### 《软件架构读书笔记》

#### 1：软件架构的定义：

软件架构是指软件系统的整体结构和高层设计，用于描述系统的组件、组件间的关系以及设计背后的原则和约束。它是软件开发的基础，指导系统的实现和演化。

#### **2：软件架构的核心特点：**

1.抽象层次高：强调系统的全局视角，聚焦于整体结构和关键部分，而非细节实现。

2.组件化：将系统分解为多个模块或组件，每个组件都有明确的职责和边界。

3.关注非功能需求：如性能、可扩展性、可靠性、可维护性、安全性等质量属性。

4.指导性：为开发团队提供约束和决策框架，以确保系统满足需求并具备长期可用性。

#### **3：软件架构和企业架构：**

从代码结构和基础到将代码成功部署到生产环境，与一个软件系统重要元素相关的所有东西就是软件架构。

企业架构一般是指整个组织的中心工作，着眼于如何组织与利用人、流程和技术来使企业有效和高效地工作。换句话说，它是关于企业如何分组或部门，业务流程如何在上层运作，以及技术如何支撑这一切。这跟软件架构形成了强烈对比，因为企业架构没有必要关注技术细节。相反，企业架构可能看重的是如何在整个组织中最好地利用技术，而无需实际介入这些技术的工作原理。

从事企业架构工作所需要的思维方式和软件架构大相径庭”，“企业架构需要更高层次的抽象”，“这关乎广度而非深度，关乎战略而非代码。

#### **:4：敏捷软件架构：**

以敏捷的方式交付软件并不能保证得到的软件架构是敏捷的

理解组织或业务变化的速度很重要，因为这能帮助你决定采用何种架构风格，可能是整体架构、微服务架构或者介于两者之间

#### **5：架构与设计**

“架构反映了一个系统成型的重要设计决策，而重要性则通过改变的成本来衡量。”换句话说，“架构不可能轻易反悔”，那会花费很高的成本。

“重要决策都是‘架构’，其他的都是‘设计’。”

那些重要的决策可能包括：

系统的形态（例如，客户端-服务器，基于Web、原生移动客户端、分布式、异步等）；

软件系统的结构（例如，组件、层、交互，等等）；

技术选择（即编程语言、部署平台，等等）；

框架选择（例如，MVC框架、持久性/ORM框架，等等）；

设计方法/模式选择（例如，针对性能、可伸缩性、可用性等的方法）。

#### **6：软件架构的好处**

让团队跟随一个清晰的愿景和路线图；

技术领导力和更好的协调；

便于与人交流，方便回答与重要决策、非功能需求、限制和其他横切关注点相关的问题；

识别和减轻风险的框架；

方法和标准的一致性，随之而来的结构良好的代码库；

正在构建的产品的坚实基础；

对不同的听众，以不同的层次的抽象来交流解决方案的结构。

#### **7：软件软件架构师“是一个角色，而不是一个级别”。**

“软件架构这个角色，可以是一个人，也可以由团队共同承担”，包含如下内容：

架构驱动力：理解业务目标和管理架构驱动力，其中包括需求（功能性需求和非功能需求）和环境的限制。非功能性需求和限制往往对软件架构有巨大的影响，因此明确地将其纳入软件架构的角色。

设计软件：建立技术战略、愿景和路线图。要理解如何解决架构驱动力带来的问题，创建软件系统的整体结构，并为交付设定一个愿景；另一个是技术选择，有一些组织有一份允许使用的技术清单，你只能从中选择，还要考虑成本、许可、供应商关系、技术战略、兼容性、互操作性、支持、部署、升级策略、最终用户环境，等等。

技术风险：发现、减轻和承担技术风险，保证架构的“运转”。

技术选择其实就是风险管理，当复杂度或不确定性高的时候降低风险，有利可图时再冒点险。所有的技术决策，在做出选择时都要把全部因素考虑在内。

一个架构，如果能满足非功能性需求，在给定的环境约束下有效，能为其他代码提供必要的基础，作为平台能解决潜在的业务问题，那就是管用的架构。

架构演化：贯穿整个软件交付过程，持续的技术领导和对架构的承担。架构师在创建了一个架构之后，还得在项目的整个交付过程中根据不断变化的需求和团队反馈来对其进行演化。

编写代码：参与到软件交付的实践部分。编码为架构师提供了一种与团队分享软件开发经验的方式，从而帮助他们更好地理解如何从开发的角度看待架构。

质量保证：引入并坚持标准、指导、原则等。不仅包括代码评审，还包括引入一些标准和工作实践，如编码标准、设计原则和工具，为项目提供一个基线保证。