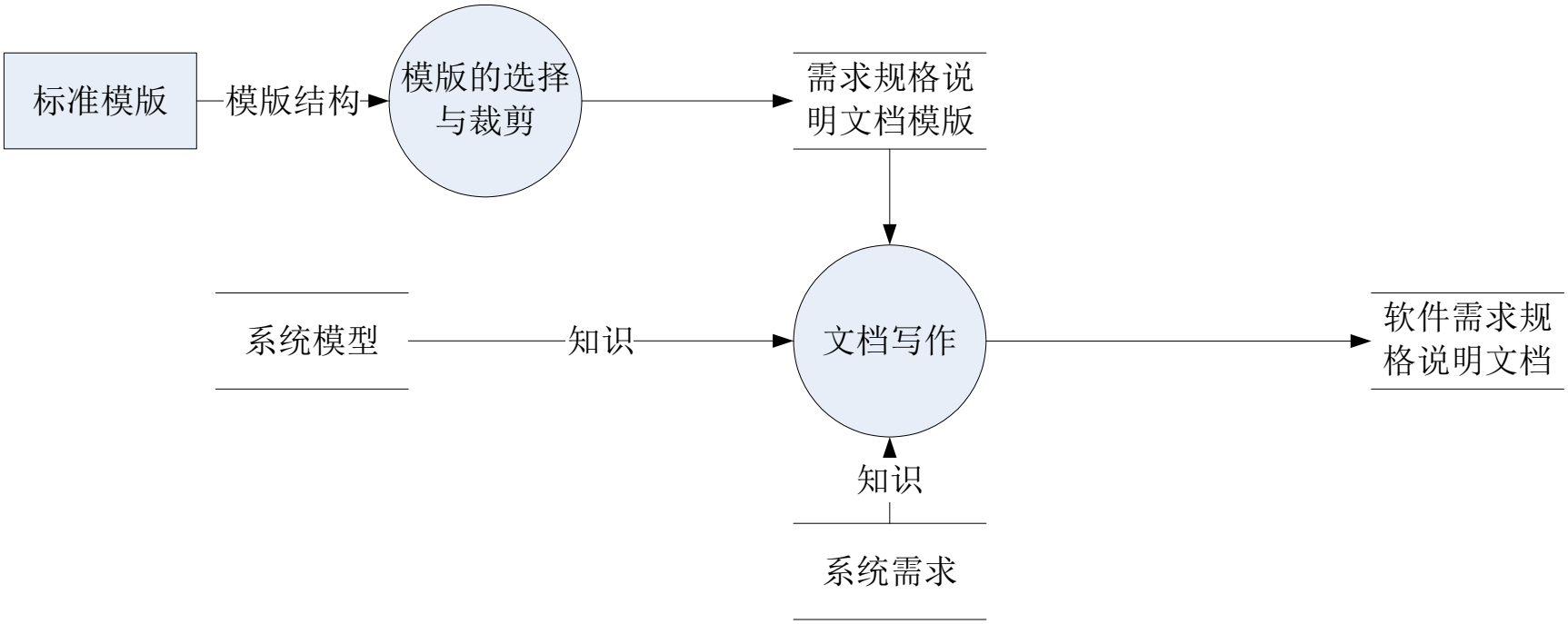
主要内容

1. 需求规格说明概述
2. 需求规格说明文档
3. 模版的选择与裁剪
4. 文档写作技巧
5. 优秀需求规格说明文档的特性

1. 需求规格说明概述
——获取 VS 分析 VS 规格说明

- 需求获取
  - 目标是得到用户需求——收集需求信息
- 需求分析
  - 目标是更深刻的理解用户需求——界定能够让用户满意的解决方案准则
- 需求规格说明
  - 目标是定义用户需求——准确描述需求及其解决方案

1. 需求规格说明概述
——需求规格说明活动



主要内容

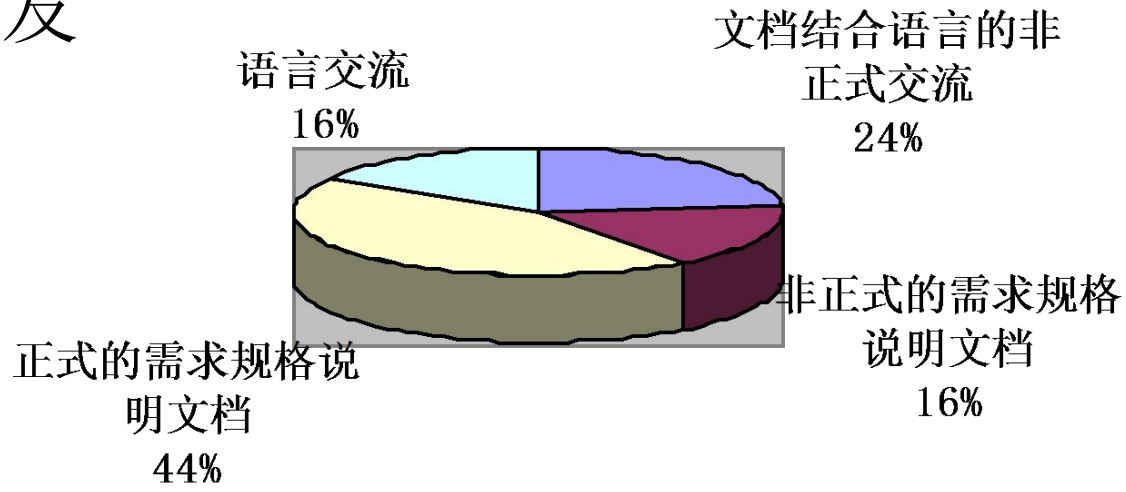
1. 需求规格说明概述
2. 需求规格说明文档
3. 模版的选择与裁剪
4. 文档写作技巧
5. 优秀需求规格说明文档的特性

2. 需求规格说明文档——作用

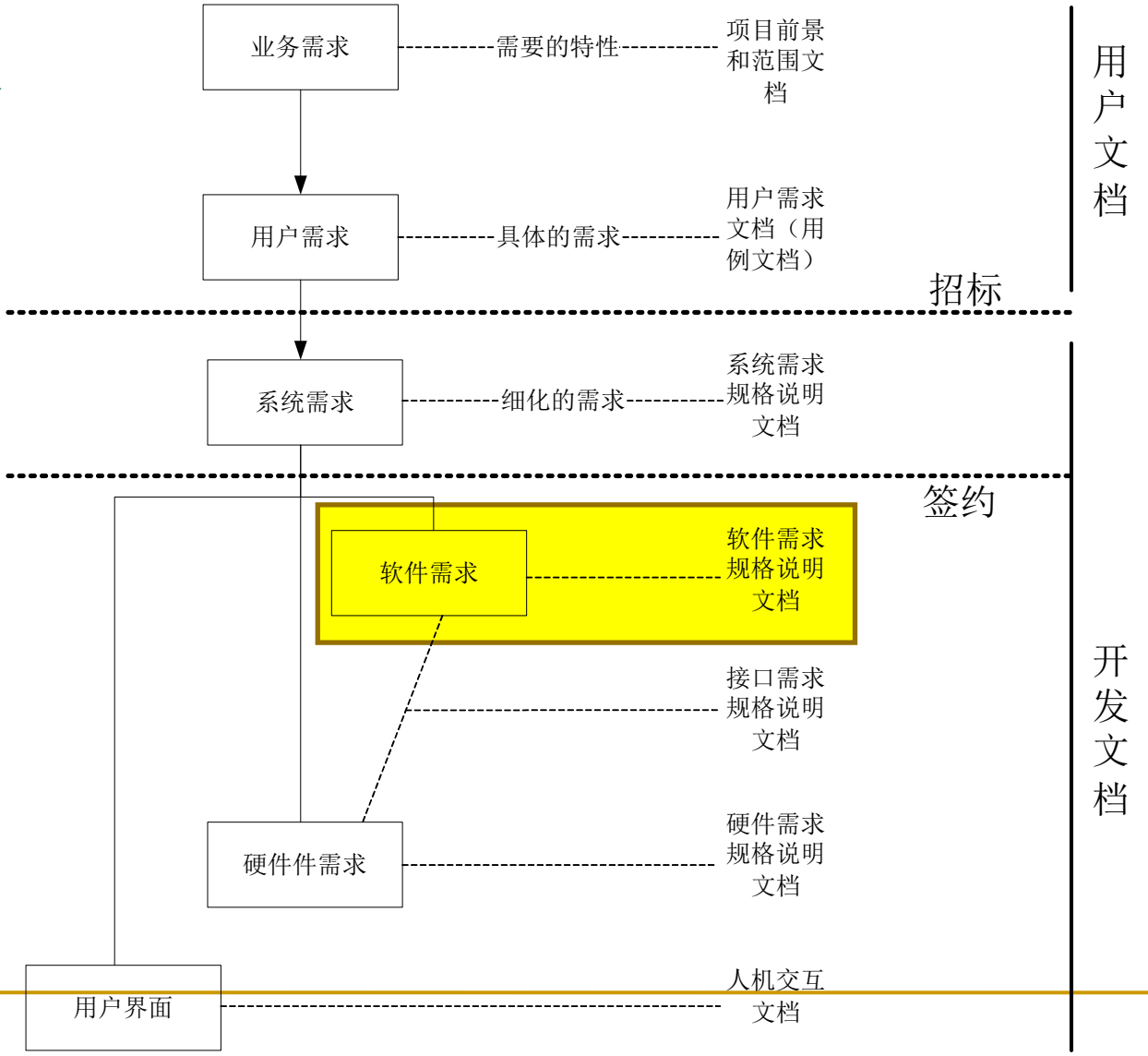
- 更好的传递软件系统的需求信息和解决方案给所有的开发者
- 拓展人们的知识记忆能力
- 作为合同协议的重要部分
- 作为项目开发活动的一个重要依据
- 发现和减少可能的需求错误，减少项目的返工，降低项目的工作量
- 作为有效的智力资产

2. 需求规格说明文档——忽视的原因

- 交流途径
- 时间压力
- 迭代式开发
  - 敏捷

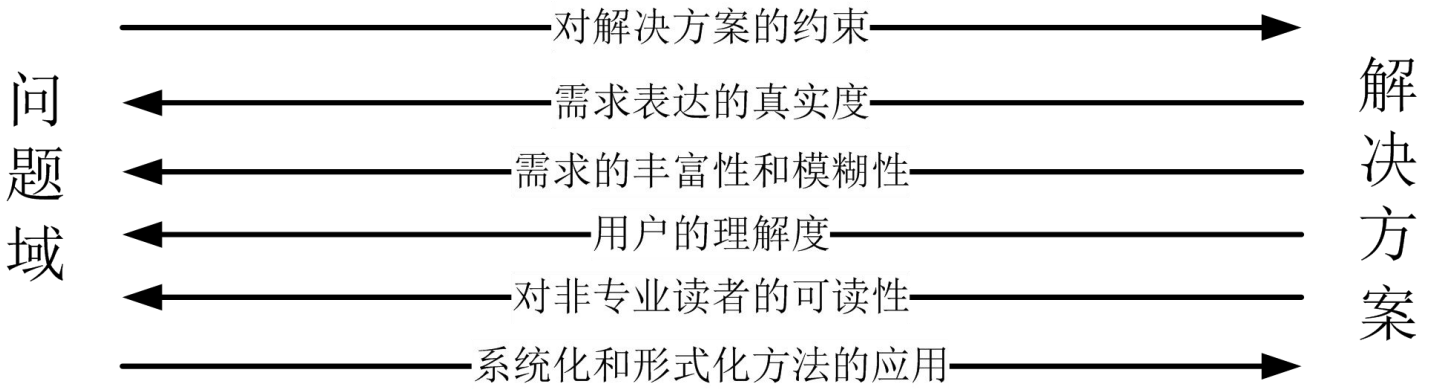


2. 需求规格说明文档——类型



2. 需求规格说明文档——类型

项目前景和范围文档	用户需求文档	系统需求规格说明文档	软件需求规格说明文档 硬件需求规格说明文档 接口需求规格说明文档 人机交互文档



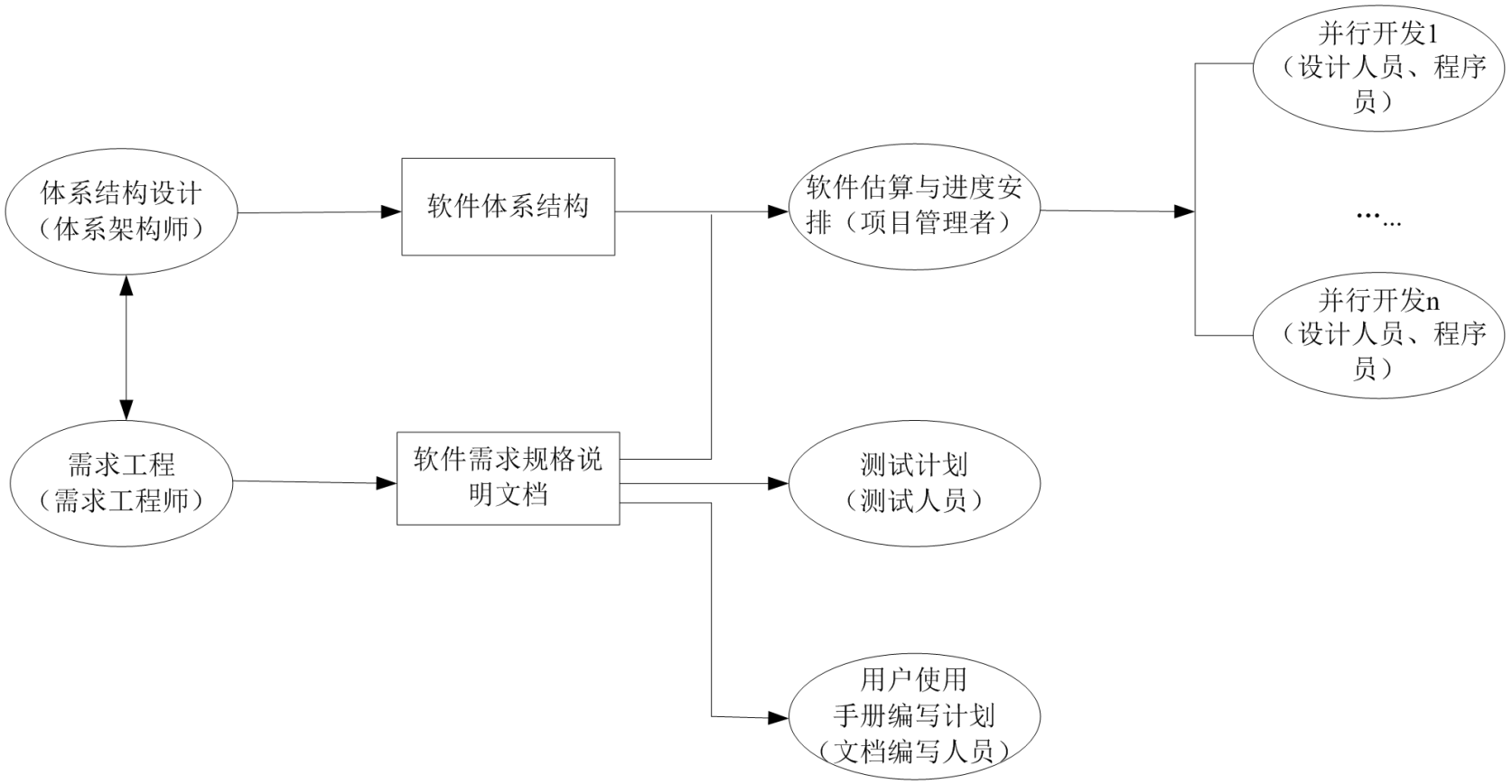
2. 需求规格说明文档——内容

- 前景和范围内
  - 问题域信息
  - 解决方案
  - 需求

2. 需求规格说明文档——作者

- 项目管理者
  - 组织安排、提供条件
- 需求工程师
  - 负责人、主导人
- 文档写作人员
  - 有时会采用，节省需求工程师的时间
- 涉众（用户）
  - 验证人

2. 需求规格说明文档
——读者



2. 需求规格说明文档 —— 手段

- 非形式化
  - 自然语言
  - 限制性文本
- 半形式化
  - 结构化文本
    - 伪码/结构化英语
  - 模型语言
    - 图、表…
- 形式化
  - 形式化语言
    - 数学语言：BNF，Z…

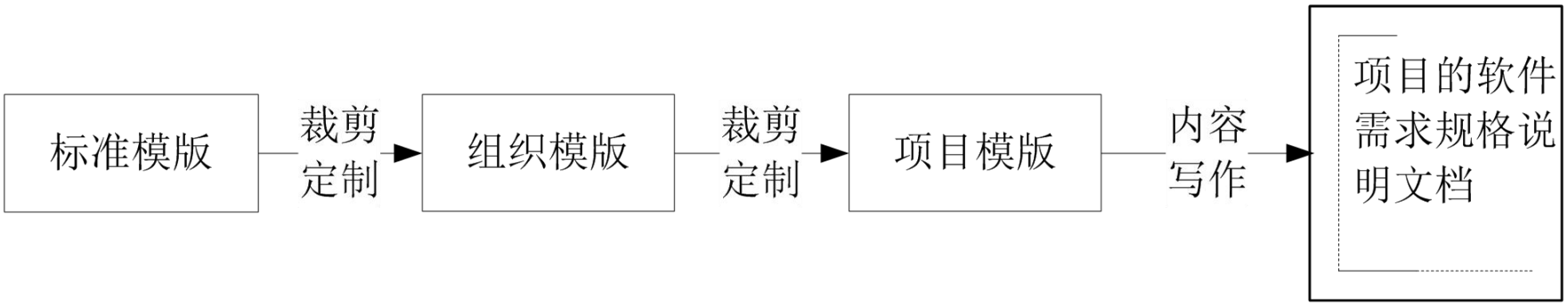
主要内容

1. 需求规格说明概述
2. 需求规格说明文档
3. 模版的选择与裁剪
4. 文档写作技巧
5. 优秀需求规格说明文档的特性

3. 模版的选择与裁剪
——动机

- 优秀的文档
  - 结构组织
    - 复用：模版
      - 选择与裁剪
  - 文字写作
    - 字词、句法
      - 写作技巧

3. 模版的选择与裁剪



3. 模版的选择与裁剪

1. 引言
- 1.1 目的
- 1.2 范围
- 1.3 定义、首字母缩写和缩略语
- 1.4 参考文献
- 1.5 文档组织

2. 总体描述
- 2.1 产品前景
- 2.2 产品功能
- 2.3 用户特征
- 2.4 约束
- 2.5 假设和依赖

3. 详细需求描述
- 3.1 对外接口需求
  - 3.1.1 用户界面
  - 3.1.2 硬件接口
  - 3.1.3 软件接口
  - 3.1.4 通信接口
- 3.2 功能需求
  - 3.2.1 系统特性1
    - 3.2.1.1 特性描述
    - 3.2.1.2 刺激/响应序列
    - 3.2.1.3 相关功能需求
      - 3.2.1.3.1 功能需求1.1
      - …
      - 3.2.1.3.*n* 功能需求1.*n*
  - 3.2.2 系统特性2
  - …
  - 3.2.*m* 系统特性*m*
- 3.3 性能需求
- 3.4 约束
- 3.5 质量属性
- 3.6 其他需求

附录

索引

示例

1. 引言
- 1.1 目的
- 1.2 文档约定
- 1.3 读者对象和阅读建议
- 1.4 项目范围
- 1.5 参考文献

2. 总体描述
- 2.1 产品前景
- 2.2 产品特性
- 2.3 用户类及其特征
- 2.4 运行环境
- 2.5 设计和实现上的约束
- 2.6 用户文档

3. 系统特性
- 3.1 系统特性X
  - 3.x.1 描述和优先级
  - 3.x.1 刺激/响应序列
  - 3.x.3 功能需求

4. 对外接口需求
- 4.1 用户界面
- 4.2 硬件接口
- 4.3 软件接口
- 4.4 通信接口

5. 其他非功能需求
- 5.1 性能需求
- 5.2 安全性需求
- 5.3 软件质量属性

6. 其他需求

附录A： 术语表

附录B： 分析模型

附录C： 待确定问题清单

3. 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 模式1

3.2.1.1 功能需求1.1

…

3.2.1.*n* 功能需求1.*n*

3.2.2 模式2

…

3.2.*m* 模式*m*

3.2.*m*.1 功能需求*m*.1

…

3.2.*m*.*n* 功能需求*m*.*n*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 详细需求描述

3.1 功能需求

3.1.1 模式1

3.1.1.1 对外接口需求

3. 1.1.1.1 用户界面

3. 1.1.1.2 硬件接口

3. 1.1.1.3 软件接口

3. 1.1.1.4 通信接口

3. 1.1.2 功能需求

3.1.1.2.1 功能需求1.1

…

3.1.1.2.*n* 功能需求1.*n*

3. 1.1.3 性能需求

3.1.2 模式2

…

3.1.*m* 模式*m*

3.2 约束

3.3 质量属性

3.4 其他需求

3. 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 用户类1

3.2.1.1 功能需求1.1

…

3.2.1.*n* 功能需求1.*n*

3.2.2 用户类2

…

3.2.*m* 用户类*m*

3.2.*m*.1 功能需求*m*.1

…

3.2.*m*.*n* 功能需求*m*.*n*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 类/对象

3.2.1类/对象1

3.2.1.1 属性（直接的或继承的）

3.2.1.1.1 属性1

…

3.2.1.1.n 属性n

3.2.1.2 功能（服务、方法，直接的或继承的）

3.2.1.2.1 功能1

…

3.2.1.2.m 功能m

3.2.1.3 消息（收或发）

3.2.2类/对象2

…

3.2.*p*类/对象*p*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 刺激因素1

3.2.1.1 功能需求1.1

…

3.2.1.*n* 功能需求1.*n*

3.2.2 刺激因素2

…

3.2.*m* 刺激因素*m*

3.2.*m*.1 功能需求*m*.1

…

3.2.*m*.*n* 功能需求*m*.*n*

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.1 信息流

3.2.1.x 数据流图x

3.2.1.x.1 数据实体

3.2.1.x.2 相关处理

3.2.1.x.3 拓扑结构

3.2.2 处理描述

3.2.2.m 处理m

3.2.2.m.1 输入数据实体

3.2.2.m.2 处理的算法或规则

3.2.2.m.3 被影响的数据实体

3.2.3 数据结构描述

3.2.3.p 结构p

3.2.3.p.1 记录类型

3.2.3.p.2 字段构成

3.2.4 数据字典

3.2.4.q 数据元素q

3.2.4.q.1 名称

3.2.4.q.2 表示法

3.2.4.q.3 单位/格式

3.2.4.q.4 精确度/准确度

3.2.4.q.5 取值范围

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

3. 模版的选择与裁剪

3. 详细需求描述

3.1 对外接口需求

3.1.1 用户界面

3.1.2 硬件接口

3.1.3 软件接口

3.1.4 通信接口

3.2 功能需求

3.2.n 用户类别n

3.2.n.x 系统特性x

3.2.n.x.1 特性描述

3.2.n.x.2 刺激/响应序列

3.2.n.x.3 相关功能需求

3.3 性能需求

3.4 约束

3.5 质量属性

3.6 其他需求

主要内容

1. 需求规格说明概述
2. 需求规格说明文档
3. 模版的选择与裁剪
4. 文档写作技巧
5. 优秀需求规格说明文档的特性

4. 文档写作技巧——原则

- 写作是一门艺术
  - 没有什么固定的规律
  - 有一些效用有限的经验原则
    - 文档的组织方式；
    - 常见情景的处理；
    - 常用的写作技巧；
    - 容易出错的地方等。
- **文档化的目标是交流**
  - 简洁、易读 VS 严格、准确
  - 不要机械的照搬某些标准和规则

有没有另外一种更容易理解的表达方式？
是否一次性提供了太多的信息？
对读者来说什么是重要的，什么是不重要的？
是否太抽象了？需不需要举例说明？
是否太专业了？需不需要解释原理？
会不会引起读者对内容的错误解释？
哪些内容有益于读者？有益于哪些读者？
文档在整体上是不是过于机械、乏味或者松散？
文档枯燥吗？令人厌烦吗？

4. 文档写作技巧
——结构组织

- 所有内容位置得当
  - 借鉴和使用标准的文档模版
- 引用或强化，但不重复
  - 引用而不是复制
  - 强化与重复
  - 引言与冗余元文本

4. 文档写作技巧
——表达方式

- 形式依赖于内容
  - 根据需要表达的内容，选择合适的表达方式
- 使用系统的表达方式
  - 人们倾向于系统的表达方式
    - 使用相同的语句格式来描述所有的细节需求。
    - 使用列表或者表格来组织独立、并列的信息。
    - 使用编号来表达繁杂信息之间的关系，包括顺序关系、嵌套关系和层次关系。

4. 文档写作技巧——细节描述

- 定义术语表或数据字典
  - 术语不一致
  - “方言”问题
  - 错误术语和冗余术语
- 避免干扰文本
  - “这一段的意思是…”
  - “上一句话是指…”
- 避免歧义词汇
  - 表15－1

歧义词汇	改进方法
可接受的、足够的	具体定义可接受的内容，说明系统怎样判断“可接受”或“足够”
大概可行的、差不多可行的	不要让开发人员来判断“大概”和“差不多”到底是否成立。应将其标记为待确定问题并标明解决日期
至少、最小、不多于、不超过	明确指定能够接受的最大值和最小值
在……之间	明确说明两个端点是否在范围之内
依赖	描述依赖的原因，数据依赖？服务依赖？还是资源依赖？等等
有效的	明确“有效”所意味的具体实际情况
快的、迅速的	明确指定系统在时间或速度上可接受的最小值
灵活的	描述系统为了响应条件变化或需求变化而可能发生的变更方式
改进的、更好的、更快的、优越的	定量说明在一个专门的功能领域内，充分改进的程度和效果
包括、包括但不限于、等等、诸如	应该列举所有的可能性，否则就无法进行设计和测试
最大化、最小化、最优	说明对某些参数所能接受的最大值和最小值
一般情况下、理想情况下	需要增加描述系统在异常和非理想情况下的行为
可选择地	具体说明是系统选择、用户选择还是开发人员选择
合理的、在必要的时候、在适当的地方	明确怎样判断合理、必要和适当
健壮的	显式定义系统如何处理异常和如何响应预料之外的操作
无缝的、透明的、优雅的	将词汇里面所反映的用户期望转化成能够观察到的产品特性
若干	声明具体是多少，或提供某一范围内的最小边界值和最大边界值
不应该	试着以肯定的方式陈述需求，描述系统应该做什么
最新技术水平的	定义其具体含义，即“最新技术水平”意味什么
充分的	说明“充分”具体包括哪些内容
支持、允许	精确地定义系统的功能，这些功能组合起来支持某些能力
用户友好的、简单的、容易的	描述系统特性，用这些特性说明词汇所代表的用户期望的实质

主要内容

1. 需求规格说明概述
2. 需求规格说明文档
3. 模版的选择与裁剪
4. 文档写作技巧
5. 优秀需求规格说明文档的特性

5. 优秀需求规格说明文档的特性

- **完备性**
  - 标准
    - 描述了用户的所有有意义的需求，包括功能、性能、约束、质量属性和对外接口。
    - 定义了软件对所有情况的所有实际输入（无论有效输入还是无效输入）的响应。
    - 为文档中的所有插图、图、表和术语、度量单位的定义提供了完整的引用和标记。
  - 前景和范围
  - TBD问题

5. 优秀需求规格说明文档的特性

- **一致性**
  - 标准
    - 细节的需求不能同高层次的需求相冲突，例如系统需求不能和业务需求、用户需求互相矛盾
    - 同一层次的不同需求之间也不能互相冲突
  - 评审
  - 自动化检查

5. 优秀需求规格说明文档的特性

- **根据重要性和稳定性分级**
  - 建立需求的优先级
- **可修改**
  - 标准
    - 它的结构和风格使得人们可以对其中任一需求进行容易地、完整地、一致地修改，同时还不会影响文档现有的结构和风格
  - 文档的可修改性要求：
    - 有着条理分明并且易于使用的组织方式，包括目录、索引和显式的交叉引用。
    - 没有重复冗余。
    - 独立表达每个需求，而不是和其他需求混在一起。

5. 优秀需求规格说明文档的特性

- **可跟踪**
  - 前向跟踪（Pre-traceability）
    - 能找到需求的来源，例如和更早期文档的显式关联。
  - 后向跟踪（Post-traceability）
    - 能找到需求所对应的设计单元、实现源代码和测试用例等，它要求每个需求都要有唯一的标识或者可供引用的名称

本章小结

- 需求规格说明定义解决方案和需求，承载需求分析的成果
- 需求规格说明是一项复杂的活动，正确的文档写作要求准确的界定文档的特性
- 掌握文档模版的裁剪技巧和文档的写作技巧，可以帮助提高需求规格说明文档写作的能力
- 优秀的需求规格说明文档需要达到一定的要求

非形式化需求描述

An elevator's direction depends on its **current direction (dir)**, the **floor** that it is on **(loc)**, and what **requests** are pending **(Req[]).**

It travels in a given direction until

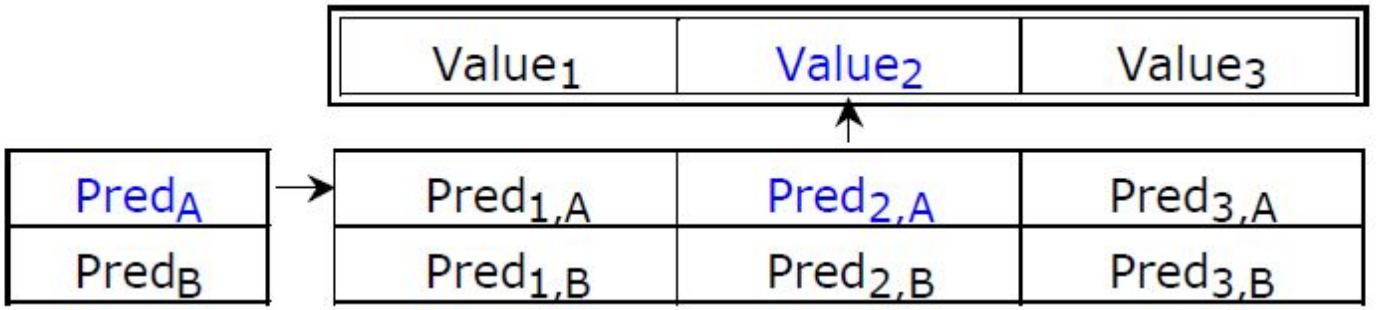
- there are **no** more pending **requests** in the **current direction**
- and there are pending **requests** in the **opposite direction**.

dir : {Up, Down}
loc: {1..n}
Req[1..n]: boolean

$$
\text{ElevDir}(\text{dir},\text{loc},\text{Req}[]) = \begin{cases} \text{Up} & (\text{dir} = \text{Up} \wedge \exists\, f.(f \geq \text{loc} \wedge \text{Req}[f])) \vee \\ & (\text{dir} = \text{Down} \wedge \neg \exists\, f.(\leq \text{loc} \wedge \text{Req}[f]) \wedge \\ & \quad \exists\, f.(f > \text{loc} \wedge \text{Req}[f])) \\ \text{Down} & (\text{dir} = \text{Down} \wedge \exists\, f.(f \leq \text{loc} \wedge \text{Req}[f])) \vee \\ & (\text{dir} = \text{Up} \wedge \neg \exists\, f.(f \geq \text{loc} \wedge \text{Req}[f] \wedge \\ & \quad \exists\, f.(f < \text{loc} \wedge \text{Req}[f])) \\ \text{dir} & \text{otherwise} \end{cases}
$$

形式化需求描述---Parnas Table

- Parnas Tables use tabular constructs to organize mathematical expressions, where
- rows and columns separate an expression into cases
- each table entry specifies either the result value for some case or a condition that partially identifies some case
- Example: Inverted Table



$F_{2,A} \equiv$ if $Pred_A \wedge Pred_{2,A}$
then Result = $Value_2$

$F \equiv \cup_{j \in A,..,B} F_{i,j}$

Inverted Table

ElevDir(dir,loc,Req[]) =

	Up	Down
dir=Up	$\exists f.(f \geq loc \wedge Req[f])$	$\neg \exists f.(f \geq loc \wedge Req[f]) \wedge \exists f.(f < loc \wedge Req[f])$
dir=Down	$\neg \exists f.(f \leq loc \wedge Req[f]) \wedge \exists f.(f > loc \wedge Req[f])$	$\exists f.(f \leq loc \wedge Req[f])$

Up $(dir = Up \wedge \exists f.(f \geq loc \wedge Req[f])) \vee$
$(dir = Down \wedge \neg \exists f.(f \leq loc \wedge Req[f]) \wedge$
$\exists f.(f > loc \wedge Req[f]))$

Down $(dir = Down \wedge \exists f.(f \leq loc \wedge Req[f])) \vee$
$(dir = Up \wedge \neg \exists f.(f \geq loc \wedge Req[f] \wedge$
$\exists f.(f < loc \wedge Req[f]))$

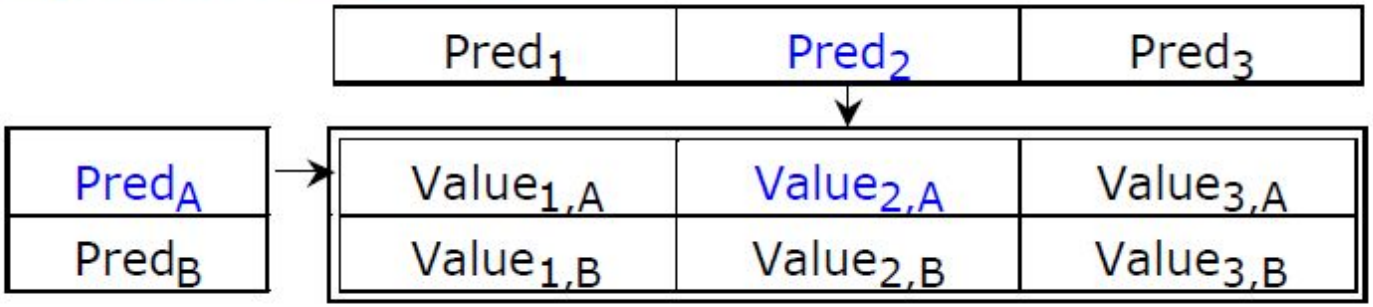
dir otherwise

Multiple Table Types

The term **Parnas Tables** actually refers to a collection of **table types** and **abbreviation strategies** for organizing and simplifying functional and relational expressions.

An expression can usually be represented in several table types. The documenter's goal is to choose (or create) a table format that produces a **simple, compact representation** for that expression.

Example: Normal Table



$F_{2,A} \equiv$ if $Pred_A \wedge Pred_2$
then Result = $Value_{2,A}$

$F \equiv \bigcup_{j=A..B}^{j=1..3} F_{i,j}$

Normal Table

ElevDir(dir,loc,Req[]) =

∃ f.(f≤loc ∧ Req[f])	
true	*false*

∃ f.(f≥loc ∧ Req[f])	
true	dir
false	Down

dir	Up
Down	dir

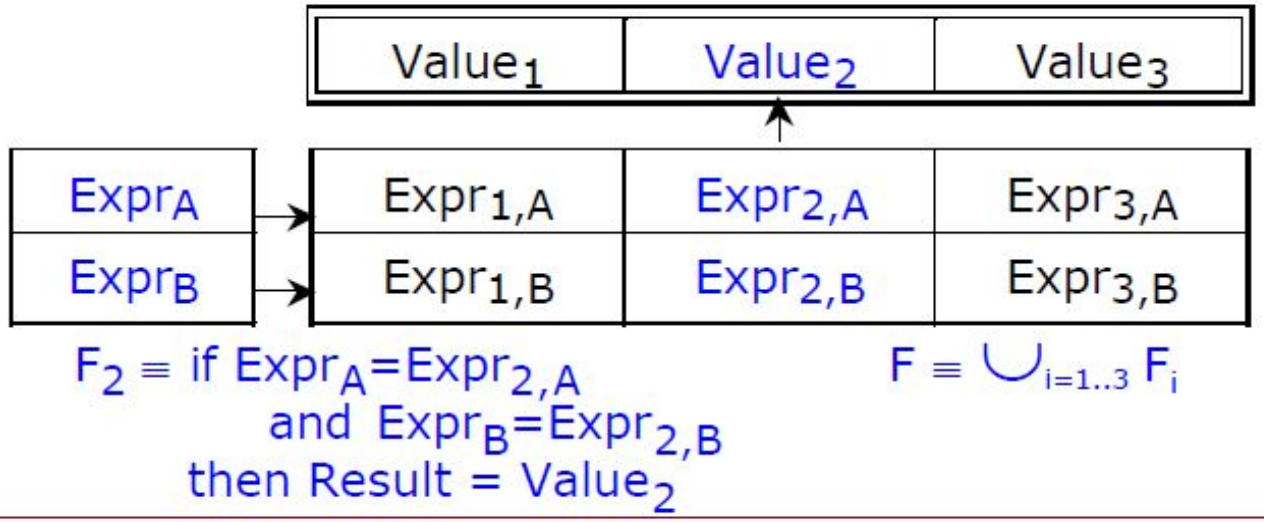
Up (dir = Up ∧ ∃ f.(f≥loc ∧ Req[f])) ∨
(dir = Down ∧ ¬ ∃ f.(f≤loc ∧ Req[f]) ∧
∃ f.(f>loc ∧ Req[f]))

Down (dir = Down ∧ ∃ f.(f≤loc ∧ Req[f])) ∨
(dir = Up ∧ ¬ ∃ f.(f≥loc ∧ Req[f] ∧
∃ f.(f<loc ∧ Req[f]))

dir otherwise

Decision Table

A **Decision Table** is useful for representing a function or relation whose **domain is a tuple** (possibly of distinct types). One dimension of the table itemizes the elements of the domain tuple.



ElevDir(dir,loc,Req[]) =

Up	Down	Down	Up

dir	Up	Up	Down	Down
∃ f.(f≥loc ∧ Req[f])	true	false	---	true
∃ f.(f≤loc ∧ Req[f])	---	true	true	false

Vector Tables

A **Vector Table** is useful for representing a function or relation whose **range is a tuple** (possibly of distinct types). One dimension of the table itemizes the elements of the range tuple.

	$Pred_1$	$Pred_2$	$Pred_3$
Var_A'	$Value_{1,A}$	$Value_{2,A}$	$Value_{3,A}$
Var_B'	$Value_{1,B}$	$Value_{2,B}$	$Value_{3,B}$

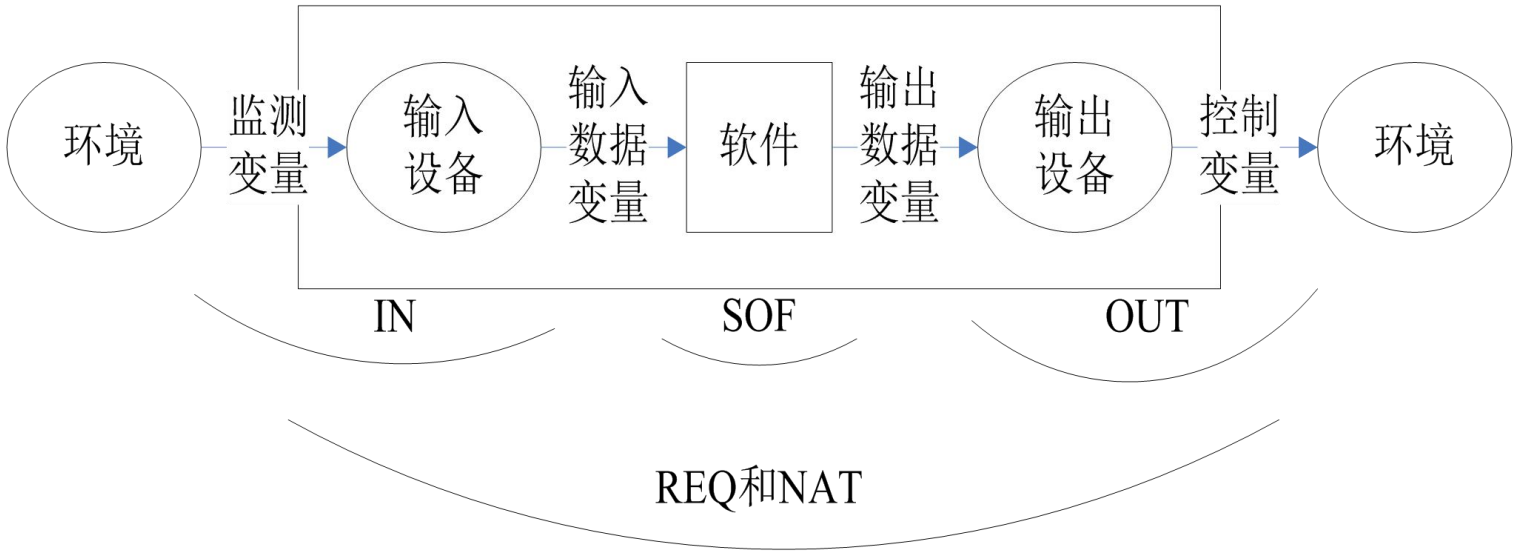
$F_{2,A} \equiv$ if $Pred_2$ then $Var_A' = Value_{2,A}$

$F \equiv \bigotimes_{i=A}^{B} \bigcup_{j=1}^{3} F_{i,j}$

Req[loc]	¬Req[loc]			
	$\neg\exists f.Req[f]$	$\exists f.(f>loc \wedge Req[f]) \wedge \neg\exists f.(f<loc \wedge Req[f])$	$\exists f.(f<loc \wedge Req[f]) \wedge \neg\exists f.(f>loc \wedge Req[f])$	$\exists f.(f<loc \wedge Req[f]) \wedge \exists f.(f>loc \wedge Req[f])$

dir'	dir	dir	Up	Down	dir
speed'	idle	idle	moving	moving	moving

形式化需求描述---四变量模型



四变量模型

- 监视变量(MonitoredVariables)：表示将会影响系统行为的，能衡量的环境属性
- 控制变量(ControlledVariables)：系统将能够控制的环境属性
- 输入数据项(InputItems)：输入设备（比如：感应器）度量被监测的属性，输入设备读取的变量被称为输入数据项
- 输出数据项(OutputItems)：输出设备设置被控制硬件的属性，输出设备写入的变量被称为输出数据项

四变量模型

- REQ，NAT，IN和OUT是四变量模型中的四个关系：
- REQ和NAT关系提供了所要求的系统行为的一种黑盒规格说明.
  - NAT关系描述对系统行为的自然约束，即，由物理法则和系统环境决定的约束;
  - REQ关系则将系统需求定义为被监视的属性和被控制的属性之间的关系，系统必须保证这些关系的可满足性。
- IN关系定义在将被监视的属性量映射为输入变量时，对被监视属性值的精度的可容忍程度.
- OUT关系定义在将输出变量映射为被控制的属性量时，对被控制属性的精度的可容忍程度。

四变量模型

- 系统需求说明希望在监视变量和控制变量之间能一直保持的关系，而软件需求则用输入和输出变量来描述。因此，如果能够首先确定系统需求（NAT关系和REQ关系），并且已知输入和输出的可容忍程度（IN关系和OUT关系），则可以准确地说明或推断出软件需求（SOF），具体来说，SOF可以通过REQ，NAT，IN和OUT推导出来。

四变量模型

- 整个系统是监视变量和控制变量之间的关系（REQ和NAT）
- 输入设备是监视变量和输入变量之间的关系（IN）
- 输出设备是输出变量和控制变量之间的关系（OUT）
- 软件系统是输入变量和输出变量之间的关系（SOF）
- REQ和NAT对应的是系统需求文档的主要内容
- IN和OUT合起来是系统设计文档的主要内容
- 输入变量和输出变量之间的关系对应的就是软件需求文档的主要内容。

四变量模型的应用实例-LCS系统

- LCS for Universitv of Kaiserslautern. Germany

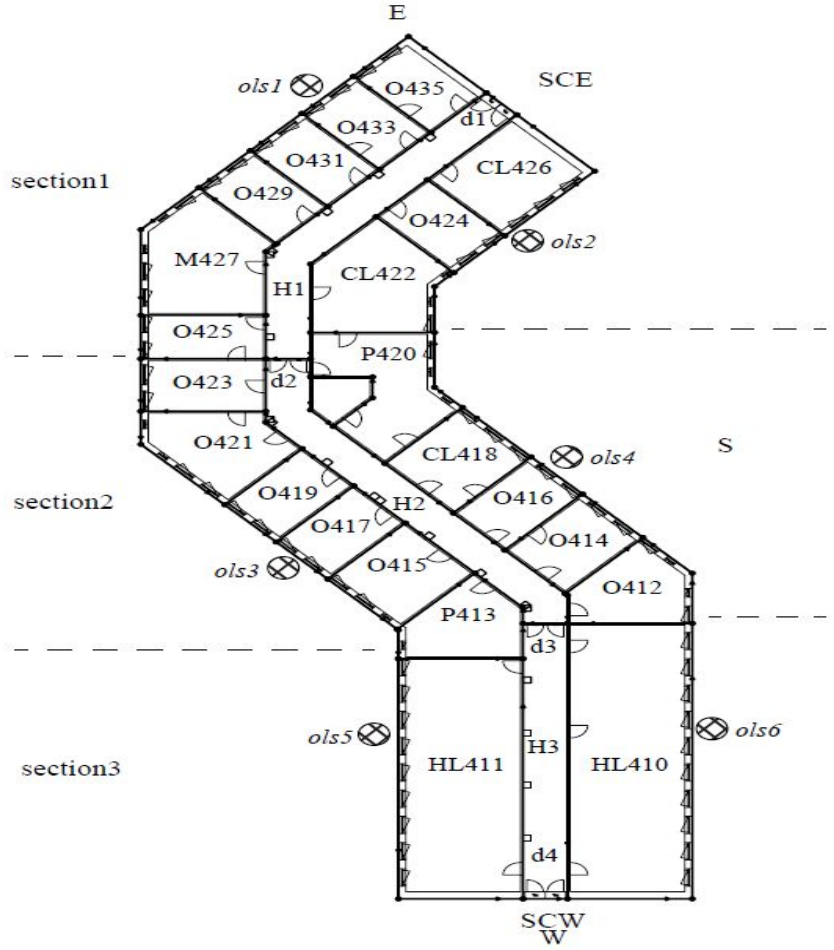
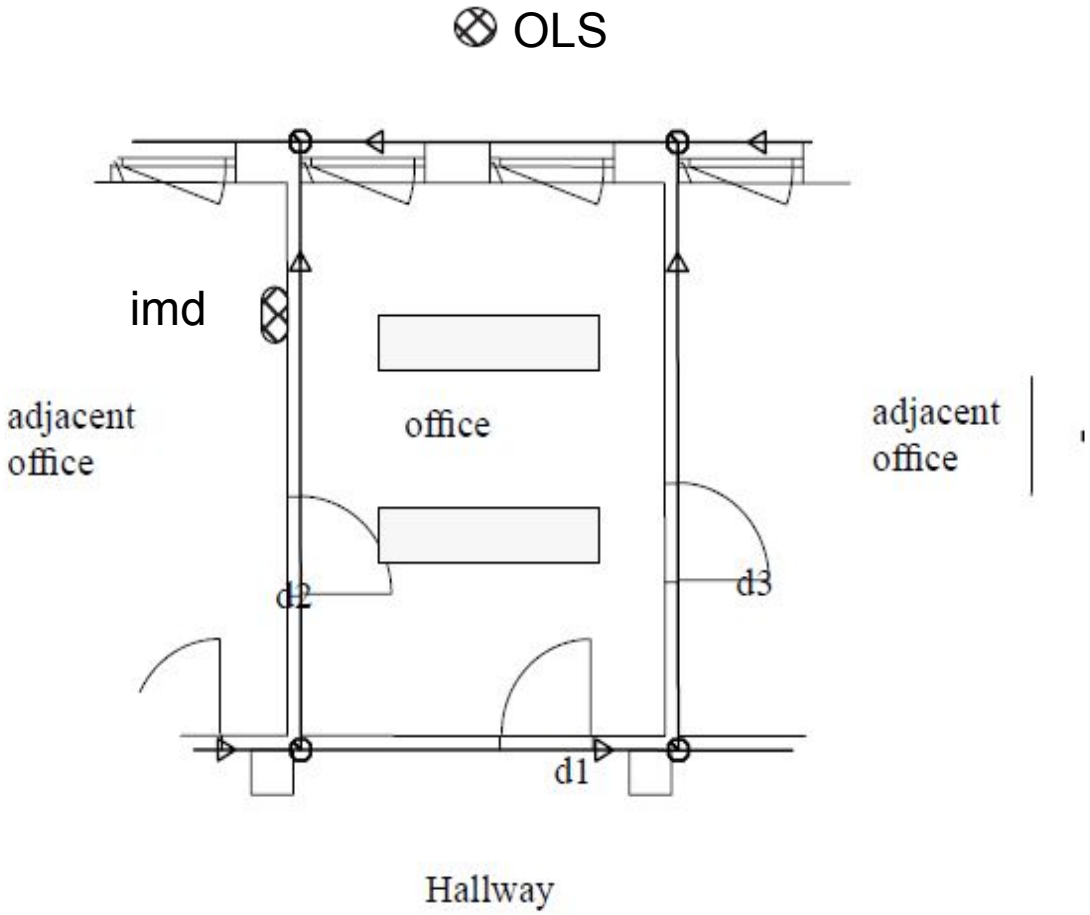


Figure 1: Architecture of the 4th Floor of Building 32

四变量模型的应用实例-LCS系统

- “The Light Control Case Study: Problem Description”



四变量模型的应用实例-LCS系统

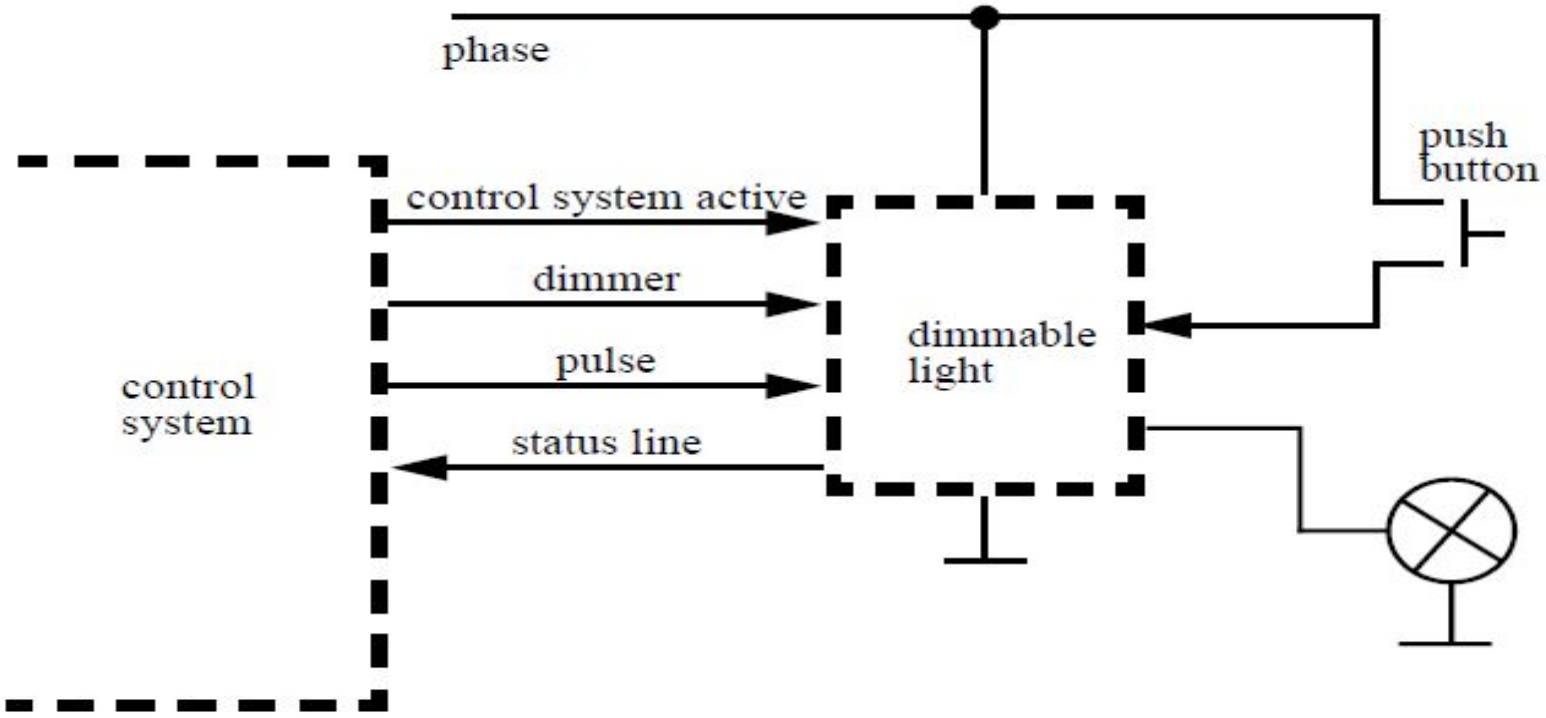
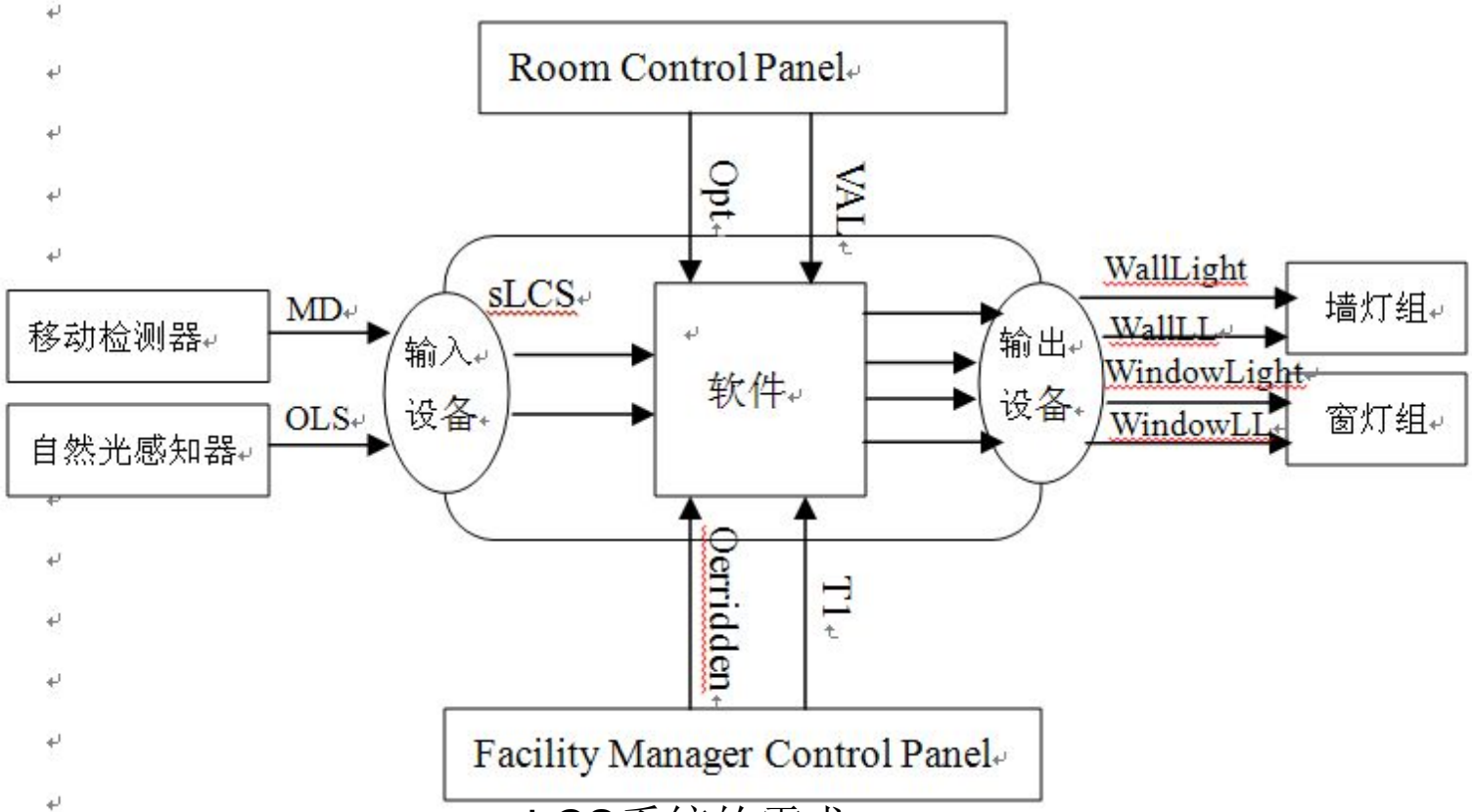


Figure 3: Dimmable Light

四变量模型的应用实例



LCS系统的需求

表1：LCS系统的监视变量表

名字	类型	初值	描述
mMD	bool	false	表示移动检测器的探测值，mMD = true表明有物体在移动
mOLS	int:[0,2000]	0	表示探测到的户外光的亮度
mLSOpt	enum of {wall, window, both}	wall	表示选择的照明模式，mLSOpt = wall，则应点亮墙灯组，mLSOpt = window，则应点亮窗灯组，mLSOpt = both，则两组等同时打开
mLSVal	int:[0,10000]	0	表示需要的照明亮度
mFMOverridden	enum of {off, on}	off	表示强制关闭按钮
mT1	int:[1,30]	10	当房间内检测不到物体移动的时间超过mT1时，系统自动关闭房间内的所有灯组。

表2：LCS系统的控制变量表

名字	类型	初值	描述
cWallLights	enum of {off, on}	off	表示墙灯组的开关控制信号
cWallLL	int:[0,10000]	0	表示墙灯组的灯光亮度
cWindowLights	enum of {off, on}	off	表示窗灯组的开关控制信号
cWindowLL	int:[0,10000]	0	表示窗灯组的灯光亮度

表3：LCS系统的输入变量表

名字	类型	初值	描述
iMD	bool	false	表示移动检测器的探测值，iMD = true表明有物体在移动
iOLS	int:[0,2000]	0	表示探测到的户外光的亮度
iLSOpt	enum of {wall, window, both}	wall	表示选择的照明模式，iLSOpt = wall，则应点亮墙灯组，iLSOpt = window，则应点亮窗灯组，iLSOpt = both，则两组等同时打开
iLSVal	int:[0,10000]	0	表示需要的照明亮度
iFMOverridden	enum of {off, on}	off	表示强制关闭按钮
iT1	int:[1,30]	10	当房间内检测不到物体移动的时间超过iT1时，系统自动关闭房间内的所有灯组。

表4：LCS系统的输出变量表

名字	类型	初值	描述
oWallLights	enum of {off, on}	off	表示墙灯组的开关控制信号
oWallLL	int:[0,10000]	0	表示墙灯组的灯光亮度
oWindowLights	enum of {off, on}	off	表示窗灯组的开关控制信号
oWindowLL	int:[0,10000]	0	表示窗灯组的灯光亮度

NAT关系

- 在四变量模型中，NAT关系表示在自然环境中一些量之间固有的一些约束、限制条件和相互关系。比如当电灯开关为打开状态时，灯泡应该是亮的。
  - 如果移动检测器探测到有人在移动，即mMD = true，则房间的灯一定是亮着的，或者是墙灯组亮着，或者是窗灯组亮着，或者两组灯都亮；
  - 如果移动检测器探测到有人在移动，即mMD = true，则房间的灯的亮度和与自然光的亮度和应该恰好等于选定的亮度值；
  - 如果移动检测器探测到有人在移动，即mMD = true，则系统操作员不能强制关灯，即按下Overridden按钮；
  - 如果相应灯组的开关是闭着的，则相应灯的亮度值应该为0；
  - 当强制关灯按钮按下时，所有灯都应关闭，且所有灯组的亮度和应该为0。

NAT关系

mMD = true	(cWallLights = on $\vee$ cWindowLights = on $\vee$ (cWallLights = on $\wedge$ cWindowLights = on) $\wedge$ (cWallLL + cWindowLL + mOLS = mLSVal) $\wedge$ mFMOverridden = off $\wedge$ ($\neg$ (cWallLights = on) $\vee$ cWallLL = 0) $\wedge$($\neg$(cWindowLights = on) $\vee$ cWindowLL = 0)
mMD = false	$\neg$ (mFMOverridden = on) $\vee$ (cWallLights = off) $\wedge$ (cWallLL = 0) $\wedge$ (cWindowLights = off) $\wedge$ (cWindowLL = 0)

REQ关系

- REQ关系则表示在待开发的软件中，为实现软件应有的功能，监测变量和控制变量之间应满足的约束关系。
  - 如果移动检测器探测到有人进入房间，即mMD = true，并且选定的照明模式是wall，则墙灯组应该是亮着的，并且墙灯组的亮度加上自然光的亮度应该等于选定的照明亮度，窗灯组应该是关闭的，且窗灯组的亮度为0；
  - 如果移动检测器探测到有人进入房间，即mMD = true，并且选定的照明模式是window，则窗灯组应该是亮着的，并且窗灯组的亮度加上自然光的亮度应该等于选定的照明亮度，墙灯组应该是关闭的，且墙灯组的亮度为0；
  - 如果移动检测器探测到有人进入房间，即mMD = true，并且选定的照明模式是both，则墙灯组、窗灯组都应该是亮着的，并且墙灯组、窗灯组的亮度加上自然光的亮度应该等于选定的照明亮度；
  - 如果移动检测器探测不到房间内有人在移动，即mMD = false，并且无人移动的时间超过mT1，则系统应自动关闭所有的灯组；
  - 如果移动检测器探测不到房间内有人在移动，即mMD = false，并且系统操作员按下了Overridden按钮，则系统应关闭所有灯组。

REQ关系

	mMD=false∧ mFMOverridden = on	mMD=false∧ mFMOverridden = off ∧ DUR(mMD=false) ≥mT1	mMD=true ∧ mFMOverridden = off
mLSOpt=wall	cWallLights=off ∨ cWindowLights= off∨ cWallLL=0∨ cWindowLL=0	同左列	cWallLights=on ∨ cWindowLights= off∨ cWallLL=mLSVal-mOLS∨ cWindowLL=0
mLSOpt=window	同上行	同上行	cWallLights=off ∨ cWindowLights= on∨ cWindowLL=mLSVal-mOLS∨ cWallLL=0
mLSOpt=both	同上行	同上行	cWallLights=on ∨ cWindowLights= on∨ cWindowLL=(mLSVal-mOLS)/2∨ cWallLL=(mLSVal-mOLS)/2

IN关系

- 由于输入设备固有的不稳定性，因此通过输入设备传递给控制系统的值常常会有一些偏差，IN关系表示的是输入变量的实际取值情况。

iMD = mMD
iOLS =[mOLS-10, mOLS+10]
iLSOpt = mLSOpt
iLSVal = mLSVal
iFMOverridden = mFMOverridden
iT1=mT1

OUT关系

- 同理，输出设备也会有传输误差，因此OUT关系表示控制变量的实际取值。

cWallLights = oWallLights
cWallLL = [oWallLL-5, oWallLL+5]
cWindowLights = oWindowLights
cWindowLL = [oWindowLL-5, oWindowLL+5]