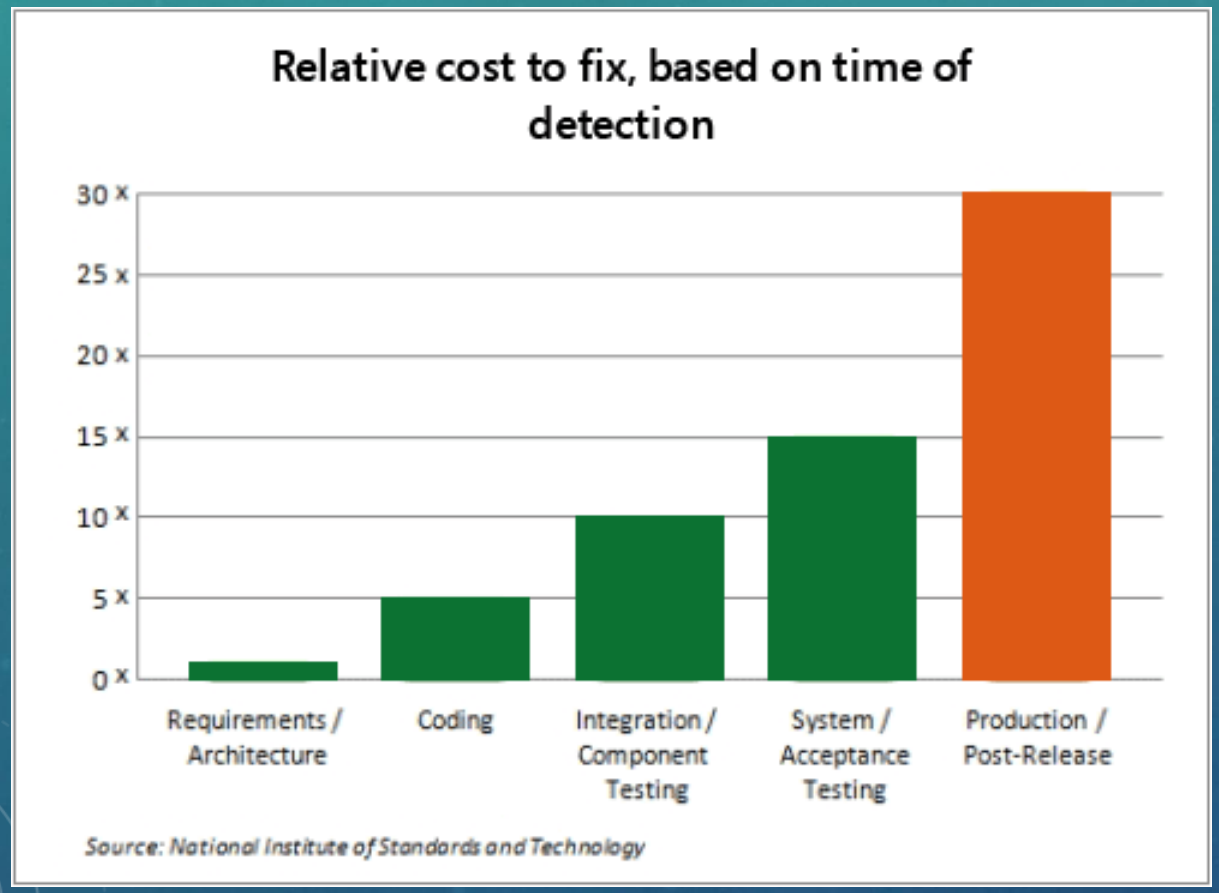
需求验证

第一部分：需求验证（第17章）

为什么要验证需求?

- 修复在软件开发生命周期需求阶段发现的缺陷平均需要 30 分钟。
- 相比之下，纠正系统测试中发现的缺陷需要 5 到 17 个小时。



为什么要验证需求?

- 需求验证活动旨在确保
  - 软件需求准确地描述了预期的系统功能、特性、外观和属性，以满足各个利益相关者的需求。
  - 要求必须是
    - 完全的，
    - 可行的，而且
    - 可验证的。

（这些是需求验证的标准。）

验证旨在确定某项开发活动的成果是否满足其要求（做事是否正确）。确认旨在评估产品是否满足客户需求（做事是否正确）。

如何进行验证?

非正式同行评审以及正式的同行评审

- 非正式同行评审
  - 同伴桌面检查，即你询问一请同事帮忙审阅你的工作成果。
  - 一个传递游戏，其中你传递需要交付给几位同事同时审阅。
  - 演示过程中，您将……带领让同行阅读交付成果，并记下同行表达的意见。

如何进行验证?

非正式同行评审和**正式同行评审**

- 正式同行评审是一个定义明确的流程，称为检查过程
  - 谁将参加?
  - 他们的角色是什么?
  - 需求文档需要达到什么级别才能开始评审（完整标准）?
  - 具体流程是什么?
  - 该流程何时可以结束（退出标准）?

如何进行验证?

- 正式同行评审的参与者:
  - SRS 的作者（BA）以及作者的同行们或许也参与其中。
  - 实际用户代表，即需要检查的需求来源。
  - 开发人员、测试人员、项目经理等将根据待检查的需求开展工作。

  - 负责与待检查需求相关的系统接口的人员，将检查外部接口需求是否存在问题。

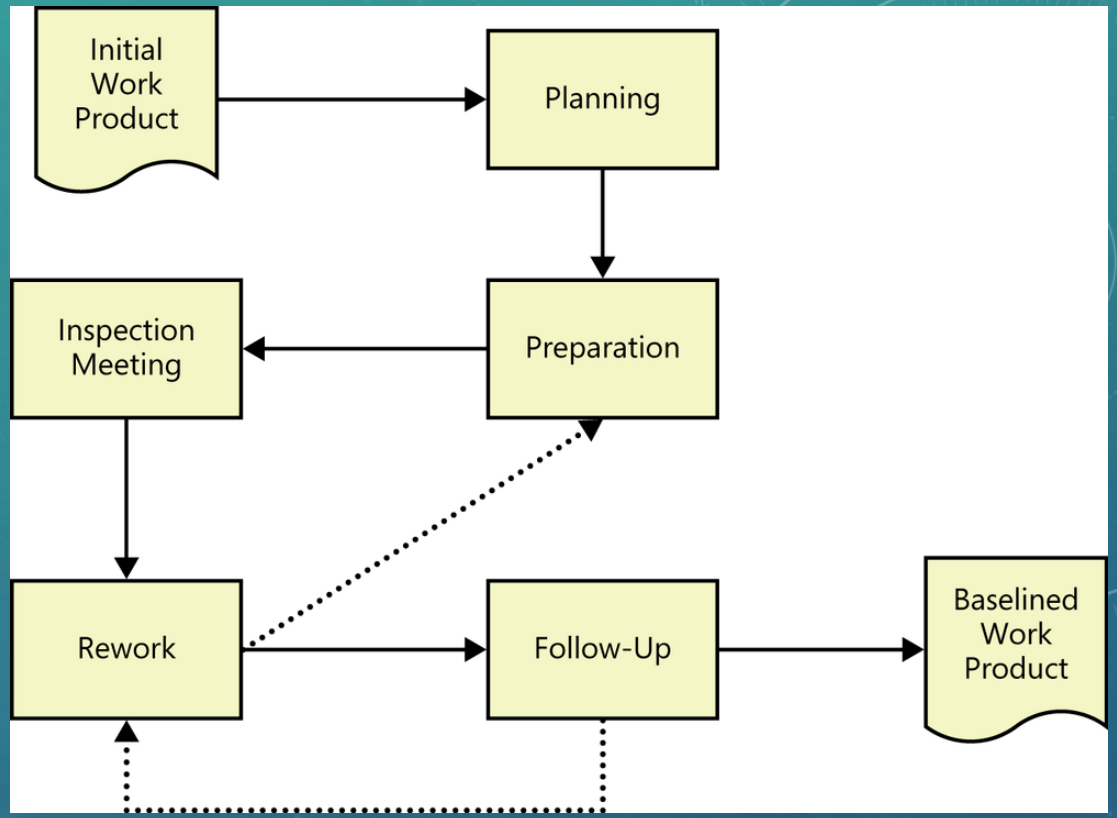
- 正式同行评审中的角色:
  - 作者——创建或维护正在接受检查的工作产品（SRS）
  - 主持人/主席——与作者共同制定检查计划，协调各项活动，并主持检查会议。

  - 读者——一名检查员被指定为读者，代表 SRS。
  - 记录员——使用标准表格记录会议期间提出的问题和发现的缺陷。

  - 其他检查员——对SRS项目提出疑问

如何进行验证?

- 正式同行评审流程
  - 规划：BA 和主持人确定谁应该参加，检查员在检查会议之前应该收到哪些材料，涵盖这些材料所需的总会议时间，以及检查应该何时安排。
  - 准备工作：在检查会议之前，BA 应与检查员分享背景信息，以便他们了解被检查项目的背景以及 BA 的检查目标。
  - 检查会议：在检查会议期间，阅读员带领其他检查员逐一阅读文件，逐项描述要求。他自己的话当检查员提出可能的缺陷和其他问题时，记录员会将它们记录在需求作者的行动项列表中。
  - 返工：业务分析师应计划在检查会议后花一些时间重新审视需求。
  - 后续跟进：在最后的检查步骤中，主持人或指定人员与业务分析师合作，确保所有未决问题都已解决，错误也已得到妥善纠正。



如何进行验证?

- 正式同行评审的准入标准——如果SRS不符合条件，则不应举行同行评审会议：
  - 该文档符合标准模板，没有明显的拼写、语法或格式问题。
  - 文档上会打印行号或其他唯一标识符，以便于查找特定位置。
  - 全部未解决的问题标记为待定（TBD）或在问题跟踪工具中可访问。
  - 主持人没有发现超过三对该文件的一个代表性样本进行十分钟检查后，发现了重大缺陷。
- 退出条件——当满足以下条件时，审查流程即可结束：
  - 检查期间提出的所有问题（包括新提出的一个问题）这会议）已讨论完毕。
  - 对需求及相关工作成果所做的任何更改均已正确执行。（这意味着业务分析师需要根据检查员在会议期间及之后提出的意见修改软件需求规格说明书。）
  - 所有未决问题均已解决，或者每个未决问题的解决流程、目标日期和负责人均已记录在案。（这意味着需要召开下一轮审查会议。）

如何进行验证?

- 需求评审是一项挑战
  - 需求文档篇幅很长——可能会有人仔细查看第一部分，一些勤奋的审阅者会研究中间部分，但不太可能有人会看最后一部分。
  - 检查团队规模庞大——会增加审查成本，难以安排会议，并且难以就问题达成一致意见。
  - 地理位置分散的审稿人（特别是，例如，由于新冠疫情导致的旅行限制）19）
  - 准备不足的评审人员——可能会导致他们在会议期间临时思考，从而错过许多重要问题。

V模型（指不在需求阶段进行的测试）

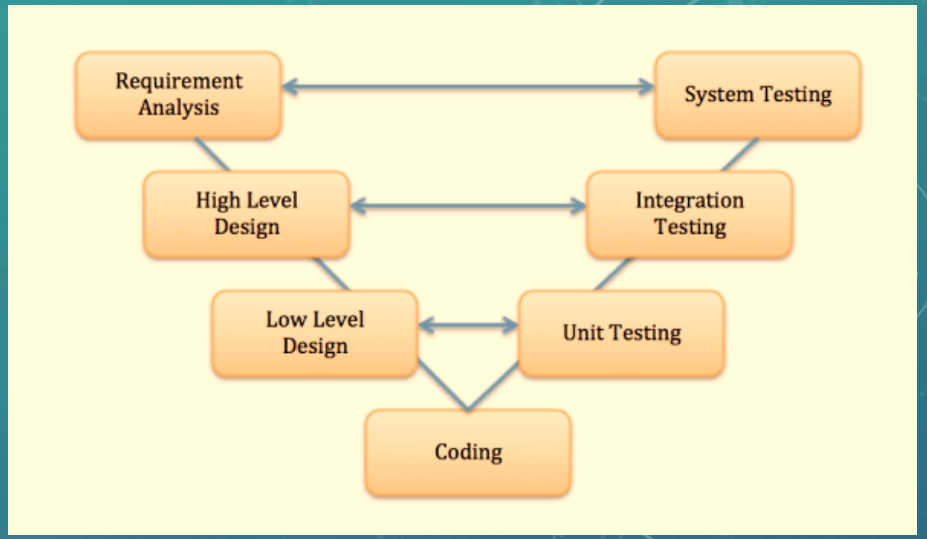
软件开发生命周期

软件开发 生命周期（SDLC）	每个阶段开展的活动
需求收集 阶段	尽可能多地从客户那里收集有关所需软件的详细信息和规格。
设计阶段	规划适合项目的编程语言（如 Java、PHP、.net）和数据库（如 Oracle、MySQL 等），以及一些高级功能和架构。
构建阶段	设计阶段之后，就是构建阶段，也就是实际编写软件代码。
测试阶段	接下来，你要测试软件，以验证它是否按照客户提供的规格构建。
部署阶段	在相应的环境中部署应用程序
维护阶段	系统准备就绪后，您可能需要根据客户要求稍后更改代码。

V型

V模型（V – 验证）的概念 测试是为软件开发生命周期（SDLC）的每个阶段而设计的。

- 模型左侧是软件开发生命周期（SDLC）。
- 模型右侧是软件测试生命周期（STLC）。
- V型模型表示
  - 验收测试源于用户需求（用于底层设计——可以更早地发现设计中的错误），
  - 系统测试基于功能需求（需求分析的新内容——确保关键细节没有遗漏），
  - 集成测试基于系统架构（针对高级设计——架构中的错误可以更早发现）。



- 单元测试是 O，因为我们总是在测试代码和测试环境时进行测试。
  编程
- 集成测试是可以的，因为例如在传统的软件工程中，我们有原型设计。
- 需求阶段的系统测试
- 如何进行测试？测试平台是什么？——参见幻灯片 4——完整、可行且可验证。

V型

测试原型设计要求

- 为同伴提供可供参考的系统模型（？？）

- 所有原型制作技术都可以使用。（我认为关键在于设计）测试场景。） **使用原型测试需求** （可行性验证）

	丢弃	进化
小样	明确并完善用户和功能需求。丢失的 功能。 探索用户界面 方法。	实现核心用户 要求。 根据优先级实施额外的用户需求。实施并优化网站。使系统适应快速变化的业务需求。
证明概念	展示技术 可行性。 评估绩效。 获取知识 改进估算 建造。	实现并扩展核心多层架构功能 通信层。 实现并优化核心算法。 测试并优化性能。

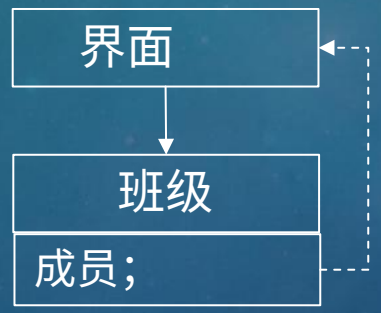
- 根据用户需求进行概念测试，涵盖每个用例的正常流程、替代流程以及您在需求收集和分析过程中确定的例外情况。
- 概念测试与实施无关。
- 验收测试——敏捷开发方法通常会创建验收测试，而不是编写精确的功能需求。每个测试都描述了用户故事在可执行软件中的运行方式。由于验收测试在很大程度上取代了详细的需求，因此敏捷项目中的验收测试应该涵盖所有成功和失败的场景。

敏捷软件开发——测试

测试驱动开发（TDD）

- 能够设计一个测试环境在实际开发/编写代码之前，展现开发人员对程序要实现的目标以及运行方式的非常清晰的思考。
- 拥有这样的测试环境的明显好处是，可以在开发过程中验证程序的所有功能。

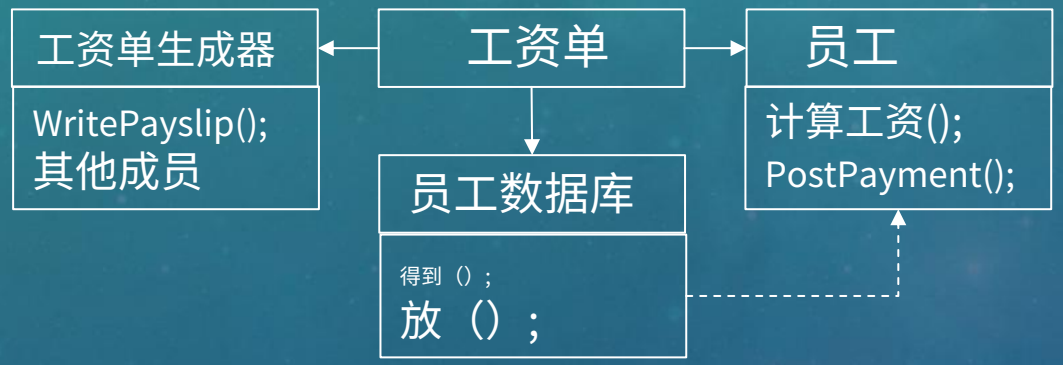
- 测试环境实际上是一个接口，允许开发人员访问待测程序，进行验证或确认。（它是程序员与函数/行为之间的接口，但不是类接口。）



敏捷软件开发——测试

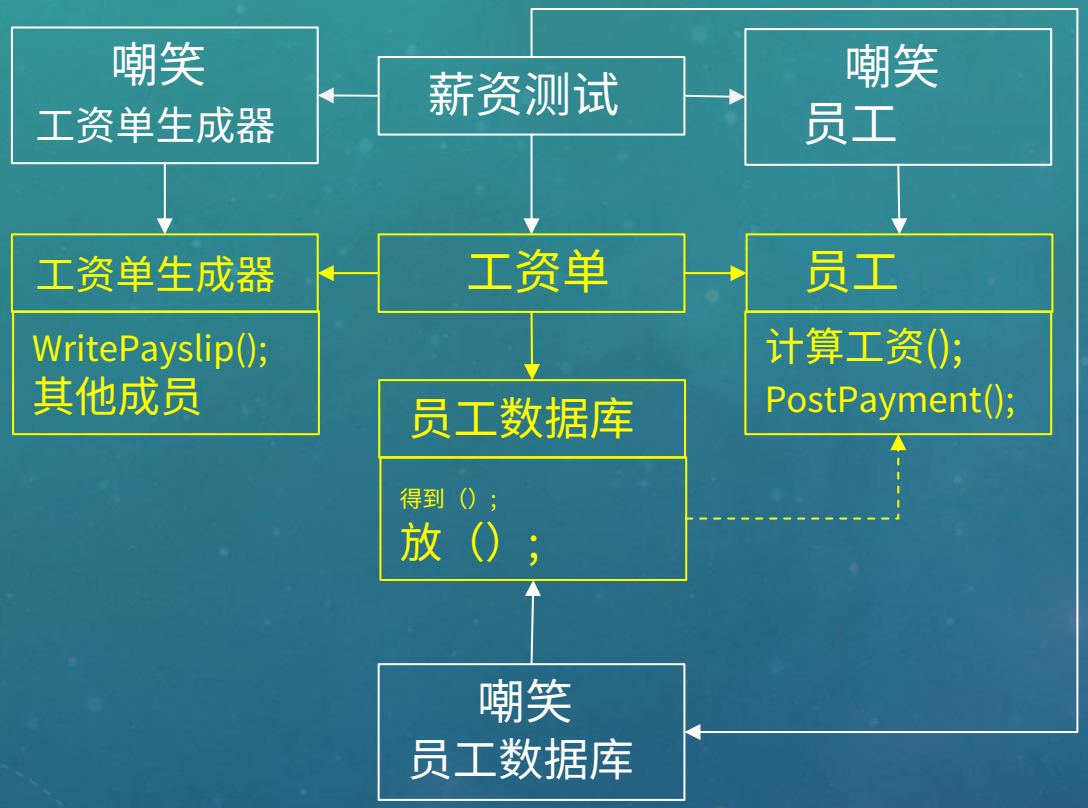
例如：假设我们要创建一个工资系统，该系统由工资类、员工类、工资单编写器类和员工数据库组成。

1. 工资系统首先调用 EmployeeDB.Get() 获取员工信息。
2. 然后调用 Employee.CalculatePay() 来计算应付金额。
3. 然后，它调用 PayslipWriter.writePayslip() 函数打印出员工的工资单。



敏捷软件开发——测试

测试环境应如下所示。



- “白线”系统是测试环境，“泛黄的线”是程序。
- MockEmployee、MockEmployeeDB 和 MockPayslipWriter 分别是 PayrollTest 与 Employee、EmployeeDB 和 PayslipWriter 对应的“真实”类之间的接口。

- 这是一个“原型”。
- 在开发 `Employee.CalculatePay()` 方法时，为了测试该方法，`PayrollTest` 会通过 `MockEmployee` 对象调用该方法，就好像 `MockEmployee` 对象本身就拥有该方法一样。然后，`MockEmployee` 对象会调用 `Employee` 类中的方法。
- 其他方法在开发过程中也会以相同的方式进行测试。

敏捷软件开发——测试

验收测试

- 任务级和单元级测试以“白盒测试”的方式验证代码。
- 验收测试以“黑盒测试”的方式验证整个US的实现，这意味着开发人员向程序提供输入，并检查输出是否是他们预期的结果。

- 理想情况下，验收测试应该在编程之前设计。
- 例如：为了测试工资管理程序是否能够添加和检索员工记录，我们可以通过 GUI（工资管理类）将以下内容输入到程序中：

  123445678，李大友，男，2000年

  然后搜索记录并将其显示在屏幕上，也是通过图形用户界面 (GUI)。

如何进行验证?

设计测试以根据验收标准验证需求。 可以通过以下方式对验收标准进行验证:

- 首先从高优先级的功能需求入手，即那些必须满足才能使产品被接受和使用的需求，或者那些可能运行不正常且必须在不延迟初始版本发布的情况下进行修复的需求。
- 然后转向必须根据内部和外部质量属性的优先级来满足的基本非功能性标准。
- 接下来是上一轮同行评审会议中遗留的未决问题和缺陷。
- 结合具体的法律、法规或合同条件。
- 继续支持过渡、基础设施或其他项目（而非产品）需求。

第二部分：需求重用（第18章）

你疯了吗？这可能吗？

有可能参见此示例

- 英国航空公司的网站允许乘客办理登机手续、选择座位。
  用户可以通过他们的应用程序支付升舱费用并打印登机牌。机场的自助值机亭也具备相同的功能。此外，他们的应用程序还支持在线值机、选座和支付升舱费用。
- 法国航空也有自己的网站、应用程序和机场自助服务终端，功能相同。
- 你会发现需求是相同的，因此它们可以在不同的产品和不同的客户之间重复使用。

好处

- 从开发者的角度来看——更快的交付速度、更低的开发成本、应用程序内部和应用程序之间的一致性、更高的团队生产力、更少的缺陷和更少的返工。
- 从用户角度来看——产品线相关成员或一组业务应用程序之间的功能一致性（为什么这很重要——想想同一家航空公司的网站、应用程序和自助服务终端）。

三维空间——需求维度与重用

- 重用程度（X轴，离散），值为
  - 个人需求说明（例如，乘客想要办理登机手续……）
  - 要求及其属性（……座位已确认，登机牌二维码已发送至手机，……）
  - 需求及其属性、上下文和相关信息，例如数据定义、术语表定义、验收测试、假设、约束和业务规则（……允许乘客在机场扫描通行证。）
  - 一组相关要求
  - 一套需求及其相关的设计元素（例如类图、交互图等）
  - 一套需求及其相关的设计、代码和测试元素（例如测试环境）

三维空间——需求维度与重用

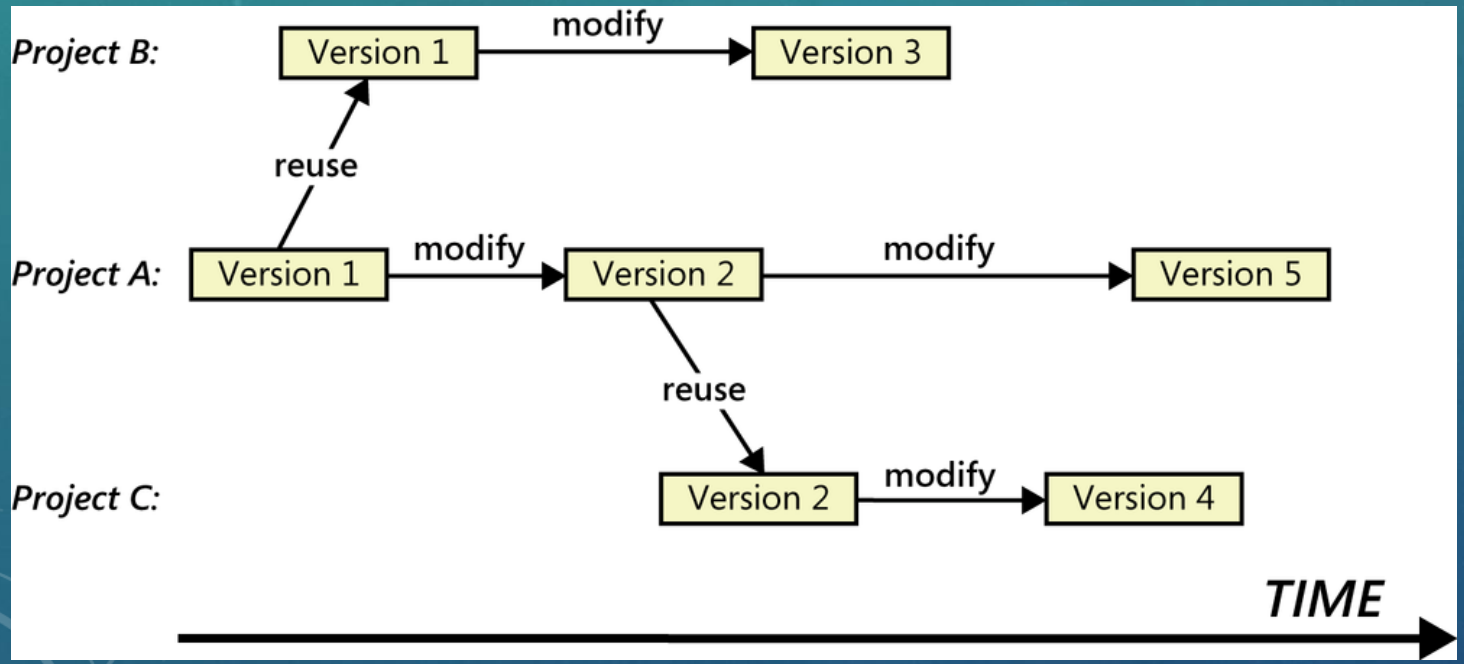
- 修改程度（Y轴，离散值），数值为
  - 例如，飞机检测和机器人自激励学习中不需要VSLAM。
  - 相关需求属性（优先级、理由、来源等），例如，在飞机检查中，“高质量图像”的优先级高于“位置信息”，以及在机器人学习中类似的情况。

  - 需求说明本身（定量而非定性）
  - 相关信息（测试、设计限制、数据定义等），例如，室内巡检中无人机飞行路径的限制与室外巡检中的限制不同。

- 重用机制（Z轴），数值为
  - 从其他规范或其他项目中复制粘贴
  - 从可重用需求库中复制
  - 参考原始资料

良好做法

数据库中存储着不同版本的需求。如果有人修改了数据库中的需求，你正在重用的旧版本仍然存在。这样，你就可以根据项目需求定制自己的需求版本，而不会影响其他重用者。（类重写的概念？为什么不先提出一个高层次的（抽象或接口）需求，然后再针对各个应用程序进行具体说明呢？——请参考“需求模式”。）



可能的再利用机会

某些类型的需求相关资产具有良好的重用潜力

重用范围	潜在可重用需求资产
在产品或应用程序中	用户需求、用例中的具体功能需求、性能需求、可用性需求、业务规则
整个产品线	业务目标、业务规则、业务流程模型、上下文图、生态系统图、用户需求、核心产品功能、利益相关者概况、用户类别描述、用户角色、可用性需求、安全需求、合规性需求、认证需求、数据模型和定义、验收测试、术语表
整个企业	业务规则、利益相关者概况、用户类别描述、用户角色、术语表、安全要求
整个企业领域	业务流程模型、产品功能、用户需求、用户类别描述、用户画像、验收测试、术语表、数据模型和定义、业务规则、安全需求、合规性需求
在运营中环境或平台	支持特定类型需求（例如报表生成器）所需的约束、接口和功能基础设施

可能的再利用机会

示例

- 在同一产品中——在 Luton DPS CRM 项目中，信息台、顾问和财务这三个不同的用户都需要访问 CRM 系统，尽管目的各不相同。

- 从产品线来看，京东的网页和应用程序共享相同的业务目标和规则，以及许多用户需求。

- 在企业内部，例如英国航空公司、中国国际航空公司等，在其自助值机终端系统中共享业务规则、利益相关者概况、用户类别描述以及许多其他信息。
- 纵观整个商业领域——亚马逊和国民西敏寺银行或许有着相同的安全要求。
- 在操作系统环境或平台内，Linux 的 GUI 也像 Windows 的 GUI 一样发生了变化。

锻炼：假设你是京东App开发项目的业务分析师。你能列出5个可以直接从京东网站应用继承的功能需求，以及至少2个你认为需要修改的功能需求吗?

可能的再利用机会

以下是一些与重用相关的其他需求信息组。

- 功能及其相关异常和验收测试
- 数据对象及其相关属性和验证
- 与合规相关的业务规则、行业其他监管限制以及组织政策导向的指令

- 对称的用户功能，例如撤销/重做（如果您重用了应用程序撤销功能的需求，也请重用相应的重做功能的需求）
- 数据对象及其相关操作，例如创建、读取、更新和删除

可能的再利用机会

再利用机会

重复使用机会	例子
业务流程	通常，业务流程在不同组织中是通用的，需要统一的软件支持。许多机构维护着一套业务流程描述，以便在不同的IT项目中重复使用。
分布式部署	同一系统经常会被多次部署，每次部署之间略有不同。这在零售商店和仓库中相当普遍。每次部署都会重复使用一套通用的需求。
接口和一体化	为了实现接口和集成，通常需要重用现有需求。例如，在医院中，大多数辅助系统都需要与入院、出院和转院系统建立接口。这同样适用于与企业资源计划系统 (ERP) 的财务接口。
安全	用户身份验证和安全要求在不同系统中通常相同。例如，这些系统可能有一个共同的要求，即所有产品都必须使用 Active Directory 进行用户身份验证，从而实现单点登录。
常见的特征	商业应用通常包含一些通用功能，这些功能的需求甚至完整的实现都可以复用。例如，搜索操作、打印、文件操作、用户配置文件、撤销/重做以及文本格式化等。
类似产品多平台	尽管不同平台在具体需求和/或用户界面设计上可能存在一些差异，但核心需求集是相同的。例如，同时运行在 Mac 和 Windows 系统或 iOS 和 Android 系统上的应用程序。
标准，法规、法律遵守	许多组织制定了一套标准，这些标准通常基于法规，并以一系列要求的形式呈现。这些标准可以在不同的项目中重复使用。例如，ADA无障碍设计标准和HIPAA医疗保健公司隐私规则。

标准化：需求模式

右侧的图表是一个模板，展示了项目可能遇到的每种常见需求类型的信息类别。不同类型的需求模式将有各自的内容类别。

需求模式包含以下几个部分：

- 指导——相关模式、适用（和不适用）的情况，以及如何编写此类需求的讨论。
- 内容——对该要求应传达的内容进行逐条解释。
- 模板——需求定义，其中包含用于放置可变信息的占位符。
- 示例——此类要求的一个或多个示例。
- 额外要求——可以定义主题某些方面的附加要求，或者解释如何编写一组详细要求。
- 开发和测试注意事项——开发人员在实现模式所指定的类型需求时需要考虑的因素，以及测试人员在测试此类需求时需要考虑的因素。

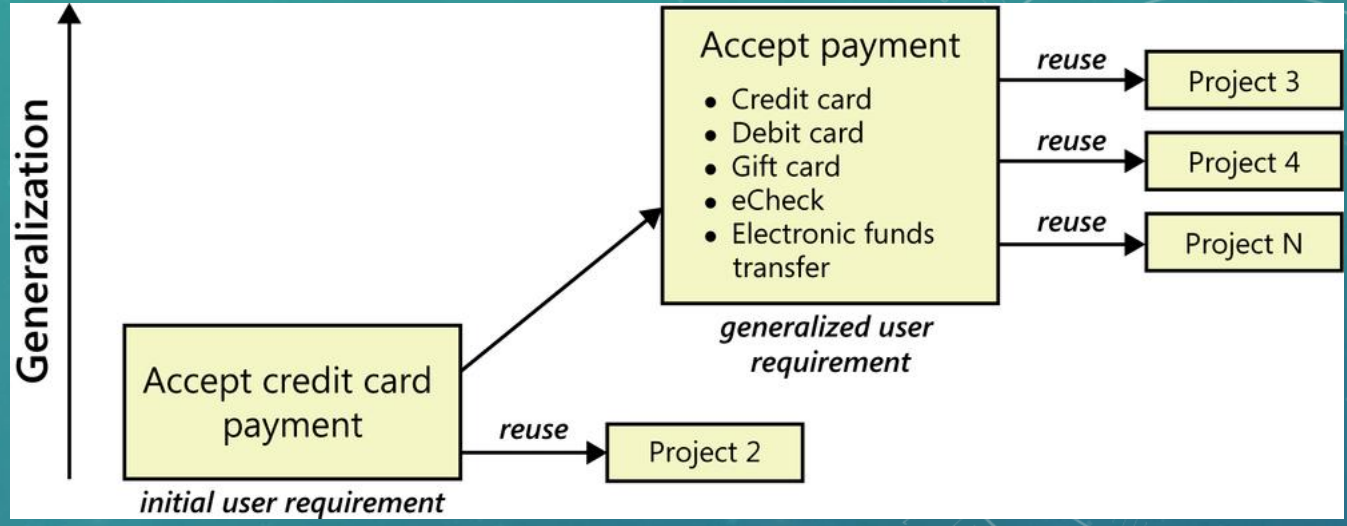
	Software Requirements Pattern Template
Pattern Name	<descriptive name>
Pattern Id	<unique identification number>
Pattern Description	The system shall store the information corresponding to <relevant concept>
Author	<author name>
Source	< Source of requirement >
Classification	{Functional/ NonFunctional Requirement}
Classification-Purpose facets	<Type of Functional or Non functional requirement>
Goal	<Gives the purpose of the requirement pattern>
Applicability	< Description of area in which the pattern may be applied.>
Constraints	<Relevant Constraints if any>
Content data-Specific	<Specific data about the relevant concept> ….

制作要求

可重复使用的

关键在于合适的抽象层次

- 可重用需求——更高层次的抽象

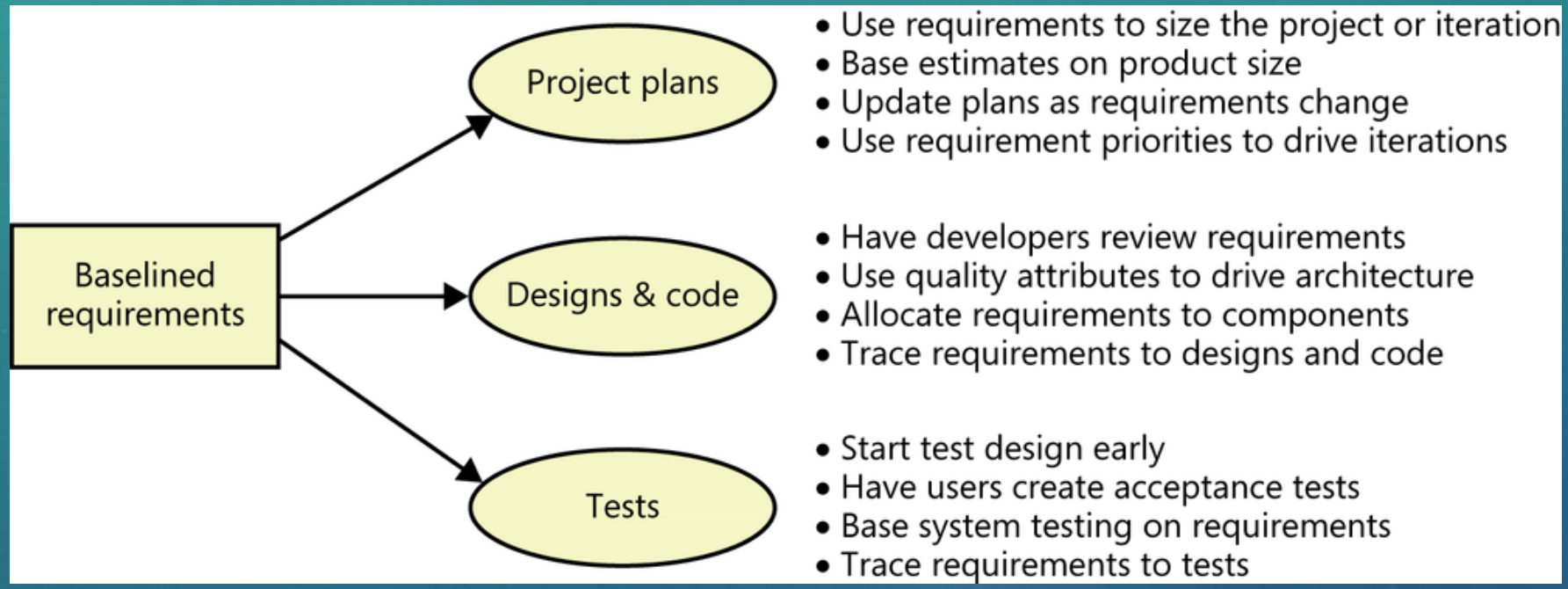


- 领域特定需求——低级
- 例如（见图）：
  - 该应用程序包含一项用户需求，即接受信用卡付款。这项用户需求将扩展为一系列与处理信用卡付款相关的功能性和非功能性需求。
  - 其他应用程序可能也需要接受信用卡付款，因此这是一套潜在的可复用需求。
  - 将用户需求概括为涵盖多种支付方式：信用卡、借记卡、礼品卡和电子货币转账。
  - 由此产生的需求使其在未来的更多项目中具有更大的复用潜力。例如，有的项目可能只需要信用卡处理，而有的项目则需要多种支付处理方式。
  - 将初始用户需求概括为“接受信用卡付款”这样的概括，即使在当前项目中也可能很有价值。

第三部分：继续
要求（第19章）

接下来该怎么做?

将软件需求转化为合理的项目计划、稳健的设计和测试（三个方向）



估算需求工作量

- 其目的是确定在软件开发项目中，需求分析工作量应占总工作量的百分比。（人力资源规划是项目规划的重要组成部分。此估算将说明所需的业务分析师人数及其工时。）

- 一些数据：
  - 一项针对15个电信和银行项目的调查显示，最成功的项目将28%的资源投入到需求分析中。平均而言，每个项目将15.7%的精力用于需求工程。
  - NASA那些将超过10%的总资源投入到需求开发中的项目，其成本和进度超支情况远小于那些在需求开发方面投入较少资源的项目。
  - 一项欧洲研究表明，产品开发速度更快的团队比开发速度较慢的团队投入了更多的时间和精力来满足需求。

	为满足要求所付出的努力
更快的项目	14%
进度较慢的项目	7%

估算需求工作量

- 方法——三种估算方法（由咨询公司Seilevel提出）
  - 首先——占总工作量的百分比，15%是一个正常的考虑因素。
  - 第二点——开发人员与业务分析师的比例，默认值为 6:1，但有些项目使用 3:1。
  - 第三——活动分解，即业务分析师在用户故事、报告等各种活动上花费的时间。

  - 需求工作量 =*f*（第一、第二、第三）。（函数 f 是什么？作者没有告诉我们。他们提供了一个电子表格工具，您或许可以访问。）

从需求到项目计划

估算项目工作量

- 尺寸取决于：
  - 可单独测试的需求数量
  - 函数的数量
  - 用户故事或用例的数量
  - 用户界面元素的数量、类型和复杂程度
  - 实现特定需求所需的估计代码行数

从需求到项目计划

项目进度安排

- 有效的项目进度安排需要以下要素：
  - 预计产品尺寸（如上一张幻灯片所示）
  - 团队规模（业务分析师、开发人员等）
  - 根据历史表现，开发团队的已知生产力
  - 为完整实现某个功能或用例，需要列出一系列任务。
  - 至少对于即将到来的开发迭代而言，需求应该相当稳定。
  - 经验可以帮助项目经理调整应对无形因素和每个项目的独特之处。



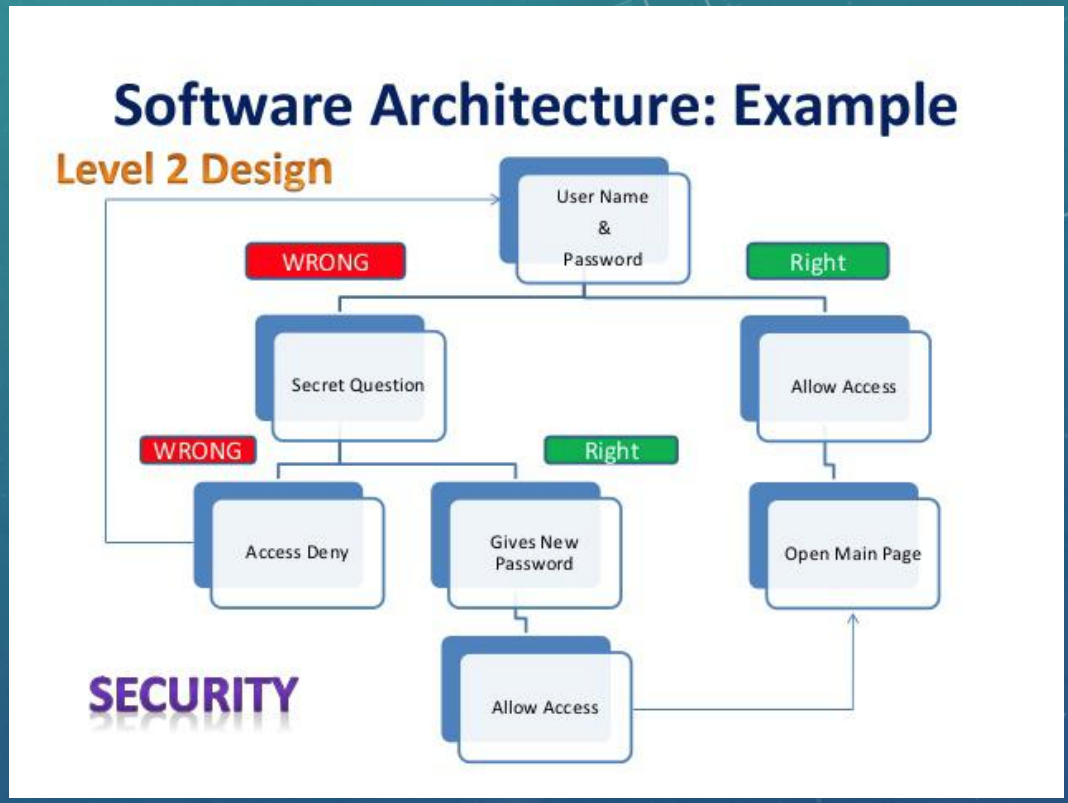
从需求到设计和代码

建筑设计

- 它是识别构成子系统控制框架的各个子系统的过程，并且
  沟通。此设计过程的输出结果是对软件架构的描述。
  架构设计是系统设计过程的早期阶段。

- 它将高层系统需求分配给各个子系统和组件。

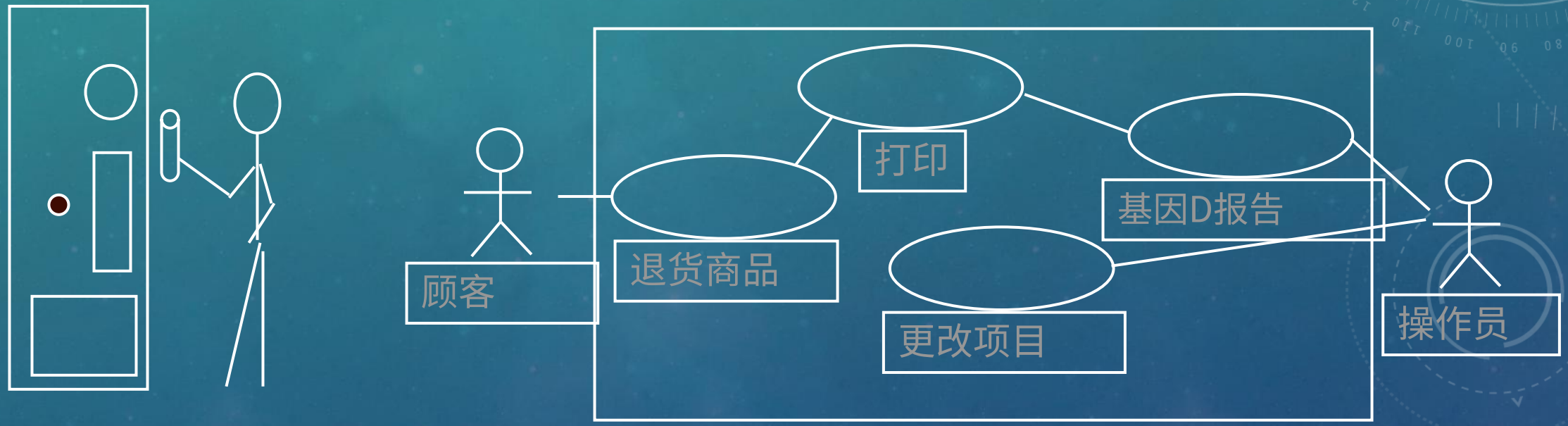
- 它允许开发团队跟踪设计中每个需求的实现情况。
- 系统需求驱动架构设计，而架构又影响需求分配。



从需求到设计和代码

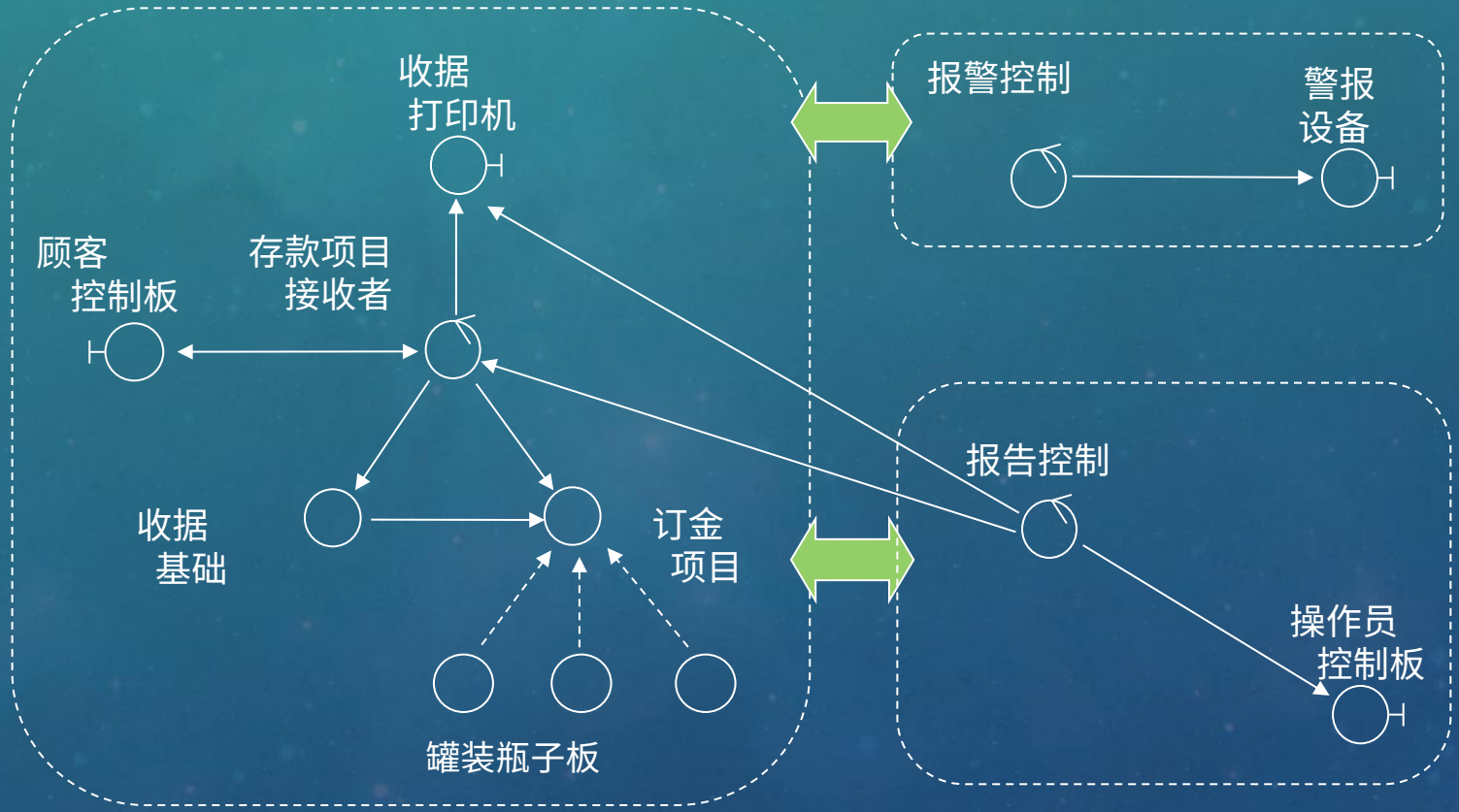
软件设计

- 面向对象的需求分析会生成类图和其他 UML 模型。设计人员可以将这些概念性的类图细化成更详细的对象模型，用于设计和实现。



从需求到设计和代码

工作人员必须查看他会指出每个项目返回的类型。登记每位顾客退回多少件商品，以及当顾客索要收据时，他会怎么做？打印计算出已存入的金额、退回物品的价值以及退款总额。有薪酬的他会向顾客说明情况。他还会每天结束时打印出已存入物品的总数。他有权……改变通过控制台输入物品的存款金额。机器出现任何故障时，都会发出特殊的警报信号通知他。



从需求到测试

- 需求是系统测试和用户验收测试的最终参考，也就是说，“应该测试产品是否能够实现其预期功能，而不是测试其设计和代码”——这当然意味着验证而不是确认。

- 要测试什么?
  - 正常情况下的预期行为
  - 异常情况下的预期行为
  - 质量属性
- 如何测试?
  - 黑匣子
  - 那么白盒测试呢？——它是用来测试设计和代码的！不过，它也有助于发现黑盒测试的失败之处。
  - 白盒测试基于设计中的逻辑，例如控制和逻辑推理。