软件开发环境国家重点实验室
State Key Laboratory of Software Development Environment

需求分析（二）

如何获得真实的需求

- 你有什么样的用户？

  - 自动存取款机及其支持系统有多少种用户？
  - 5分钟时间列出所有可能用户

  - 小额存取款用户、大额存取款用户
  - 查询账户余额
  - 安全维护人员
  - 银行管理人员
  - 技术支持人员
  - 转账用户
  - 敌对分子

至少有10类用户

1. 当前银行的客户
2. 其他银行的客户 (跨行查询/取款可收手续费)
3. 值班经理、上级支行经理
4. 柜员
5. 银行数据库管理员
6. 银行安全官员
7. 银行营销团队 (理解用户的行为或投放广告)
8. 硬件和软件维护工程师
9. 银行业监管者

以及……

X. 黑客算不算一种用户？

获取需求

- 如何从这10类不同用户中获取不同的需求？
- 从不同的视角组织用户

  - 从不同的视角组织ATM用户

视角太多...

- **如何获得具体的需求**
- **会谈**
  - **封闭式会谈**
    - 被访问者回答预先定义和的问题
  - **开放式会谈**
    - 临时的讨论
  - **焦点小组 (松散定义的)**
    - 以团组的形式组织讨论
- **深入实际/体验生活**
  - **使自己沉浸到客户的实际环境和工作流程中**

如何获取需求

- 用户调查(User study)
  - 可用性调查(Usability Studies)
  - 焦点小组(Focus Groups)
  - 卡片分类(Card sorting)
  - 问卷调查(Surveys)
  - 人类学调查(Ethnographic Studies)
  - 纸上模型研究(Paper Prototype Studies)
  - 眼动追踪研究(Eye Tracking Studies)
  - 日志调查(Diary Studies)
  - 深入面谈(In-depth Interview)

用户调查方法


Usability Studies
可用性调查


Focus Groups
焦点小组


Card Sorting
卡片分类

Surveys
调查问卷


Ethnology Study
人类学调查

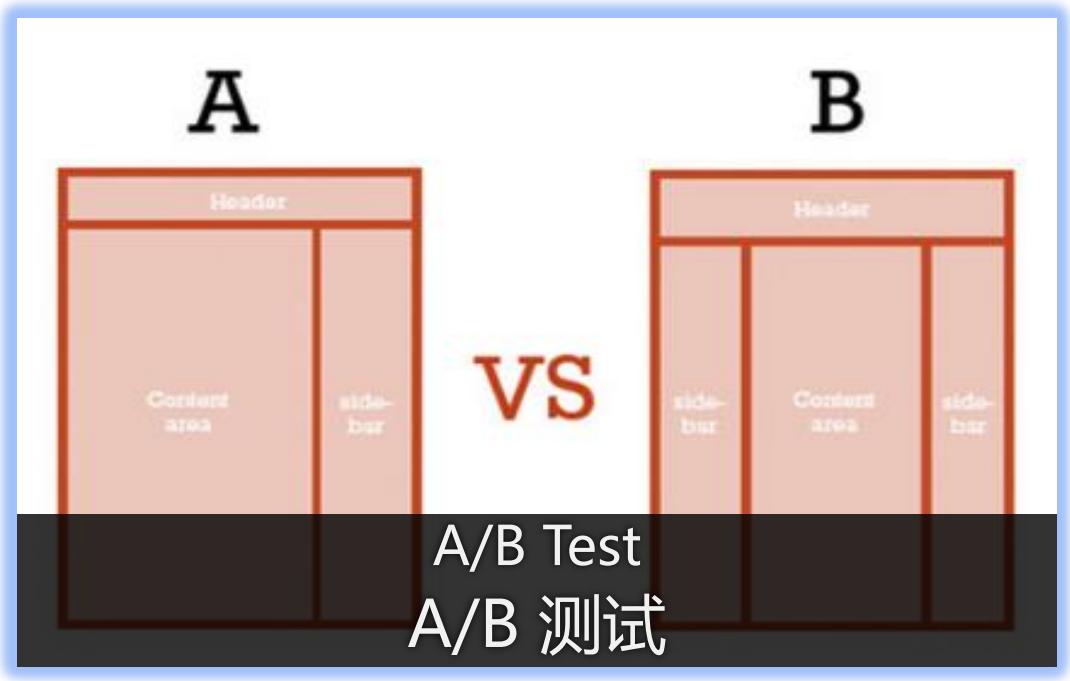

Eye Tracking Studies
眼动跟踪研究


Paper Prototype Studies
纸上模型研究

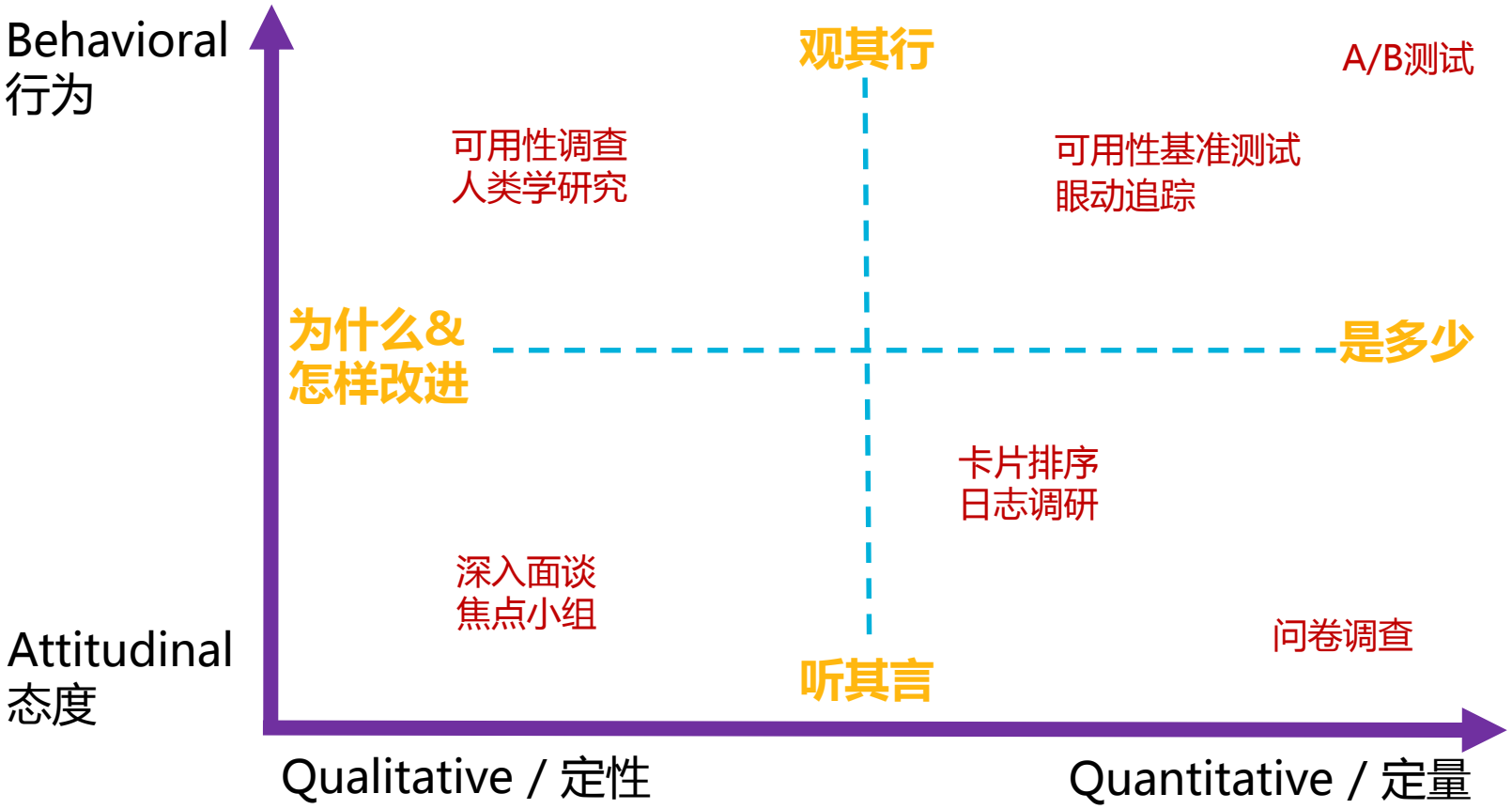
Diary Studies
日志调研


In-depth Interview
深入面谈

用户调查方法 – A/B测试


A
B
Header
Content area
side-bar
VS
Header
side-bar
Content area
side-bar
A/B Test
A/B 测试

不同方法的特点


Behavioral
行为
观其行
A/B测试
可用性调查
人类学研究
可用性基准测试
眼动追踪
为什么&
怎样改进
是多少
卡片排序
日志调研
深入面谈
焦点小组
问卷调查
听其言
Attitudinal
态度
Qualitative / 定性
Quantitative / 定量

用户有多种

- **我们说了这么多用户调查，很多人假设评价软件的就是购买软件的，就是使用软件的，但是未必。看下面的例子:**
  - a) **你要写一个中学生学习英语的软件，你找谁去做用户调查?**
    - 中学生 - 最终用户
    - 家长 - 他们是要掏钱的人，他们不会每天都用软件，有些人都不太会英语，但是他们也有需求
    - 学校老师 - 他们是有巨大影响力的人，他们说不定立下一道规矩，我们班级就用某某软件!
  - b) **你要写一个企业管理软件, 你要找谁去做用户调查?**

什么是快速调查

- **比传统的用户调查更为迅速，可以视为其敏捷版本**
  - 通常征集3个用户参加
  - 每个参与者15 – 30分钟
  - PM或设计提供测试材料
- **快速收集用户反馈，不需要过多的提前准备**
- **在没有足够时间进行完整可用性调查时，提供一种直观的检查**
- **提供一种对低优先级事项的快速可用性检测方法**
- **不是完整可用性调查的替代品**

需求规格

- The process of writing down the user and system requirements in a requirements document.
- User requirements have to be understandable by end-users and customers who do not have a technical background.
- System requirements are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

系统需求的表示法

表示法	描述
自然语言句子	使用数字编号的自然语言句子来书写需求。每个句子应当表达一条需求。
结构化自然语言	基于一个标准的表格或模板用自然语言书写需求。每个字段提供关于需求的一个方面的信息。
图形化表示法	定义系统的功能性需求，辅以文本注释的图形化模型。统一建模语言（unified modeling language，UML）用例和顺序图被广泛使用
数学规格说明	这些表示法基于有限状态机或集合等数学概念。虽然这些无二义的规格说明可以减少需求文档中的二义性，但是大多数客户不理解形式化的规格说明。他们无法检查其是否表达了他们的想法，因此不愿意将其作为系统合约接受。

需求与设计

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
  - This may be the consequence of a regulatory requirement.

自然语言规格说明

- 用自然语言书写需求，同时用图表进行辅助说明
- 自然语言表达能力强、直观、具有普适性。这意味着能被需求能被使用者和客户理解。

自然语言的问题

- 不够清晰
  - 如果不使文档变得难读，就很难做到精确
- 需求混淆
  - 功能性和非功能性的需求容易混在一起
- 需求合并
  - 不同的需求合在一起表述

写需求的指南

- 使用一种标准格式并确保所有的需求定义都遵循该格式。
- 以一致的方式使用语言。用“必须”表达必须满足的需求，用“应该”表达期望达成的需求。
- 使用强调性的文本（粗体、斜体或颜色）来突出需求中的关键部分。
- 不要假设读者理解技术性的软件工程语言。
- 尽量将每个用户需求与其原理关联起来。

胰岛素泵软件系统需求示例

3.2系统必须每10分钟测量一次血糖，如果需要的话就提供一次胰岛素。（血糖的变化相对比较慢，因此不需要更频繁的测量；测量间隔时间过长的话会导致不必要的高血糖水平）

3.6 系统必须每分钟运行一次例行的自检，测试表1定义的条件，并执行表1中所定义的相关动作。*(例行的自检可以发现硬件和软件问题并警告用户常规操作可能有问题)*

结构化规格说明

- 结构化自然语言是一种书写系统需求的方式，即使用标准的方式而不是自由文本的方式书写需求。
- 结构化适用某些种类的需求，如嵌入式控制系统，对某些需求显得死板，如商业系统

标准格式规格说明

- 对所刻画的功能或实体的描述
- 其输入以及输入来源的描述
- 关于其输出和输出目的地的描述
- 关于计算所需的信息或者所需要的系统其它实体的信息
- 采取的行动的描述
- 前置和后置条件（恰当）
- 关于该操作的副作用（如果有）的描述

胰岛素泵的需求结构化规格说明

胰岛素泵/控制软件/SRS/3.3.2

功能 计算胰岛素剂量；安全的血糖水平。

描述 当前测量的血糖水平在安全区间 3~7 个单位时，计算要供给的胰岛素的剂量。

输入 当前血糖读数（r2），前两个读数（r0 和 r1）。

来源 当前读数来自传感器。其他读数来自存储。

输出 CompDose—要供给的胰岛素剂量。

目的地 主控循环

胰岛素泵的需求结构化规格说明

动作

如果血糖水平稳定或在下降，或者虽然在升高但是上升率在下降，那么 CompDose 为 0，如果血糖水平在升高并且上升率也在增长，那么 CompDose 通过将当前血糖水平与前一血糖水平之差除以 4 并四舍五入来计算。如果结果为 0，那么将 CompDose 设为可以供应的最小的剂量。

要求

有前两个读数，这样可以计算血糖水平的变化率。

前置条件

胰岛素存储存有至少一个所允许的最大单剂量胰岛素。

后置条件 r0 被 r1 替代，而 r1 被 r2 替代。

副作用 无

表格规格说明

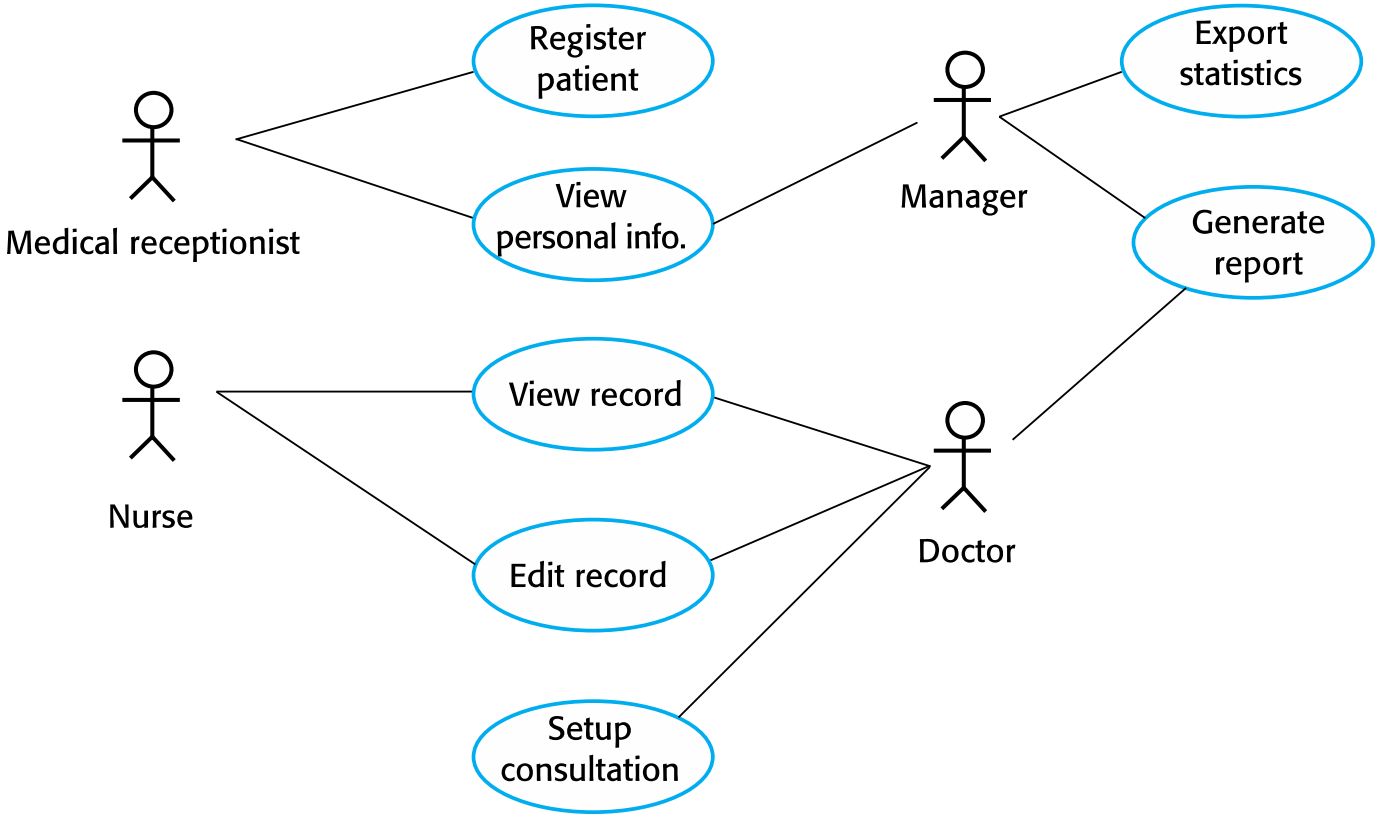
- **用于补充自然语言**
- **当存在多种可能的情况并且需要描述每种情况下要采取的动作时，表格尤其有用。**

条件	动作
血糖水平降低(r2 < r1)	CompDose = 0
血糖水平平稳 (r2 = r1)	CompDose = 0
血糖水平在升高 但上升率在下降 ((r2 – r1) < (r1 – r0))	CompDose = 0
血糖水平在升高 且上升率稳定或者在增加 ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) 如果四舍五入后为0，那么 CompDose = MinimumDose

用例

- 用例是统一建模语言（UML）的一个基本特性
- 用例识别参与一个交互的参与者（actor），并且对交互的类型进行命名
- 一组用例的集合表示系统需求中将要描述的所有可能的交互
- 可以配合顺序图，补充用例的事件顺序细节

Mentcare系统的用例



需求文档的结构

章	描述
前言	这部分定义文档所期望的的读者人群，并且描述文档的版本历史，包括创建一个新版本的原因以及对于每个版本中所作修改的总结。
引言	这部分描述系统的需要。其中应当简要描述系统的功能，并解释系统将如何与其他系统一起工作。还应当描述系统如何适应委托开发软件的组织的总体业务或战略目标。
术语表	这部分定义了文档中所用的技术术语。不应该对读者的经验或专业知识进行假设。
用户需求定义	这部分描述为用户提供的服务。非功能性系统需求也应当在这部分描述。这些描述使用客户可以理解的自然语言、图形或者其他表示法。必须遵循的产品和过程标准也应该在这里描述。
系统体系结构	这部分描述所预计的系统体系结构的高层概览，显示各个系统模块上的功能分布。复用的体系结构构件应当进行强调。

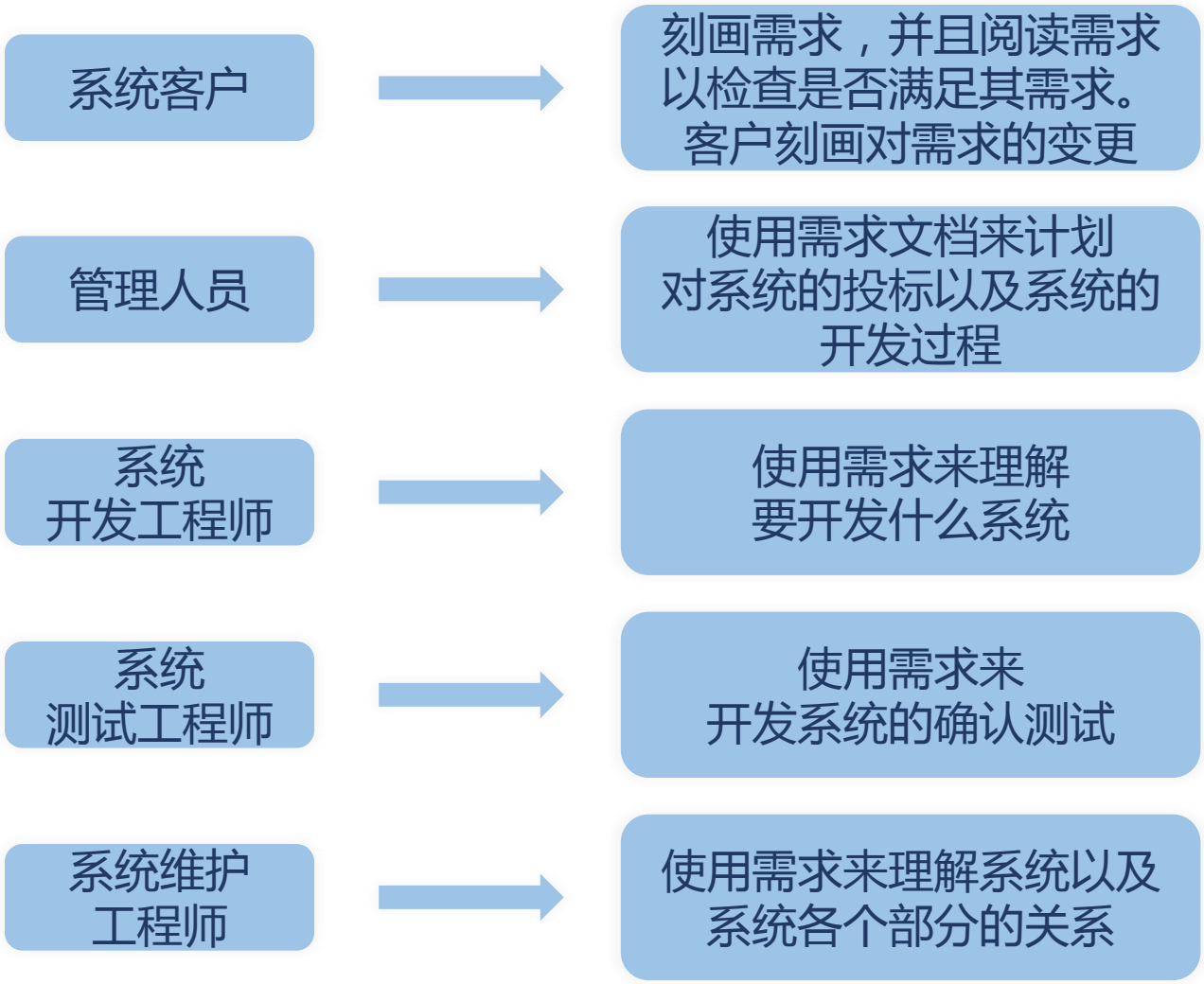
需求文档的结构

章	描述
系统需求规格说明	这部分更详细地描述功能性需求和非功能性需求。如果有必要，可以向非功能性需求中增加进一步的细节。还可以定义与其他系统的接口
系统模型	这部分包括图形化的系统模型，显示系统构件之间以及系统及其环境之间的关系。可能的模型包括对象模型、数据流模型或语义数据模型
系统演化	描述系统所基于的基本假设，以及所预计的由于硬件演化、用户需求变更等导致的变化。这部分对于系统设计者很有用，可以帮助他们避免做出会限制系统未来可能的变化的设计决策。
附录	提供与所开发的应用相关的详细、特定的信息，例如硬件和数据库描述。硬件需求定义了系统的最小配置和优化配置。数据库需求定义了系统所使用数据的逻辑组织以及数据之间的关系。
索引	可以包含几个文档索引。除了常规的字母序索引，还可以有图索引、功能索引等

需求文档

- 软件需求文档是关于系统开发者应当实现的所有东西的正式陈述。
- 需求文档应该同时包括一个系统的用户需求以及对于系统需求的详细规格说明。
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.
- 需求文档不是设计文档，理论上讲，需求文档应该指明系统需要做什么而不是如何去做。

需求文档的用户


系统客户
刻画需求，并且阅读需求以检查是否满足其需求。客户刻画对需求的变更
管理人员
使用需求文档来计划对系统的投标以及系统的开发过程
系统开发工程师
使用需求来理解要开发什么系统
系统测试工程师
使用需求来开发系统的确认测试
系统维护工程师
使用需求来理解系统以及系统各个部分的关系

需求文档易变性

- 需求文档中应当包含的详细程度取决于所开发的系统类型以及所使用的开发过程。
- 如果使用迭代化的开发过程，需求文档可以不那么详细
- Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.需求文档有一个通用的IEEE标准。这些大都可以用于大型工程项目系统的需求

延长Spec的寿命

- 记录版本修订的时间和负责人，这样出了问题好去找人
- 在Spec中要说明如何验证关于功能的描述，从Spec的描述中就能知道单元测试该怎么写，最好把测试用例也链接上
- 把Spec和测试用例、项目任务等放在一起（例如放到TTS上），相互链接
- 变化是一定会发生的，与其在Spec中有意忽略这一点，不如主动挑明哪些部分是容易发生变化的，提前做好预案。说明哪些部分如果发生改变，会有何种连锁反应
- 在做任何改动的时候，一定要事先参考Spec，事后更新Spec，团队领导人不应该在没有Spec的情况下做拍脑袋的决定

需求确认

- 需求确认是检查需求是否定义了客户真正想要的系统的过程。
- 需求错误的代价很高因此需求确认非常重要。
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.在交付之后修复需求错误可以导致100倍于实现错误的代价。

需求检查

- 正确性检查。 检查需求是否反映了系统用户的真实需要
- 一致性检查。文档中的需求不应该冲突
- 完整性检查。需求文档中的需求应当定义系统用户想要的所有功能以及约束
- 现实性检查。通过使用关于现有技术的知识，应当对需求进行检查以确保它们可以在系统预算范围内实现
- 可验证性检查。为了减少客户与承包商之间可能的争议，所描述的系统需求应当总是可验证的
- 可追踪性。需求的原始出处可追踪。
- 可适应性。如果某条需求需要变更，会不会存在大量需求同时需要调整。

需求验证技术

- 需求审评
  - 由一个评审团队系统性地分析需求
- 原型化
  - 用一个可执行的模型来检查需求
- 测试用例生成
  - 需求应该是可测试的

计划：初步的功能列表

- 对于Alpha版本发布
- 对于稳定版本发布

估计

- 合作的基础
  - 是对你和队友的任务消耗的估计
- 下述三个是需要区分的概念
  - 目标 (Target)
  - 决心 (Commitment)
  - 估计 (Estimation)

目标、决心和估计

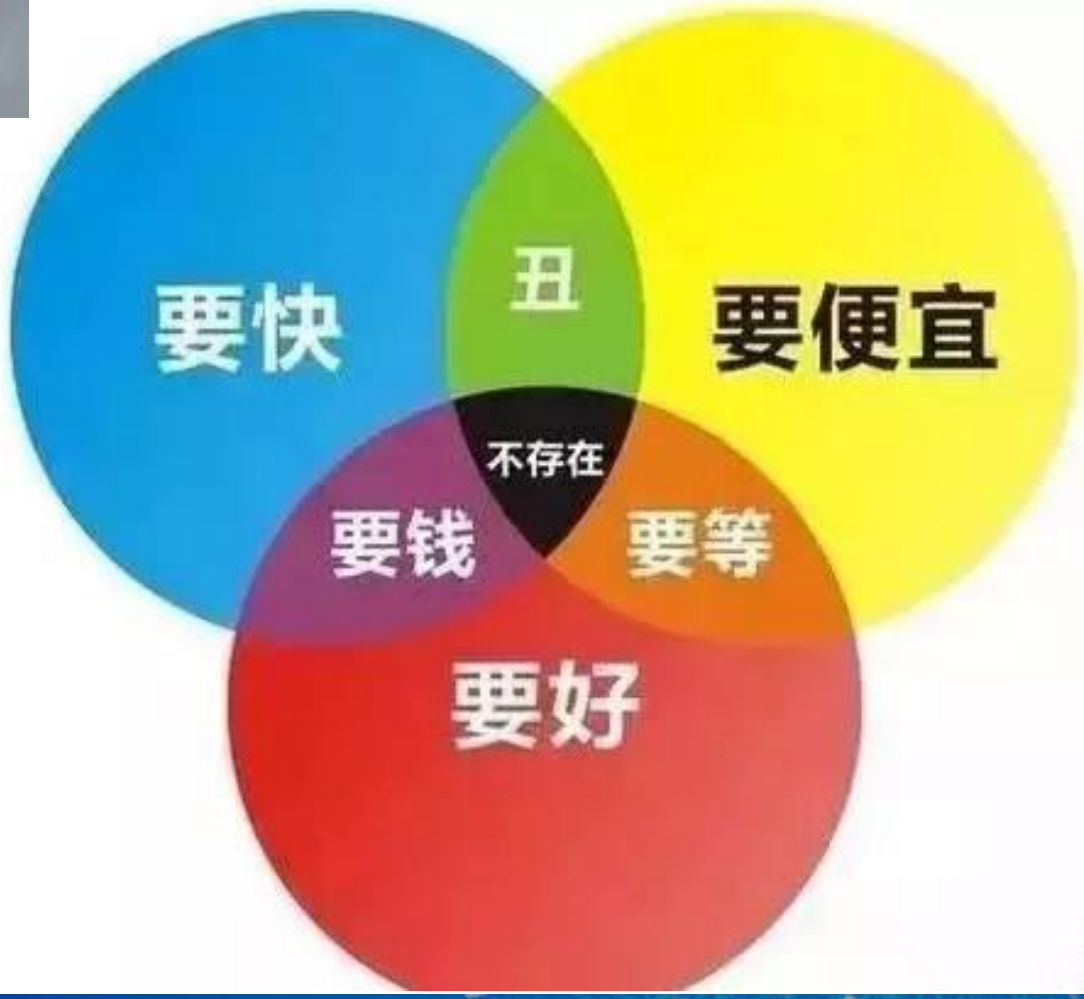
- 目标
  - 是对理想商业目标的声明
  - “我们需要在____发布v1版本”
  - 商业目标是独立于软件估计的
  - 目标是理想的或强制性的，但并不意味着它是可以实现的
- 决心
  - 是对在一个确定的时间点交付具有明确质量水平的预先定义功能的承诺
- 估计
  - 在资源有限的前提下，一个任务需要花费多长时间或者消耗多少资源

目标

- 一个项目可以有下述因素驱动
  - 时间 (必须在确定的时间点之前完成)
  - 功能 (必须在完成之前具备这些功能)
  - 预算/资源 (当所有的预算都用完时，项目结束)
- 需求(必需品)
  - 快(Fast)
  - 多、好 (Good)
  - 省 (Cheap)
- 这三件事情不是总能同时满足

你只能三选二

We offer three kinds of service:
GOOD - CHEAP - FAST
You can pick any two



我们的项目

- 时间驱动
- 时间固定的时候，能够调整的是
  - 功能 (多，好)
  - 资源 (省)
- 当我们问“给我一个估计”时
  - 需要完成工作的估计，或
  - 指出如何达成目标
  - 这两者是有区别的

软件开发估计

- 宽泛德尔菲法
  - 团队成员之间更好的交互和更多的交流
- 例子：
  - 给出一个乐观和一个悲观的估计候选
  - 北京火车站到八达岭长城花多长时间？
  - 邀请bf/gf从北航到天坛公园祈年殿花多长时间？

给估计者建议

估计有2种角色：估计者、辅助者

1. 以小时为单位估计，不要乐观也不要悲观
2. 使用一张卡片写下你的估计，另一张写下你的假设，但是只提交估计的卡片
3. 在辅助者将所有估计填入电子表格之后，请他输入你的假设
4. 讨论并对假设达成共识
5. 在一张卡片上写下你的新估计并提交
6. 重复上述3个步骤直到所有人都对所有估计总体感到满意

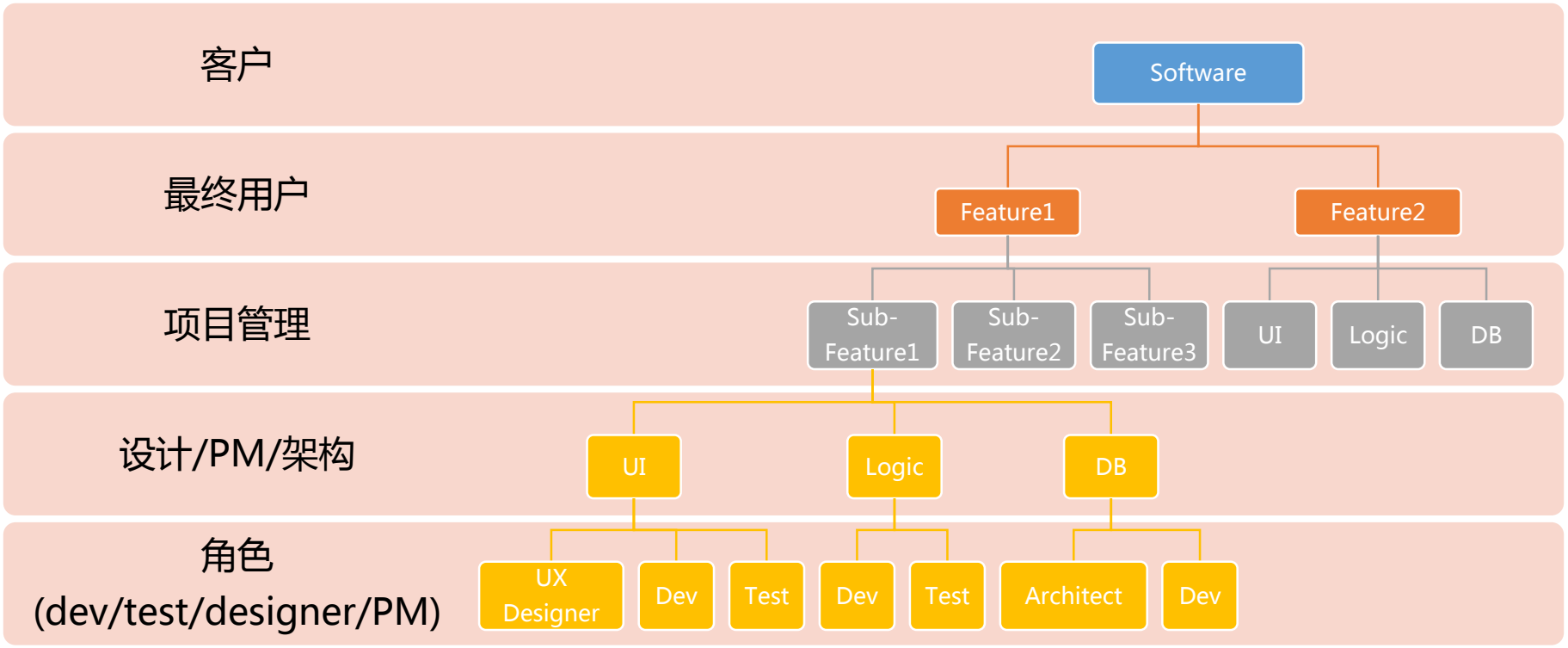
软件开发估计

- 中国陆地边界长度
- 这只是一个因素，还有一个因素的话，如何估计
  - 如果你们小组要徒步走完所有边界，你要花多长时间？
- 两个方面的因素
  - 工作量 (可以变化)
  - 你的能力和毅力 (也可以变化)

工作分解结构(WBS)

- 分治法
- 最上层 (产品)
- 中层 (功能) – 用户的角度
- 底层 (功能实现) – 团队的角度 (PM、测试人员)
- 最底层 (模块) – 开发者的角度

WBS – 不同的层次


客户
最终用户
项目管理
设计/PM/架构
角色
(dev/test/designer/PM)
Software
Feature1
Feature2
Sub-Feature1
Sub-Feature2
Sub-Feature3
UI
Logic
DB
UI
Logic
DB
UX Designer
Dev
Test
Dev
Test
Architect
Dev

估计 – 自底向上

- 估计 – 自底向上
  - 一个一个功能的估计
- 目标 – 自顶向下
  - 我们需要在DATE1发布整个软件
  - 这表明测试要在DATE2之前完成
  - 这表明编码要在DATE3之前完成
  - 这表明我们要在DATE4开始编码
  - 我们已经晚了！

谢谢！