# Requirements

# 基本概念

需求定义是对应当执行的任务的规范说明，描述系统的行为特征或属性，是一种对系统开发进程的约束。

需求工程：用工程的理念和方法来指导软件需求实践，它提供了一系列的过程、策略、方法学和工具，帮助需求工程师加强对业务或领域问题及其环境的理解，获取和分析软件需求，指导软件需求的文档化和评审，以尽可能获得准确、一致和完整的软件需求，产生软件需求的相关软件制品

需求工程特点：1.知识密集型工作，需要交叉多学科的知识2.多方共同参与3.需求获取的多种形式和源头，获取、构思、创作等，要采用多种形式和手段4.持续迭代和逐步推进，贯穿于软件整个生命周期

需求工程的方法学

抽象：如何理解和抽象软件需求？软件需求本质是什么？应采用什么样的抽象来刻画软件需求？；20世纪70年代结构化数据和数据流抽象：1.功能表现为数据处理，要理解软件功能需求，就要知道软件具有哪些数据以及要对这些数据进行什么样处理2.提出了一组抽象来表征软件的功能性需求，包括数据流、数据字典、数据加工或处理、数据源等，设计了数据流图建模语言；20世纪90年代面向对象抽象：1.软件功能表现为系统中的对象所展示的行为，要理解软件功能需求，就要知道软件有哪些对象、有什么样行为、交互和协作2.设计面向对象建模语言UML，产生了面向对象需求分析方法学

建模：如何刻画和建立软件需求模型？清晰地表达软件需求，目的是理解和交流；采用自然语言或结构化自然语言？存在描述不直观、二义性和模糊性等问题；图形化的需求建模语言？直观、易于理解，数据流图、UML图

分析：如何精化和分析软件需求？循序渐进地获得软件需求细节，进而逐步得到详细的软件需求提供策略和手段：1.指导一步步地精化和分析软件需求2.建立准确和一致的软件需求模型3.防止漏掉重要的软件需求4.发现并解决其中的问题和存在的缺陷，以保证软件需求的质量

需求获取

识别用户类别： 1.用户群的访问权限或安全级别2.用户群在业务操作中执行的任务3. 用户群使用的特性4. 用户群使用产品的频率5. 用户群在应用领域和计算机专业技能经验6. 用户群使用的平台（台式机、笔记本、平板、智能手机等）7.用户群的语言8.用户群是直接还是间接与系统交互

确定干系人代表：干系人是指积极参与项目的某个人、群体或组织，可能会受项目过程和结果的影响或影响项目的过程和结果：开发组织以外，开发组织，项目组。

确定决策者：在软件项目中，需要在关键路径上做决定，或是解决一些冲突，接受（或拒绝）某个需求变更或者批准一组即将发布的需求。决策人或决策小组对决定工作方向、保证项目进度具有重要作用。//决策规则：1.决策领导做决定2.小组投票，少数服从多数3. 小组投票，但结果必需获得一致通过4.小组讨论和协商达成共识。每个人都拥有这个决定并承诺支持5.决策领导授权一个决策人6.小组达成决策，但允许有人有权否定小组决定

需求达成共识：对需求或其中某部分达成一致是客户-开发关系的核心，1.客户确认：需求描述了他们的需要2.开发人员确认：理解需求并且可以实现3.测试人员确认：需求是可验证的4.管理层确认：需求满足业务目标//需求基线，一个特定时间点的需求快照，一组需求，作为后续开发的基础。

从需求到实现

需求获取和理解困难：确定系统边界困难：1.做什么不做什么2.和其他系统的接口如何界定；确认系统功能困难：1.用户要求不明确2.不同用户的需求不一致3.表达不清晰，存在二义性；灵活响应变化困难：1.用户、用户要求、技术发展、市场等变化带来需求的不断变化

需求记录和表达困难记录描述需求之间的关联关系1.由“粗” 到“细”功能分解，颗粒度问题2.依赖关系：输入/输出数据、执行顺序；准确描述功能的各种非功能约束，描述功能的各种异常处理需求；需求表达“语言”：1.“共同语言”-- 用户与开发人员之间沟通、理解、共识2.不同层次、不同类型的需求需要不同的“语言”

需求建模

泳道图—跨功能：识别出谁要用这个系统，不同用户之间什么关系每，类用户在一个泳道上。纵向描述的是业务过程。 横向表示他们之间的交互，纵向表示顺序。 泳道图与实现无关，描述业务过程中设计的步骤或软件系统的操作；子模块分解为可视化的泳道，表示在流程中执行操作的不同系统或执行者，绘制流程：1.分割流程的泳道2.流程（操作步骤，角色之间的协同，事务）3.流程步骤之间的事务4.条件判定

用例图—功能分解

UML:UML是一种用于指定、可视化、构造、记录软件系统的工件的图形化语言，**用来可视化(visualize) 、描述(specify)、构造(construct)和文档化(document)软件密集型系统的各种产品，** **支持不同人员之间的交流(Communication);**Unified(统一)：1.提取不同方法中最好建模技术，如OMT(James Rumbaugh)，Booch method(Grady Booch )和OOSE(Ivar Jacobson)2.采用统一、标准化的表示方式;Modeling(建模)：1.对现实系统和软件系统进行**可视化建模**2.建立系统模型;Language(语言)：1.图形化语言：语法、语义和语用2.包括规则，约束 扩展机制

成分：1.用例 ：系统与某个外部角色之间的一系列交互//2.角色：与系统交互执行某个用例的人（或其他软件系统、硬件设备）//3.用例图：对用户需求的一种概要性可视化表达。

定义：对用户需求的一种概要性可视化表达，用例不应该深入设计细节，只要描述用户设想的交互活动就够了。

绘制流程：1.确定角色，分析角色和系统交互活动2.分析业务过程，将场景概括为用例3.识别系统外部事件，并关联到角色和用例

规则：演员：1.所有与系统交互的角色都由演员表示2.每个演员与至少一个用例关联3.合并演员如果一个演员包括另一个4.分解演员如果他们和不同的用例联系；用例: 1.系统的所有主要功能都用用例表示2.用例之间的共性分析来识别新的用例以及用例之间的扩展、泛化和包含关系3.分解复杂用例

误区：1.用例用户不明白：**“用户的视角”**将用例关联到业务过程或目标2.用例过于复杂：**“用例的分解”**如果一个流程超过10到15步，就要考虑分解为多个场景3.用例内含设计：**聚焦用户与系统之间的交互**，用例 <不等于> 用户界面，用例 <不等于> 系统结构4.用例内涵数据定义：根据输入/输出关系区分功能，但用例说明中不应含有数据定义。5.用例过多：**“用例的组合”**，不要强行将每一个需求都“硬塞”到一个用例。业务需求数目 < 用例数目< 功能点数目

用例场景说明补充详细信息，记录用例的基本信息，是图的**必要的**补充说明。基本信息包括：1.一个唯一的ID和一个简洁的名称（指明用户目标）2.一个简短的文字说明，用来描述用例的意图3.开始执行用例的触发条件4.用例开始需要满足的零个或多个前提条件5.一个或多个后置条件，描述用例成功完成后的系统的状态6.一个有编号的步骤序列，展示了角色与系统之间的交互顺序

用例条件：1.触发条件：事件，引发相关用例的执行，例如“户主申请在账户中取出现金”2.前置条件定义系统开始执行用例之前必须满足的先决条件（描述系统的状态，例如：ATM系统的用例“提取现金”，必需满足前置条件“ATM机中有现金”；系统在用例执行前，需检测满足所有的前置条件。）3.后置条件描述用例执行成功后系统的状态（用户可观察到的内容（例如：系统显示帐户余额），物理产出（例如：自动取款机吐钱并打印收据），内容系统状态变化（例如：账户被扣除取现的金额与交易费用）

用例流程：1.正常流程（用例事件正常执行的步骤序列，缺省的行为；场景，又称主要流程、基本流程、快乐路径等）2.可选流程（用例内一些步骤可变，出现分支，但同样成功执行；二级场景）3.异常流程（不能成功执行的场景，描述错误条件及其处理方法）

顺序图—用户故事—用户交互：定义在一个用例中为单个场景导致某些所需结果的事件序列。（对象之间发送的消息，消息发生的顺序）；两个维度（垂直，自上而下，消息的时间序列调用发生；水平，从左到右，这些消息被发送到的对象）；实例设计图实现特定——数据库对象或特定的GUI组件作为对象

判定树判定表—条件相应：条件组合：软件系统经常受制约复杂的逻辑，有很多不同的条件组合，导致系统产生不同的行为。功能需求是在所有可能条件组合下对系统响应动作的描述，忽略条件组合将导致需求遗漏。；系统的逻辑和判定条件变得复杂之后，判定表和判定树有助于辅助分析和描述“条件组合—系统响应行为”之间的对应关系。列出影响系统行为的所有因素的各个取值，并且表明在每一个因素组合条件下系统预期的响应动作。

数据流图—数据

需求规约

记录需求：Well-formed(定义良好)需求：可验证（可验证性）：可验证；必要（必要性）：必须由系统达到或拥有以解决利益相关者问题或实现利益相关者目标；可量化（可量化性）：符合可测量条件并受约束

约束限制：1.约束条件限制了系统工程过程的设计解决方案或实现。2.可以适用于所有的需求3.可以与特定需求被指定关系，或者是和一组需求指定关系4.可以被确定为独立的需求

需求改写：增加限制，条件，和具体数值

需求缺陷：需求是昂贵的错误的主要来源。根据应用程序和环境的不同，范围从百分比到50%不等。这些漏洞最伤人的是它们是最早入侵系统的，最后离开的。一个错误的要求通过所有的开发测试、测试和初始现场使用，却在安装了数百个站点后被捕获，这并不罕见。

戴维斯假设：Davis关于需求规范的重要性的假设（戴维斯假设）：故障在生命周期中被发现的时间越晚，修复故障的成本就越高（需求分析：0.1~0.2；设计：0.5；编码：1；单元测试：2；系统测试：5；软件维护：20）

需求特性：必要，实现无关，一致，可追溯，可行，唯一；明确：正确，准确，无二义性,需求中应避免使用的歧义性词语（1.尽可能、最好的、最多的：明确量值2.不少于、不超、过、在…之间：明确边界值3.有效地、迅速的、灵活的：明确标准4.通常、大多数情况下、理想状况下：说明异常情况5.支持…功能：说明支持的内涵6.匹配的、符合的、同样的：明确比较的标准)；可验证：可以基于客观的方法，检验需求是否被正确实现。不完整、不一致、不可实现或是有歧义的需求都是不可验证的；完整：每个需求包括了必要的信息，需求文档覆盖所有功能和约束

需求优化样例:what?how?which?decompose?how to understand?

补充概念

面向对象建模语言：概念（基于面向对象的概念和抽象，提供图形化的图符，用来表示软件系统的一种语言）目的（用于建模：绘制和描述软件系统模型（分析模型和设计模型）；支持交流：便于开发人员之间的交流、沟通和讨论）组成（语法：图形化的符号表示；语义：形式或半形式的语义；语用：如何使用语言来建立模型、提供策略和原则）

多视点建模：结构视点（Structural View）用于描述系统的构成；行为视点（Behavioral View）刻画系统的行为；部署视点（Deployment View）刻画目标软件系统的软件制品及其运行环境；用例视点（Use Case View）刻画系统的功能

需求分层与分类：业务需求：希望获得的业务收益，提出系统需求的客户的业务目标；用户需求：对用户满意度最为关键的产品特性或特征的描述；软件需求：开发人员需要实现的功能以便用户能够完成的任务，进而满足业务需求。

良好的用户故事（INVEST）Independent 独立性（故事不依赖于其它故事，可以被单独地开发、测试甚至交付。）Negotiable 可协商（故事不是详尽的需求说明书（系统应该完成的事），而是可协商的意向表达。）Valuable 有价值（聚焦与由用户规定的价值，而非功能分解结构）Estimable 可估算（故事相对而言更容易估算，可以快速确定实现相关功能的工作量。）Small 小型（故事代表有价值功能的小型增量，可以在一到两周内交付。）Testable 可测试（清晰可测试）

需求管理

