LimKruscal 书知识整理，别指望这个给你好成绩，你只读这个大概会跟我考一个分数： 59

# 第二章

需求：

1. 用户为了解决问题或为了达到某些目标所需要的条件或能力
2. 系统或系统部件为了满足合同、标准、规范或其他正式文档所规定的要求而需要具备的条件或能力
3. 对 1 或 2 中的一个条件或一种能力的一种文档化表述

> 或
>
> 需求是人们对现实世界问题解决的期望

问题域：解决问题必须涉及的事件和事物

解系统：通过影响问题域帮助人们解决问题的软件系统

共享现象：解系统部分模拟了问题域

规格说明：以一种完全的、精确的、可验证的方法规定系统或部件的需求、设计、行为或者其他特性的文件，并经常指明判定这个规定是否满足的过程

问题域特性：问题域的背景信息

约束：不受解系统影响，却会给解系统带来极大影响的问题域特性

业务需求：针对整个业务的期望

用户需求：针对具体任务的期望

系统级需求：针对用户与系统一次交互的期望

解决方案约束：

| 约束源 | 约束 |
| --- | --- |
| 经济的 | |
| 行政的 | |
| 技术的 | |
| 系统的 | |
| 环境的 | |
| 进度与资源的 | |

# 第五章

问题分析过程：获取问题、明确问题（对问题达成共识、判断问题明确性、发现问题后的问题）、发现业务需求、定义解决方案与系统特性（定义解决方案、确定系统特性、确定约束条件）

目标分析过程：高层目标获取、低层目标获取、目标分析（精华、分解）、目标实现

需求获取困难：需求获取困难用户背景/立场不同，缺乏概括综合能力，有认知困境、越俎代庖、缺乏参与（太多/不愿/抵制/无明确用户）

如何建立活动图：上下文、行为、数据流、参与者、职责分配

# 第六章

不同系统的涉众关系：

| 系统 | 关系 |
| --- | --- |
| 小型系统 | 涉众有限且明显 |
| 组织级系统 | 分析组织内各类人群的互动关系 |
| 战略信息系统 | 分析组织内各类人群的互动关系，各种风险/机遇对既有互动关系的影响 |
| 组织间系统 | 分析组织间的互动关系，分析关系到组织间互动的各类人群在组织内的互动关系 |

涉众分析过程：涉众识别、涉众代表选择、涉众参与策略制定、涉众描述、涉众特征、涉众评估

涉众识别：先膨胀后收缩，检查列表，涉众网络

涉众代表选择：涉众采样（完整采样、态度积极、数量适中、比例恰当）、用户替代源

涉众描述：要求，输赢条件（需求冲突），力量与意愿（潜在风险），个人与工作特征（功能需求）

优先级评估：参与者、被影响者、环境设定者、观众

风险评估：弱支持者、弱反对者、强反对者、强支持者

> 化解风险，将环境设定者和被影响者转化为参与者，将弱反对者转化为弱支持者，强反对者转化为强支持者

共赢分析：Stakeholder/Issue关系图

# 第七章

用例：对系统和环境行为的局部描述

场景：相关场景集合的叙述性文本描述

用例/场景模型优缺点：场景/用例易于使用，有层次性，但为非结构化模型，不严谨，无法做到对用户需求的正确性、完备性、一致性的验证

获取流程：从前景范围出发，迭代展开，验证并维护得到的模型，用例图的包含、扩展、泛化关系，场景的生命周期

# 第八章

面谈准备：阅读背景资料、确定面谈主题和目标、选择被会见者、通知被会见者准备、确定问题和类型

群体面谈准备：确定参与人员、安排面谈时间、选择面谈地点、准备面谈内容

面谈优缺点：面谈的开展条件较为简单，经济成本较低，能够获得广泛的信息，促进友好关系，提高涉众的项目参与热情；耗时长，受地理限制，依靠人际沟通能力，知识域不同会影响面谈效果，可能会有数据偏差。

面谈记录内容：事实和问题，对象的观点和感受，组织和个人的目标

群体面谈优缺点：节约时间，节约成本；比一对一组织困难得多。

问题类型：开放式问题（更加丰富，但容易有不相干细节，容易失控）、封闭式问题（切中要点，但会使人厌烦）、程序性提示、探究式问题、诱导式问题、元问题

问题准备过程：前期开放式问题为主，决策层与专家为主，遵循问题到目标到解决方案路线，分析涉众特点（角色、任务、个人目标、频率、优先级），后期封闭式问题为主，抓住主题和线索，问题针对性，事先准备面谈材料

问题准备：问题-目标-特征，角色-任务，任务-任务特征，开放-封闭（同上）

面谈主体：礼貌，倾听，握手，控制，探究，观察（记录），道具

面谈结构：结构化面谈（事先有问题准备）、半结构化面谈（灵活变动）、非结构化面谈（开放）

> 非结构化面谈也有准备

# 第九章

原型：将未来事物所含的知识带回到现在进行展示，用于解决不确定性、变更和模糊问题

用途分类的原型：演示原型（展示可行性）、严格意义上的原型（探索和解决不确定性，针对需求）、试验原型（试验和解决不确定性，针对设计构造）、引示系统原型（用来增长，针对迭代式开发）

原型法流程：确定原型需求、原型开发、原型评估、原型修正

原型类型：探索式（以缺陷式需求开始，既而不断调整和修正需求）、实验式（有明确需求，但可行性未知，需要定义评估方法）、演化式（持续开发的一部分）

成本控制：快速构建抛弃式原型、控制水平原型成本（人机交互、功能、实现）、使用简单介质降低成本

开发原型类别：水平原型（实现某个特定层次）、垂直原型（选定功能实现所有层次）

# 第十章

观察：用于解决情景性问题，常见有采样观察和民族志

观察解决的问题：理解复杂的协同事件（具有突出的情景性）、获取工作中的异常处理、获取与用户认知不一致的实际知识、了解用户的认知、获取默认知识

采样观察：

| 方法 | 优点 | 缺点 | 场景 |
| --- | --- | --- | --- |
| 时间采样 | 1. 通过随即观察减少偏差<br>2. 对频繁发生事件取代表观察 | 1. 分段收集数据不能提供全面信息<br>2. 漏掉不经常发生却很重要的事件 | 1. 发现异常流程<br>2. 验证用户知识和实际工作的一致性 |
| 事件采样 | 1. 允许在行为展开中观察<br>2. 允许对指定重要事件进行观察 | 1. 消耗大量时间<br>2. 漏掉频繁发生的代表性样本 | 1. 获取默认知识<br>2. 验证用户知识和实际工作的一致性 |

民族志：应用于复杂的协同问题，优点是深度理解信息，并且让社会性因素可见化，缺点是耗时，调研结果很难传递到开发过程

文档审查：需求重用（需求规格说明）、文档分析（硬数据）、需求剥离（需求文档）

# 第十一章

Wieringa框架和Zachman框架启示：

Wieringa框架和Zachman框架都是对需求分析技术的综述

Wieringa通过描述系统与外界环境的交互将交互描述分为功能式描述、通信式描述、行为式描述，从功能到通信到行为的逐步精华是对系统功能的展开。另一方面框架对系统进行分解，将系统内外区分为：外部功能、外部通信、外部行为、概念组元、组元功能、组元通信、组元行为

Zackman认为系统开发是由不同观点的若干人员共同完成的，因此是从人角度进行的系统分解。Zackman的行分别是目标/范围（规划者视图）、企业模型（所有者视图）、系统模型（设计师视图）、技术模型（构建者视图）、组件模型（集成者视图）、实际运行的系统，列分别是What、How、Where（network）、Who、When、Why，分析技术就是用来对企业建模各列进行建模和描述的技术，其中面向对象分析的用例图、OCL、交互图、有限状态机、类图皆可以从企业建模中找到

这些综述更好的使得需求分析更有结构性

# 第十四章

如何建立领域模型：发现对象和类（概念类分类列表、名词分析、行为分析）、确定概念类、建立关联、添加类的重要属性

如何建立交互图：确定上下文、找出对象、建立框架、添加描述交互行为

> 系统交互图要剔除非系统交互的

如何建立状态图：确定上下文、识别状态、建立状态转换表、补充信息

> 注意可能有层次状态机

OCL模板：操作+函数名、引用+用例、前置条件、后置条件

# 第十五章

术语表的作用：避免

1. 术语不一致
2. 方言问题
3. 错误术语和冗余术语

如何保证需求的完备性、一致性、正确性：

定义：

需求的完备性：优秀的需求是完备的，即需求描述了开发人员设计和实现这项功能所需的所有信息。需求开发结束之前所有的TBD需求都被解决。

正确性：每一项需求都必须正确描述所需要的系统功能，要真实的反映用户的意图。在需求传递给开发人员之前，必须得到足够的重视。

一致性：细节的需求不能同高层次的需求冲突，同一层次的不同需求之间不能互相冲突。

做法：

建立目标模型、利用用例/场景模型进行需求获取、面向对象需求分析

需求规格的优秀特性：

1. 完备性：描述了用户的所有异议的需求，包括功能、性能、约束、质量属性、对外接口，每一条需求都是完备的，定义了软件对所有情况的所有实际输入，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。
2. 一致性：细节的需求不能同高层次的需求相冲突，同一层次的不同需求之间也不能互相冲突
3. 根据重要性和稳定性分级：优先级
4. 可修改：任意需求进行容易的、完整的、一致的修改，不会影响文档结构风格
5. 可跟踪：来源清晰
6. 正确性：每一项需求都必须正确描述所需要的系统功能，要真实的反映用户的意图。在需求传递给开发人员之前，必须得到足够的重视。

对文档化的3种手段——非形式化、半形式化和形式化看法：

信息的描述语言可以分为三种类别：

非形式化语言，即自然语言。

半形式化语言，比自然语言具有更丰富的语义和更严格的语法同时又没有严格到可以完全基于数学方法的语言，例如ER图、DFD图、UML等图形语言。

形式化语言，基于数学的语言，例如VDM、Z语言等。

自然语言具有复杂的规则和多样化的表达方式，所以它的表达能力最为强大。这使得它无法被机器所理解，它所描述的信息内容也无法准确的映射为机器行为。

形式化语言是基于数学方法的语言，具有数学的表示法特性。使用形式化语言描述的信息内容是可以进行逻辑一致性推导和证明的，所以它能够保证信息的正确性。这对普通的用户而言显然要求过高，以至于大多数用户无法读懂以形式化方法描述的信息。

半形式化语言是介于自然语言和形式化语言之间的描述语言。一方面，半形式化语言具有严格的语法，定义方式比自然语言更加严格，这使得它可以避免自然语言模糊、松散、歧义、凌乱等不好的特性。另一方面，半形式化语言具有丰富的语义，使用规则比形式化语言更复杂和多样，这使得它具有比形式化方法更强的表达能力。但是，丰富的语义使得半形式化语言的语法无法严格到可以等价于数学方法的程度，所以它描述的信息还需要进行额外的处理才能够被机器所理解或者准确的映射为机器行为。同时，严格的语法限制也使得半形式语言的表达能力无法达到自然语言的程度。而且因为具有独特的语法和语义，所以半形式语言对普通用户而言无异于一门全新的语言，它所描述的信息很难被用户所理解。

为了让需求规格说明文档的内容能够同时满足用户和开发人员的需要，需求工程师在实践中更多的会综合使用自然语言、半形式化语言和形式化语言。例如，为半形式化语言和形式化语言添加自然语言的注释，或者分别使用自然语言和半形式化语言（或者形式化语言）重复描述同样的信息，或者使用半形式语言和形式化语言描述概要与抽象信息，然后再用自然语言进行详细信息的描述。

# 第十六章

多种需求验证的方法应该如何结合运用：

需求验证方法有：需求评审（同级评审，静态分析）、原型与模拟（动态行为）、开发测试用例、用户手册编制、利用跟踪关系、自动化分析

每个需求都需要进行评审，对于动态行为评审不能完成的通过原型与模拟来验证。一般过程中辅以开发测试用例、用户手册编制、利用跟踪关系等方法进行验证，从而进行高效的综合运用。

用于需求获取的原型与用于需求验证的原型有何异同：都是原型的一种方法，用于获取的原型起始于不明确的需求，侧重于可调整方案，用于验证的原型起始于比较明确的需求，侧重于功能评价方案

# 第十七章

如何处理需求的变更：

1. 认识到需求变更的必要性，并为之制定计划，计划应包括：定义明确的变更控制过程，建立变更控制的有效渠道；所有提交的需求变更请求都需要进行仔细的评估；是否变更决定应该有变更控制委员会统一做出；必须对变更的实现结果进行验证；需求的变化情况要及时的通知到所有会受到影响的项目涉众。

2. 维护需求基线，审计变更记录：有效的变更控制需要项目团队建立和维护需求基线。可以把新需求与已有的基线进行比较，确定它合适的位置以及它是否会有其他已有需求冲突。
3. 管理范围蔓延：范围蔓延是指在需求基线确定之后，再行大幅度增加新的特性、功能和需求，管理范围蔓延，并不意味着要绝对拒绝任何的范围蔓延。
4. 灵活应对变更请求：一个更加灵活的做法是和客户重新协商原先的项目约定，可能包括：加班、增加人手、推迟、删需求
5. 使用辅助工具：自动化工具能够帮助变更控制过程更有效地运作

如何对待需求的变更：

1. 认识到需求变化是不可避免的，因为问题发生了改变，环境发生了改变，需求基线本身是有缺陷的，需要为之计划
2. 维护需求基线：需求基线是已经通过正式评审的和批准的规格说明或产品，它可以作为进一步开发的基础，并且只有通过正式的变更控制过程才能修改他。维护需求基线主要为纳入配置管理和维护需求状态。
3. 控制变更：提请需求变更、接受变更请求、变更评估（表单）、变更决策（可拒绝）、执行变更、验证变更
4. 灵活应对变更：推迟时间、增加人手、加班、推迟低优先级的需求
5. 需求跟踪：通过前向跟踪和后向跟踪评估需求变更的风险