链滴

架构师之路二 - 架构设计方法论

作者：jianzh5

原文链接：https://ld246.com/article/1584617222084

来源网站：链滴

本系列文章教你怎么样成为一名架构师，本篇文章目的是让你掌握一套架构方法论，掌握规范的计方法，设计出更好、更稳定的架构设计。
<h2 id="概念解析">概念解析</h2>
在文章开始之前需要先理解几个概念：

什么是方法论？
我们拿到一个输入，然后根据这个输入预期一个输出，把中间这个过程描述出来就是方法论。
所以我们本篇讲的架构师方法论就是架构师先拿到经过需求分析出来的输入，然后完成架构设计，这过程就是架构设计方法论。

什么是设计？
设计是实现意图的书面表现形式，而非口头的东西；
设计是要让实现者能理解设计者的意图，是给别人看而非自己看；
设计是要让不同的实现着做出来的东西差不多；
设计是严肃的，后续实现者不能随意偏离设计

什么是系统架构师
作为系统架构师你需要跳出代码层面的设计，站在更加宏观的角度进行把握。你关注的是整个系统而是其中的一两个查询模块，你看到的元素更多的是： 人 、硬件 、软件 、网络。

<h2 id="业务分析">业务分析</h2>
<h2 id="业务分析概述">业务分析概述</h2>

业务分析是在系统开发之前对系统要解决的业务领域的研究过程，目的是搞清楚该业务领域的概以及业务的运转过程
开发系统的目的一般是为了优化业务流程，使业务运转得更加高效、经济
系统的价值主要在于实施后能够帮助客户带来多少业务价值
不管有无系统，业务通常是不变的

<h2 id="业务分析阶段活动模型">业务分析阶段活动模型</h2>

/b3logfile.com/file/2020/03/image-73672e17.png?imageView2/2/interlace/1/format/jpg"></
>
业务分析阶段是由业务分析师 基于自身的业务知识和类似产品的参考，再结合客户、领域专家咨询和指导输出业务分析阶段的成果，主要包括 领域模型 和 业务型
<h3 id="领域模型">领域模型</h3>
领域模型是对领域内的概念类或现实世界中对象的可视化表示。又称概念模型、领域对象模型、析对象模型。它专注于分析问题领域本
身，发掘重要的业务领域概念，并建立业务领域概念之间的关系。概念比较深奥，其实说白了就是我把基于对业务的理解画成一个类图，并画出这些类之间的关系（面向对象），下面我们看一个实际的域模型然后加深一下理解。

ogfile.com/file/2020/03/image-61a636a1.png?imageView2/2/interlace/1/format/jpg">
在领域模型绘制时经常会出现下面两种典型错误：
• 将待开发系统也放在领域模型里面
待开发系统要不要出现在领域模型中取决于你的业务离开待开发的系统能不能玩的转。举个例子：如开发的是共享单车的信息系统，共享单车离开信息系统肯定玩不转，所以这时候信息系统需要出现在域模型。
• 概念划分不清，关系没有画到位

比如属性画成了类，继承关系搞错
<h4 id="领域模型的作用">领域模型的作用</h4>
领域模型可以整理业务中的概念以及关系，帮助团队中的成员对业务的理解保持一致；往后可以导数据库设计、系统功能设计、指导开发。

现在有种开发模式是基于 UI 指导开发，根据 UI 进行数据库数据库设计、代码编写，我们称之“急功近利式”开发模式。因为 UI 是系统表面性的东西，是异变的，不稳定的，这种模式下 UI 变化我们的的设计可能也需要跟着变。
而右边是基于领域驱动开发，在开发前先去思考业务的本质，先把领域层分析出来，再根据分析来的领域层进行界面设计、架构设计、代码开发，这是由内而外的设计，这样做出来的系统就会比较定。
<h3 id="业务模型">业务模型</h3>
前面讲的领域模型是基于静态的关系，要理解业务其实更多的需要从动态的角度来了解业务运转过程，所以这时候就需要做业务模型。
理解业务模型需要先理解以下几个概念：
<h4 id="业务对象">业务对象</h4>
业务主体主要有 业务执行者、业务工人、业务实体。
要理解这三个对象，我们首先需要知道什么是业务，业务是指一个组织可以向组织外的人供服务。
业务执行者(Business Actor) ： 组织外的人，来享受这服务的人就称为业务执行者
业务工人(Business worker) ： 提供业务的组织的内部支撑人员</strng>
业务实体(Business Entity) ： 提供业务的组织的内部信息系统</strog>
理解这几个概念还是有点拗口，我们来看看下面一张图，一个餐厅对象的业务建模

右边大的黑框是提供业务的组织，是餐厅；图的左边是个业务执行者，是顾客；而在餐厅内部又为业务工人（领位员、点餐员、厨师等），业务实体（餐厅点餐管理系统）
我们在做业务建模时需要注意，所有业务对象在圆圈的右下方需要有个斜线，这是一个建规范
<h4 id="业务用例">业务用例</h4>
概念：组织对外提供的业务服务

一个银行是一个提供业务的组织，这就是业务用例，考察的对象是银行这个组织而不是系统。</
>
<h4 id="业务流程">业务流程</h4>
业务用例是最基本的定义，而要分析业务动态的过程就需要业务流程，我们一般用时序图来表示
餐厅现状的业务流程

这时候所有的动作步骤全部靠人参与
建设系统后的业务流程

有了系统后系统可以承担一部分工作，有了系统能改善业务流程、提高价值、降低成本，这就是strong>建设系统的价值以及意义 ，否则就没必要做系统建设了。
<h4 id="业务流程分析的作用">业务流程分析的作用</h4>

动态表达业务运转的过程

只有很好的理解了业务流程，才能设计出更好的支持该业务的系统
通过对比系统实施前后的流程变化，分析优化点，评估系统价值

<h2 id="小结">小结</h2>
在准备做一个系统之前需要先分析业务，将业务理解清楚。理解业务既有静态的理解（领域模型又有动态的理解（业务流程），只有将业务理解清楚才能做出良好的系统。
<h2 id="需求分析">需求分析</h2>
需求分析是需求工程的环节，整个需求工程分为两大块

前期主要是做需求开发，包括需求调研、需求分析、需求定义；后期需要做需求管理，包括需求认、需求跟踪、需求变更控制。
咱们架构师主要聚焦在 需求分析 和 需求定义 两个环节。/p>
<h2 id="需求分析阶段活动模型">需求分析阶段活动模型</h2>

需求分析阶段是由系统分析师 基于业务分析师输出的领域模型、业务模型 再结合 需求调研成果出需求分析阶段的成果，主要包括 系统上下文， 功能型需求（用模型） 和 非功能性需求（性能等）。
<h3 id="系统上下文">系统上下文</h3>
系统上下文是指系统与周边用户和其它系统之间的关系

系统上下文的绘制很简单，就是将准备开发的系统画在中间，把用户和对接的周边系统画出来这叫系统上下文。
<h3 id="系统用例">系统用例</h3>
系统用例是指系统被使用的案例，主要是从业务流程中推导出来，系统用例的命名规范主要是使动词短语，如：添加用户、查询话费等短语。

我们可以对系统用例细化，如上对登记入座信息这一用例进行细化

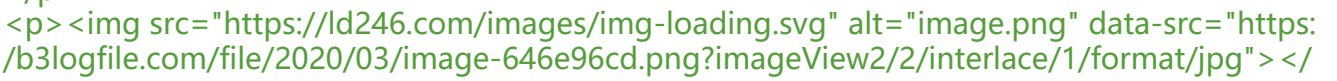
<h4 id="系统用例与业务用例的区别与联系">系统用例与业务用例的区别与联系</h4>

业务用例是描述组织对外提供的能力，系统用例是描述系统对外提供的能力，两者考察的对象不样
系统用例是业务用例相应流程中对系统的一个操作

<h4 id="功能与用例的区别和联系">功能与用例的区别和联系</h4>

用例是需求分析的产物，描述的是某种用户使用系统完成什么业务
功能是设计阶段的产物，是根据系统用例和架构中的组件推导出来的
功能是静态的，用例是动态的
从语法角度来说，用例是动词短语，功能是名词短语
例：用例命名（查询空位），功能命名（空位查询）
常见的错误：描述需求时拿出一份功能清单

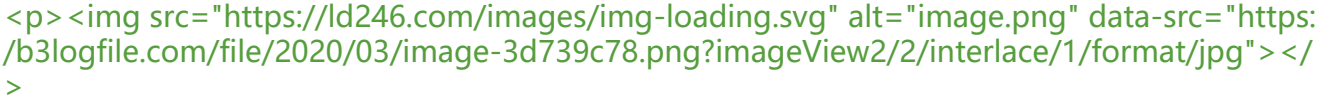
<h3 id="非功能性需求">非功能性需求</h3>
主要是确定一些非功能性需求，比如： 可用性、 性能、 安全性、 经济性、可扩展性、 可伸缩性、可移植性等等
可用性、性能、安全性是需要重点关注的内容，我们后期专门拿出来讲。

<h2 id="架构设计-重点-">架构设计（重点）</h2>
前面的业务分析与需求分析一般是由其他专人来做，那么这一块的内容则是架构师的工作，需要点关注。
在系统简单时我们需要从 功能角度 对系统进行分解和拆分，这个时候我们要做下概要设计和详细设计就可以。
在复杂系统时我们需要从 组件角度 对系统进行分解和拆分，这个时候我们就需做架构设计与概要设计
<h2 id="组件-功能-模块">组件、功能、模块</h2>
组件是架构设计阶段考虑的单元（进程级别），功能、模块是概要设计、详细设计考虑的单元；一个组件可包含多个模块，涉及多个功能； 一个功能的实现可能需要多个组件中的相应模块来协作完


我们用一张图来理解他们三者之间的关系
前后端分离的一个项目从进程角度划分出三个组件，分别是 web 前端、后端接口服务、后台服务，
为了实现用户查询这个功能必须要在相应组件里都需要有响应的模块
一个组件里可以有多个不同的模块，各个组件里的模块相互协作完成某一个功能
<h2 id="架构">架构</h2>
如果用一句话来描述什么是架构，那应该可以这样定义：架构是系统的内部结构（组件以及它们间的关系）还要包含系统的技术要素。做架构设计其实就是干这两件事。

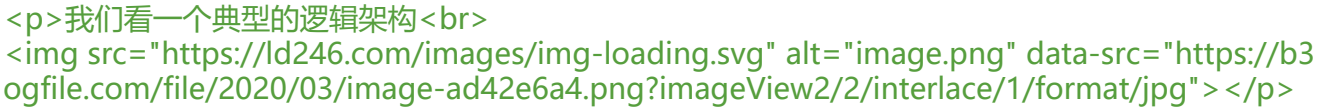

架构设计有两个目标：

满足功能性需求
满足非功能性需求

<h2 id="架构设计阶段活动模型">架构设计阶段活动模型</h2>


架构设计阶段是由系统架构师 参照需求分析的产物（SRS），再通过对系统分析师、项目经理的询输出架构设计阶段的成果，主要包括 架构工作计划 、 逻辑架构</trong>、物理架构、开发组件一览表、部署组一览表、技术选型一览表
如何来衡量一个架构的设计好坏呢？
在设计完成时我们可以通过设计资料的规范性以及设计思路、方案决策、技术选型的合理性来校验
在系统实现后可以通过功能性和非功能性需求的满足程度来校验
<h3 id="逻辑架构设计-非技术型-">逻辑架构设计（非技术型）</h3>

将系统从非技术角度分解成若干逻辑组件，并建立它们之间的关系，以满足系统需求。关系分静和动态，其中静态关系用组件图表示，动态关系用序
列图表示。
逻辑架构中，组件名称使用母语以便理解
逻辑架构不涉及技术元素，只是纯概念上的表述
逻辑架构的读者可以是非技术人员
逻辑架构设计完成后应和系统分析师、产品经理等人员一起确认，检查是否满足需求

我们看一个典型的逻辑架构


<h3 id="物理架构设计-技术型-">物理架构设计（技术型）</h3>

将逻辑架构中的组件转换为技术性的物理组件，名称使用英文，在实现时应遵循这些命名
物理组件粒度有大有小，可表现为子系统、进程、对象等多种形式
物理架构还需要解决非功能性需求
物理架构还要和后续设计和实现进行衔接
非技术人员可能难以理解

我们看一个典型的物理架构

<h2 id="小结-">小结</h2>
架构设计为系统的总体设计，决定了系统的组件划分、关键技术方案决策、技术选型
架构设计上接需求，下接进一步的设计和实现，是决定系统实现的质量、效率、成本的关键阶段
<h2 id="概要设计与详细设计">概要设计与详细设计</h2>
<h2 id="概要设计">概要设计</h2>
概要设计阶段的主要内容是进行功能模块划分以及接口定义（接口名称、功能概要、参数、返回）
<h3 id="概要设计阶段活动模型">概要设计阶段活动模型</h3>

概要设计阶段是由开发组长 基于系统用例、开发组件一览表 再结合对架构师和系统分析是师的询输出概要设计阶段成果，主要包括 功能一览表， 接口说明书。
<h2 id="详细设计">详细设计</h2>
详细设计阶段的主要内容是描述内部模块实现、界面设计以及数据库设计
<h3 id="详细设计阶段活动模型">详细设计阶段活动模型</h3>

详细设计阶段可能会根据工作内容进行分工，主要结合之前的产出输出详细设计阶段的成果，主包括 界面设计， 模块内部设计， 数据库设计/strong>。
<h2 id="后续工程阶段">后续工程阶段</h2>
<h2 id="开发阶段活动模型">开发阶段活动模型</h2>

开发阶段主要是由开发人员结合架构设计的产物以及详细设计的产物编写相应代码。
<h2 id="测试阶段活动模型">测试阶段活动模型</h2>

测试阶段主要是由测试人员结合架构设计的产物以及详细设计的产物进行功能测试，包括功能性求以及非功能性需求，需要对外输出 测试计划，测试用例以及 测试报告。
<h2 id="部署阶段活动模型">部署阶段活动模型</h2>

部署阶段主要是由部署人员结合架构设计的产物以及跟开发人员的咨询进行组件部署，这一阶段要输出部署计划、部署方案、部署手册、部署脚本、部署实施 等

<h2 id="运维阶段活动模型">运维阶段活动模型</h2>

运维阶段主要是由运维人员结合架构设计的产物进行系统运维，需要输出运维计划、运维方案、运维手册、运维脚本、运维报告 等