软件体系结构

《软件体系结构作业三》

学　号 22920212204396

姓　名 黄子安

2024 年 3 月 24 日

Reading《Software Architecture4+1》试给出 SA 中 4+1 视图的描述。

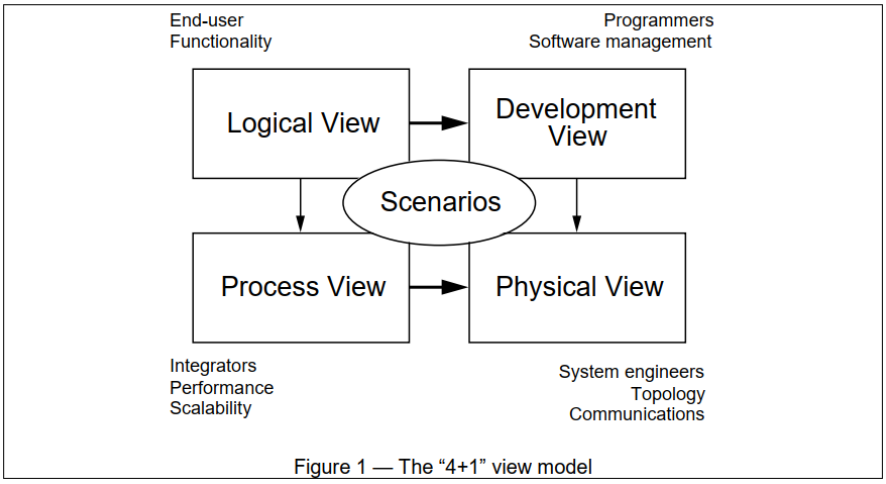

Figure 1 — The "4+1" view model

1. 逻辑视图 Logical View

逻辑视图关注的是**系统的功能**和在使用面向对象的设计方法时使用的**对象模型**。使用逻辑视图来描述软件主要支持功能需求，即系统应该以何种服务形式提供给用户，通过面向对象的封装、继承、抽象等方法实现功能的分析和识别出系统各部分的共同机制和设计元素，实现代码的复用。

逻辑视图对应 UML 中的类图，符号元素包括类、包、接口等，类图展示了一组类及其逻辑关系：关联、使用、组合、继承等。一组相关的类可以被分组到类别中。类模板关注每个单独的类，强调主要的类操作，并识别关键的对象特征。如果定义对象的内部行为很重要，可以使用状态转换图或状态图来完成。通用机制或服务在类实用程序中定义。

2. 过程视图 Process View

这个视图主要是**并发和同步**方面的设计。过程视图关注系统动态运行时，主要是进程以及相关的并发、同步、通信等问题。所以过程视图考虑了一些**非功能性需求**，如**性能和可用性**，同时考虑并发性和分布、系统完整性、容错等问题，对应到 UML 就是时序图和流程图，例如其中有各种元素符号表示消息的同步、异步等

3. 开发视图 Development View

开发视图描述**软件在开发过程中的静态组织**。开发视图侧重于软件模块在软件开发环境中的**实际组织**。软件被打包成**小块（程序库或子系统）**可以由一个或少数几个开发者开发，这些子系统按层次结构组织，每一层为上面的层提供狭窄但是明确定义的**接口**。开发视图主要考虑与开发轻松性、软件管理、重用或通用性相关的**内部需求**以及受工具集或编程语言约束的因素，可以更加合理的进行需求分解、项目进度监控，提高代码开发的效率与可重用性、可移植性和安全性。

4. 物理视图 Physical View

物理视图描述**软件到硬件的映射**，并描述**硬件的分布**。物理视图主要考虑系统的**非功能需求**，如**可用性、可靠性(容错)、性能(吞吐量)和可伸缩性**。软件在计算机网络或处理节点上执行。所识别的各种元素(网络、流程、任务和对象)需要映射到各个节点上，因为期望使用几种不同的物理配置:一些用于开发和测试，另一些用于为不同站点或不同客户部署系统，所以软件到节点的映射需要高度灵活，并且对源代码本身的影响要最小。可以根据对不同硬件资源的性能/容量的诉求，设计出不同物理视图的部署方式。可以对应 UML 中的部署图，展示软件在各个计算机上的分布。

5. 用例或场景 Scenarios

用例视图将上述的四种视图连接在一起，使用重要的场景（通常是用户用例的实例）来促进与其他四个视图协调工作，用例视图的作用是作为在架构设计过程中发现架构元素的驱动器，另外在完成架构设计后作为验证和说明的角色，作为架构原型测试的起点。用例视图的符号元素与组件的逻辑视图非常相似，但是在对象之间的交互方面使用了进程视图的连接器。该视图一般与其它四个视图冗余（因此是 4+1 的形式）