第7章 架构驱动的软件开发

第7章 架构驱动的软件开发

- 7.1 简介
- 7.2 架构需求获取
- 7.3 架构设计、文档化和评估
- 7.4 架构的实现与维护

7.1架构驱动的软件开发简介

- 架构驱动的软件开发主要包括以下几个步骤
  - 1）架构需求获取；
  - 2）基本架构设计；
  - 3）架构记录文档化；
  - 4）架构评估；
  - 5）架构实现；
  - 6）架构维护。

7.1架构驱动的软件开发简介

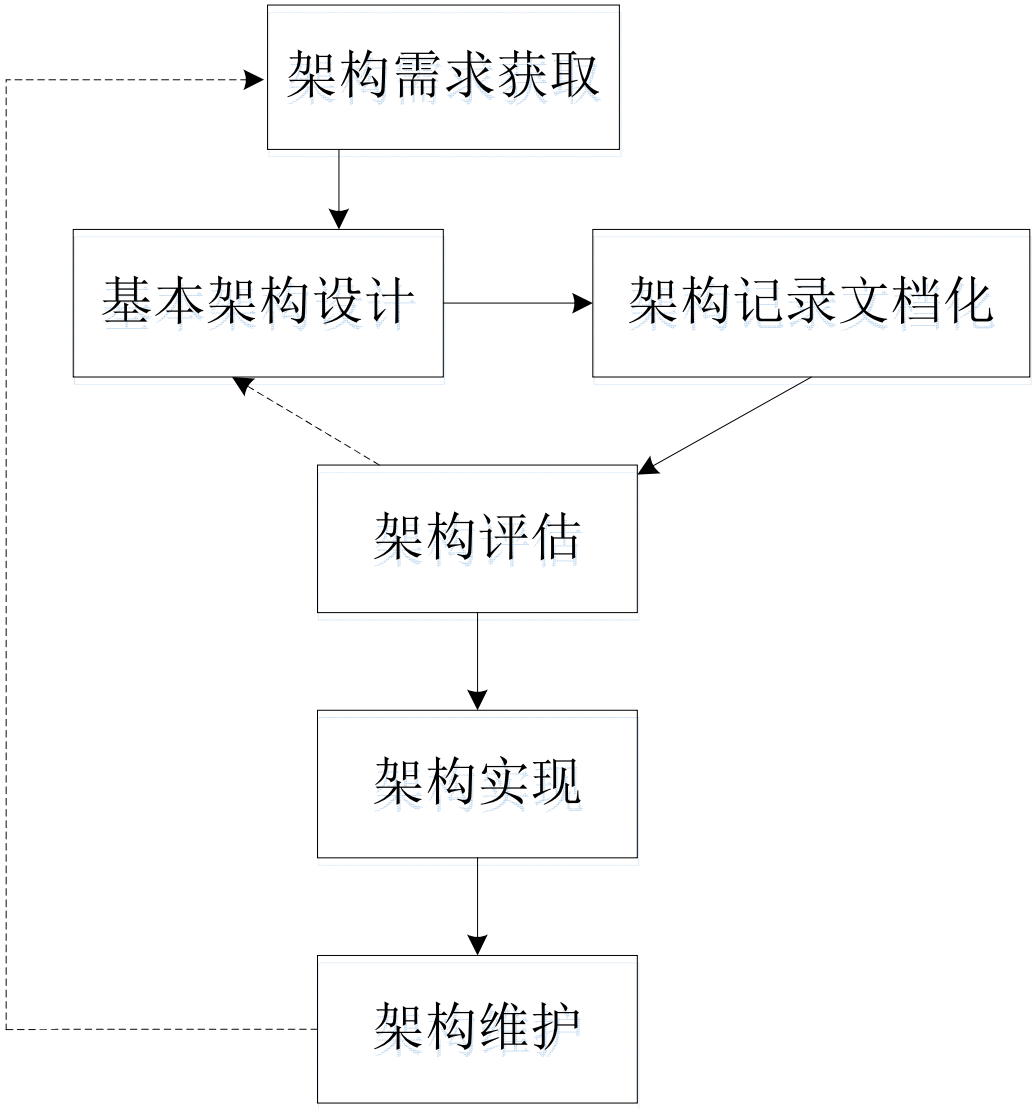


图7.1 架构驱动的软件开发流程图

7.2 架构需求获取

- 架构相关需求的获取是从软件的用户群和软件的使用环境中提取的
- 架构的需求源于：系统的质量目标、系统的业务目标、软件的利益相关者
  - 架构的需求与功能的需求是不同的。在设计架构的时候只需要讨论架构层次的需求，没有必要再深入到功能性需求。
- 如何抽取恰当的架构相关的需求信息，与参与设计的架构师的经验密不可分。

7.2 架构需求获取

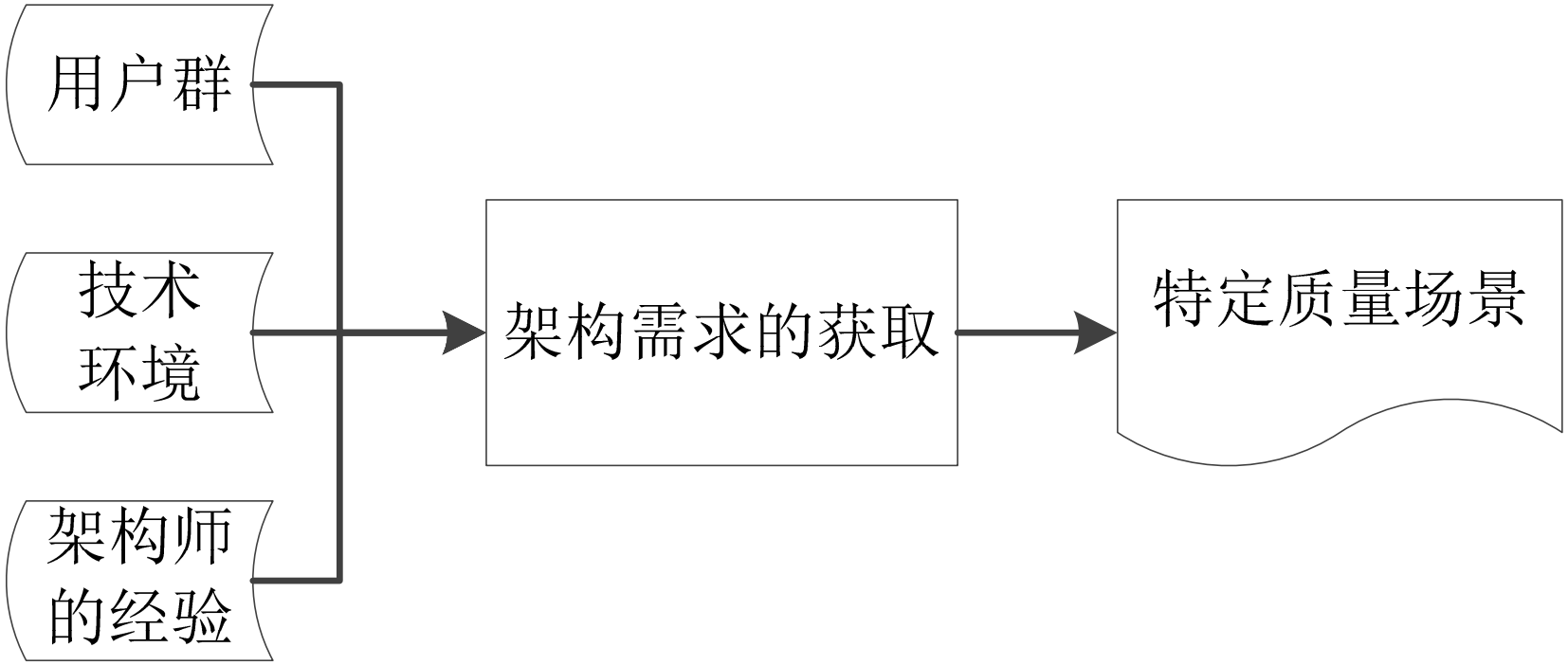


图7.2 架构需求的获取阶段任务描述图

7. 2. 1通过质量模型描述架构需求

- 通常架构的需求是十分抽象的，没有什么编程语言会提供实现架构需求的机制。
- 因此，进行架构需求描述的步骤通常为：
  - 首先要将架构的需求通过一定的场景进行描述
  - 接着通过一定的模型描述这些场景，获得架构需求的结构化描述。

7.2.1通过质量模型描述架构需求

- 1. 质量场景
  - 架构需求要用质量场景进行描述。
    - 抽象场景：根据软件的使用进行一定层面的分类（如：软件流水线方式、三层结构等），这些分类就会对相应软件提出一定的需求，此类需求即为架构需求的抽象场景。
    - 例如：由Yinzer系统架构需求的质量场景描述（见下页），可以看出该描述可用于所有用到浏览器和客户端的软件系统的架构分析中。

7. 2. 1通过质量模型描述架构需求

表7.1 对Yinzer系统架构需求的一个完整的质量场景的描述

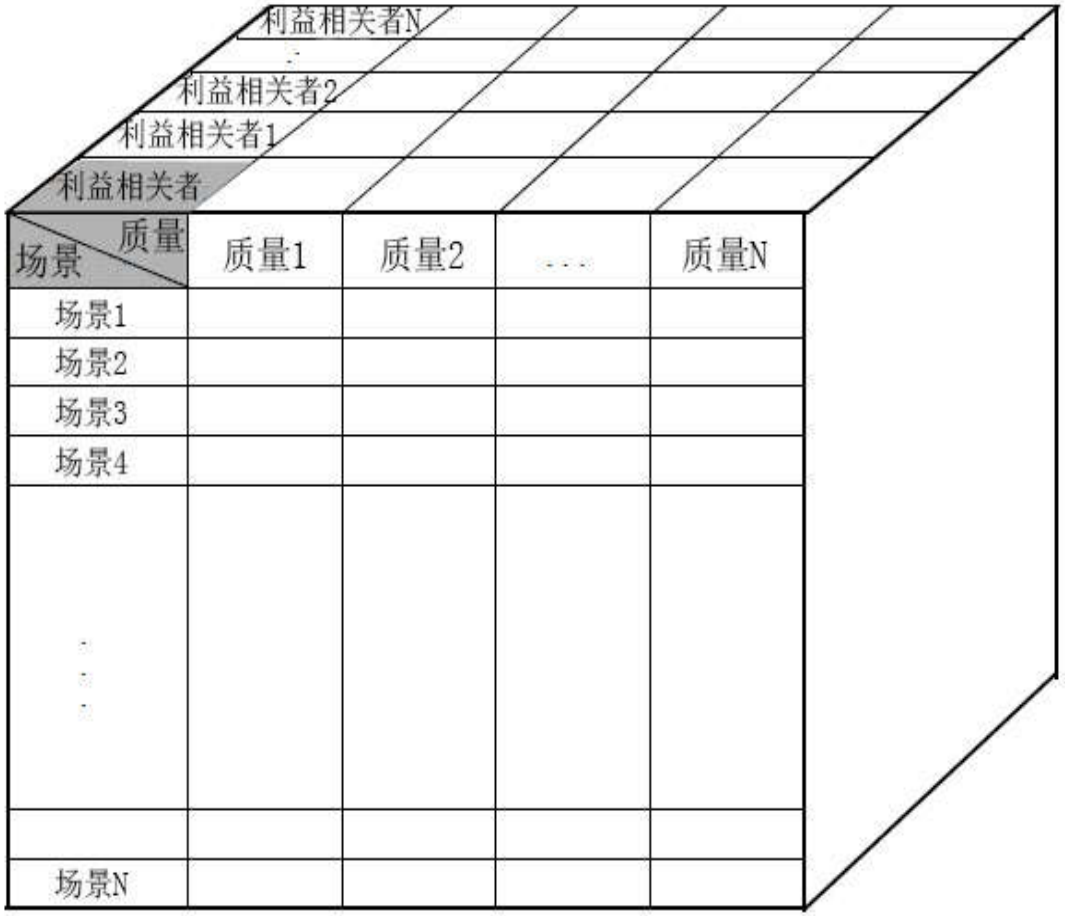
源	Yinzer用户
触发	请求Yinzer服务器上的网页
环境	正常操作
制品	整个系统
响应	服务器返回网页
响应测试	Yinzer系统在1s内返回页面
完整地质量场景	Yinzer用户点击浏览器中的链接；浏览器向Yinzer系统发送请求，Yinzer系统在1s内返回页面。

7.2.1通过质量模型描述架构需求

- 1. 质量场景
  - 架构需求要用质量场景进行描述。
    - 对于架构师和领域专家来说，需要做的是从抽象场景描述中获得特定的质量属性场景。
    - 例如：“Yinzer用户点击浏览器中的链接；浏览器向Yinzer系统发送请求，Yinzer系统在1s内返回页面” 就对系统性能提出了要求。
    - 通常来说，我们考虑的特定的质量场景是对性能、可移植性、可替换性、可重用性等质量属性产生影响时的质量场景。

7.2.1通过质量模型描述架构需求

- 提取质量场景模型示意图



7.2.1通过质量模型描述架构需求

- 2. 软件质量模型
  - 软件质量理想模型：对质量场景进行描述，可以用来描述，评估和预测质量属性的模型。

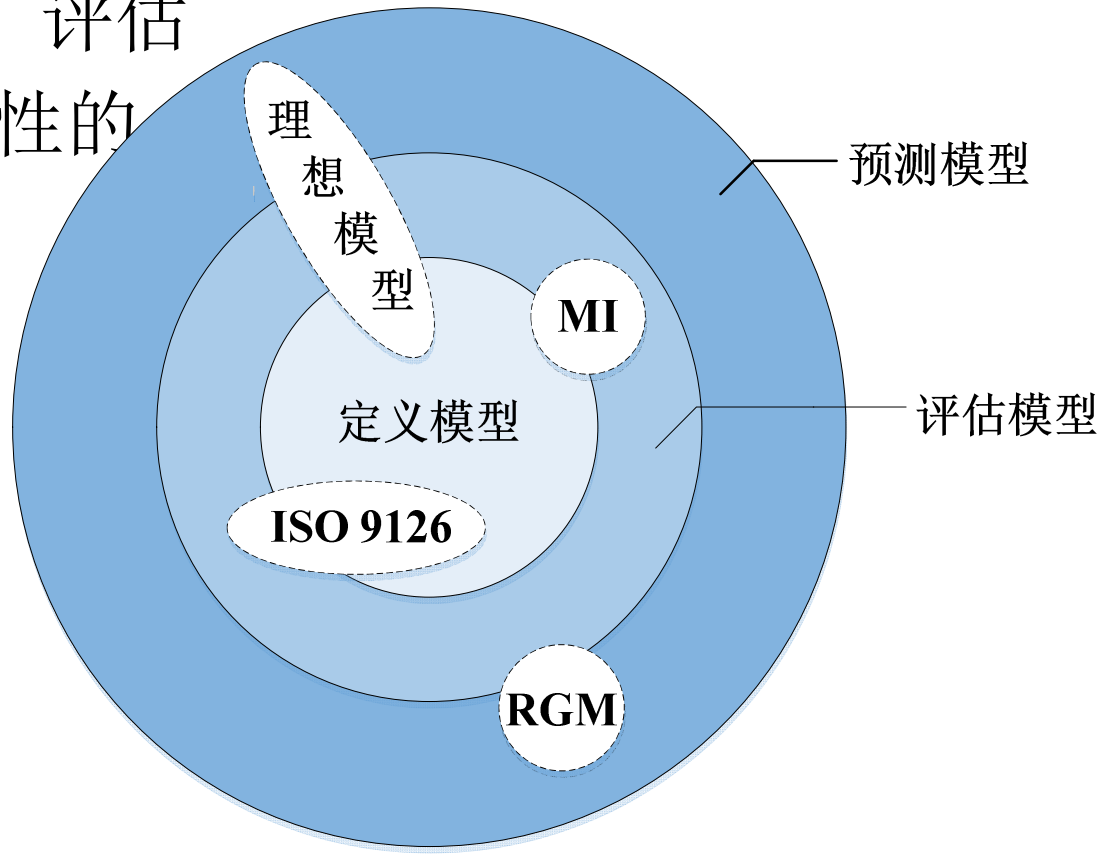


图7.4 软件质量理想模型结构图

7.2.1通过质量模型描述架构需求

- 2. 软件质量模型
  - 为了使得软件的质量模型能足够的清晰和结构化，就需要有明确的质量元模型的定义
  - 软件质量元模型（软件质量理想模型设计原则）：可以清晰地描述质量模型中元素和元素之间相互关系的模型。
  - 软件质量元模型包括了实例化的软件质量理想模型和对质量场景的注解。

7.2.1通过质量模型描述架构需求

- 2. 软件质量模型
  - 软件质量理想模型框架：描述了如何将软件质量元模型进行实例化，并且可以对质量属性进行定义、评估、预测，切实提高软件质量属性的框架 。

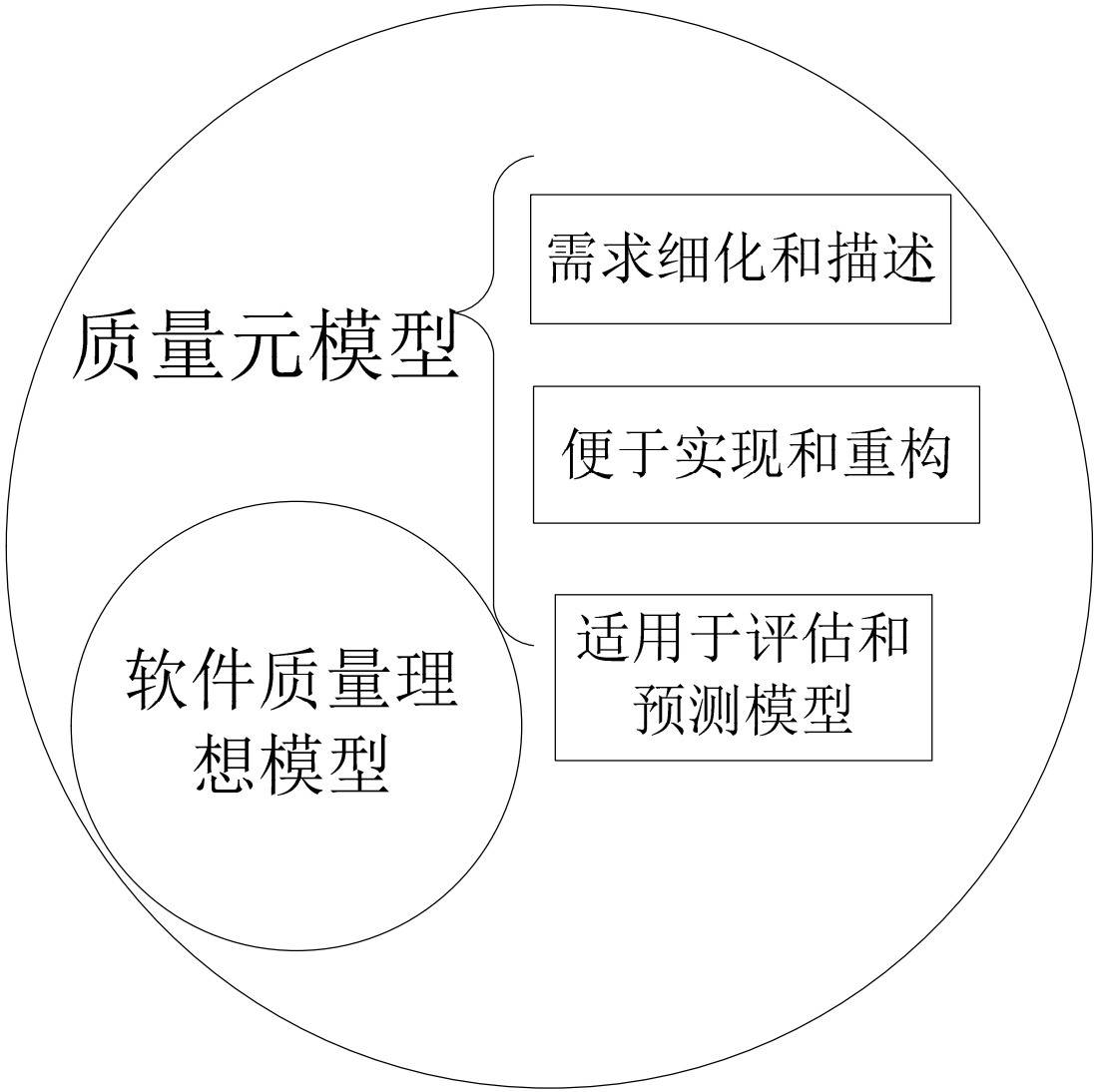


图7.5软件质量元模型结构图

7.2.1通过质量模型描述架构需求

- 3.将质量场景映射到软件质量理想模型
  - 通常，我们获得的质量场景都是以文本的形式呈现的，将质量场景映射到质量模型中需要对质量场景描述的内容进行分解和细化。
  - 对于质量场景中那些很抽象层次的描述我们可以将其规约为抽象场景质量模型，对于特定的质量场景也会将其规约到特定质量的模型中。

7.2.1通过质量模型描述架构需求

- 3.将质量场景映射到软件质量理想模型
  - 软件质量元模型包括了实例化的软件质量理想模型和对质量场景的注解。细化后的质量场景也需要注解描述：
    - （1）这一质量场景的来源，可以追溯到哪些文档中；
    - （2）这一质量场景的目的是什么：商业目标还是业界标准？
    - （3）这一质量场景的改变会影响到其他哪些质量场景。
  - 设计出来的软件质量理想模型也必须要能够被开发人员理解和实现，也需要能够在产生错误的情况下进行重构

7.3 架构设计、文档化和评估

- 7.3.1. 架构设计、文档化和评估是一个迭代过程
  - 架构设计、文档化和评估是一个迭代过程也是架构驱动的软件开发的核心所在。
  - 其步骤如下：
    - 1. 基本架构设计
    - 2. 架构文档化
    - 3. 架构评估

7.3.1. 架构设计、文档化和评估是一个迭代过程

- 1. 基本架构设计
  - 通过获得的架构需求信息，架构师对架构进行设计并通过文档进行记录.
  - 从零开始的软件系统开发的基本架构设计过程：
    - （1）通过功能性需求列表，抽取出架构需求列表和类功能列表。
    - （2）选择进行开发的子系统，将其对应的设计子系统记录到基于功能的架构结构中。
    - （3）通过基于功能的架构结构完成并发架构结构的设计。根据并行单位的分布，从而确定物理架构结构的设计。

7.3.1. 架构设计、文档化和评估是一个迭代过程

- **1. 基本架构设计**
  - 架构基本设计也是一个迭代的过程，有一些设计的决策，需要进行反复的推理，验证以至于重构，直到所有的架构设计都能够满足架构的需求。

7.3.1. 架构设计、文档化和评估是一个迭代过程

- 2. 架构文档化
  - 架构的设计文档旨在方便程序员和分析师的工作。它可以加强软件系统的利益相关者之间的联系，从而确定出满足需求的软件架构。
  - 软件系统的架构结构唯一的体现方式就是它的文档。因此，文档的质量和完整性是架构驱动软件开发成功的关键因素。

7.3.1. 架构设计、文档化和评估是一个迭代过程

- 2. 架构文档化
  - 对架构文档的几点建议：
    - 软件架构的设计文档必须是完整，并且可以追溯的。
    - 整个架构设计文档必须要包含一个显著的起点，同时所有的子系统也必须要链接成一个完整的集合，并提供相应的指针指向子系统内部的详细设计文档。
    - 在架构设计文档中应该有一个预先规定的约束框架。使得软件能够在性能，容错性，可维护性，安全性等方面能够有良好的表现。
    - 架构设计文档要向所有的利益相关者公开。

7.3.1. 架构设计、文档化和评估是一个迭代过程

- 3. 架构评估
  - 评估的目的是分析架构以识别潜在风险并验证设计中已经满足的质量需求。

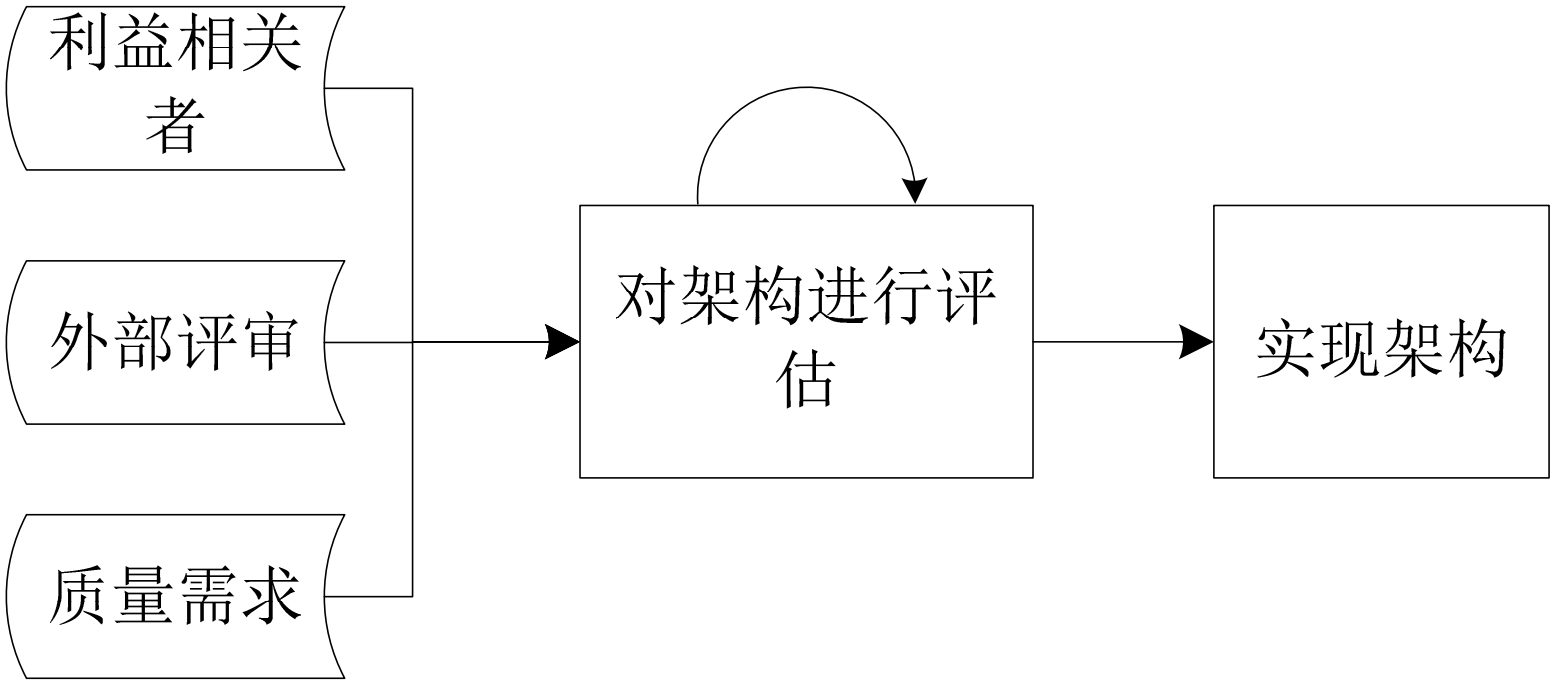


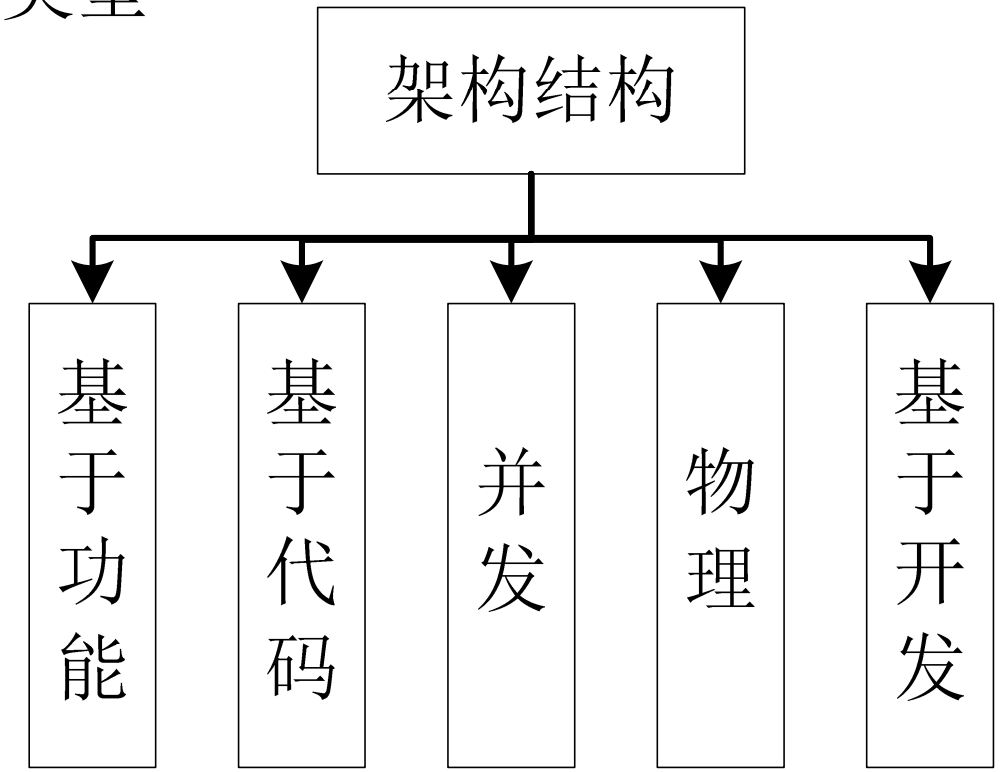
图7-9 对架构进行评估阶段任务描述图

7.3.2. 什么是架构的结构

- 架构结构
  - 通过一定的结构对软件的架构进行描述，把这样的结构称为架构结构。
  - 架构结构描述了架构的基本信息，也包括了类，方法，对象，文件，库等所有需要人做的设计和编码。
  - 架构视图是由架构结构派生而出的，它可以是架构结构的子部分，也可以是多个架构结构信息的综合。

7.3.2 什么是架构的结构

- 架构结构的分类
  - 对架构结构的描述有很多种方法，其中有五种最基本的类型



7.3.2 什么是架构的结构

- 架构结构的分类
  - 基于功能的结构
    - 对软件系统的功能性需求的分解和描述。
    - 在此类架构结构中，“组件”是功能（域）实体；“连接件”是数据的传输实体。
    - 这种结构有助于我们理解功能实体之间的交互，对软件系统的功能性需求有更加深刻的理解，从而设计出满足功能性需求的软件架构。

7.3.2 什么是架构的结构

- 架构结构的分类
  - 基于代码的架构结构
    - 基于代码的架构结构对软件设计过程中关键性代码的抽象描述。
    - 在基于代码的架构结构中，“组件”是包，类，对象，过程，函数，方法等，用于在各种抽象级别封装功能的载体；“连接件”可以使控制流，传输流，共享流，调用关系，对象的实例等。
    - 这种结构对于了解系统的可维护性，可修改性，可重用性和可移植性至关重要。

7.3.2 什么是架构的结构

- 架构结构的分类
  - 并发架构结构
    - 并行架构结构与软件系统的并发逻辑性有关，包含着系统的线程和/或进程的数量，执行时间和优先级等信息。
    - 在并行架构结构中，“组件”是可以细化为线程或者进程的并发单元；“连接件”包括有同步，优先级，数据的传输和互斥。
    - 这种结构对软件系统的运作有着至关重要的作用，同时也有助于实现软件系统的安全性和可靠性。

7.3.2 什么是架构的结构

- 架构结构的分类
  - 物理架构结构
    - 物理架构结构涉及中央处理器（CPU），存储器，总线，网络和输入/输出设备。
    - 可以清晰了解软件系统的可用性，带宽，容量等信息。

7.3.2 什么是架构的结构

- 架构结构的分类
  - 基于开发的架构结构
    - 开发架构结构涉及文件信息和目录信息。
    - 这种架构结构对于软件系统的管理和控制有着重要的作用，同时也可作为团队分工的重要依据。

7.3.2 什么是架构的结构

不同的架构结构与软件质量属性之间的影响关系表

质量属性	架构结构
性能	并发架构结构，物理架构结构
安全性	并发架构结构，基于代码的架构结构
可靠性/可用性	并发架构结构，物理架构结构
可修改性/可维护性	基于功能的不同的架构结构与软件质量属性之间的影响关系表架构结构，基于代码的架构结构，基于开发的架构结构

7.3.3从架构需求出发的评估

- 架构权衡分析法（ATAM： Architecture Tradeoff Analysis Method ）
- ATAM的方法来进行权衡评估是目的是：当需要满足某一质量需求时，会使得到其他的质量需求变差，我们需要权衡这种改变是否值得。

7.3.3从架构需求出发的评估

- ATAM的过程
  - 一个完整地ATAM通常会花费三个整天，其中每天都要进行如下三个过程：场景提出、架构提出和将场景映射到架构上，并进行分析评估。

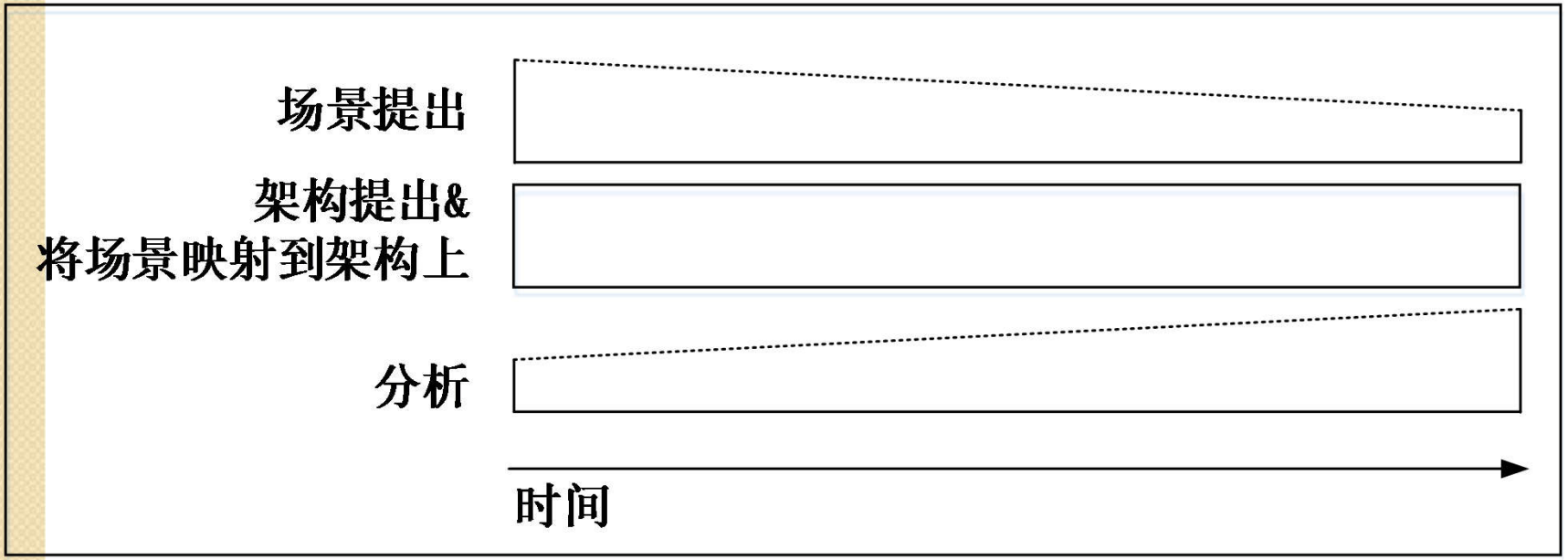


图7-11 ATAM活动及其重要性对比图

7.4架构的实现与维护

- 当将架构转换为代码时，必须考虑所有常见的软件工程和项目管理注意事项：进行详细的设计、实现、测试、配置管理等工作。
- 架构驱动的软件开发的特点之一是软件架构结构与开发团队组织结构具有一致性。也就是说，开发团队的组织结构必须反映到软件架构结构上，反之亦然。

7.4架构的实现与维护

- 只是拥有一个软件架构与拥有一个具有良好文档，良好组织结构和良好维护的架构是完全不一样的。
- 如果没有这些，那么架构将不可避免地偏离其原来的方向。
- 这种风险会导致实现经过精心设计的架构在实现的时候发生不可弥补的错误。