28 | 节点四：架构规划之如何确认规划完整性？

2022-04-05 郭东白

《郭东白的架构课》 课程介绍 >

讲述：郭东白

时长 18:15 大小 16.72M

你好，我是郭东白。这节课我们来讲架构规划的最后一个环节——规划确认。

在上节课，我们已经梳理出了一组必保需求，然后从这组需求中推导出了一组任务。并且，我们以最大化某个项目目标的方式，将这组任务分配到了执行团队中。

完成这些工作后，我们距离整个项目的正式启动就只有一步之遥了。而这个重要的一步是对整个架构活动的规划做确认。这个环节的价值在于：

是控制风险的重要机会。

是保障交付的重要手段。

> 是为团队和企业沉淀知识的重要时机。因为在这个环节，你和其他参与者会产生大量的规划文档。而这些文档，就是沉淀知识的利器。

可以说，通过架构规划中的确认环节来控制风险与保障交付，就是我们在这个环节的核心关注点。具体而言，规划确认包含八个部分。接下来我们就来详细拆解一下。

定稿架构规划文档

在规划确认之前，你已经收集了几乎所有与架构规划相关的文档。那么规划确认，也就是定稿架构规划文档的过程。

在定稿的过程中，你可能会和不同团队、企业外部专家产生诸多交互。这个时候你就需要与执行者确定规划内容。因为之前的收集主要是作为规划的输入，而不是执行者的承诺。所以你就需要**把输入转成一个可行且有约束力的规划。**

架构师的角色有点像律师。除了最小化交付风险外，还要确保所有参与者有能力且有意愿履行他们的责任。怎么确保呢？答案是**为参与者拟定一个合同**。

这是一个用来提升交付确定性的切实有效的工具。规划确认过程中的每一个交付项，比如用例文档、必保任务、领域模型、API 设计、消息和数据流、整体交付节奏和完整性验证等，都指向同一个目的，即**提升交付的确定性**。这也是架构规划的实质。

组织用例文档

用例文档是关于交付内容的最简洁的描述。它的作用是描述架构活动中某个团队或者小组要为某个用户角色，在某个场景中，创造出某种价值。举个例子：

> 商家团队要为中小商户，在店铺经营过程中，提供有明确行动点的生意参谋功能，从而提升商家的经营效率和成交额。

需要格外注意的是，无论是写用例，还是用一张图来表示用例，都需要避免堆积。在一个架构活动中，不论是十个人还是上百人参与，最顶层的用例**最多也不能超过十个**。

我们可以把这些用例组织成一个树状结构，从粗到细有数百个节点。但是到了最顶层，节点最多不能超过十个。这些节点，是你作为架构师所要保障交付的价值，因而必须跟架构

活动的目标相匹配。如果一下子拿出上百个价值点，然后去跟决策者和赞助者做确认。要知道，他们根本就没这个时间。

当然，并不是说我们不需要关注细节。一个节点可能会细分成十个子节点，甚至继续细分到更多层。但是从交付保障这个层面来说， 我们只需要保障顶层的十个节点，以及这十个节点之间的强依赖即可。其他的子节点，则需要相应的人员来保障交付。在互联网时代，架构活动是一个时间高度紧张、交付风险极大的场景，架构师不应该把注意力过多地放在几十甚至上百个节点的交付情况上。

这些用例描述的作用是，确保所有研发人员都能**对各自所交付的单元目标有清晰的认知**。拿我们刚才的例子来讲，在这个把多个业务的商家工具做整合的架构活动中，用例的作用就格外重要。

想想看，整合之后，每个垂直业务线的商家，都能通过这个大一统的商家工具的生意参谋功能，来获得各自清晰的行动点。这样一来，度量这些行动点的就有了办法。相应地，提升商家经营的效率和成交额就不是什么问题了。正如我在上个模块中提到的，关注商业价值是架构师最重要的生存法则之一。因而在架构规划中，我们必须要锁定整个架构活动的商业价值。

不过有时候也没必要这样做，因为整个架构活动就只有一个价值，比如稳定性治理、审计合规或者商业模式升级。

对用例的描述除了要尽量简洁外，还要**将其整理成一个文档**。这样的话，几乎所有人都能通过阅读这些用例来获得更为宏观的视角。而具体的每个场景的细节描述和需求文档，可以通过链接记录到另外的文档中。

我曾经见过上百页的用例文档，读者很快就会迷失在细节中。我不太相信有人能完整读一遍这样的文档。

确认必保任务的交付节奏

在架构规划的环节，我们还需要重新梳理一遍在任务边界划分中，做出来的具体的任务描述。并且要确保这些必保任务**录入到了 Jira 之类工具中**。

借助这个工具，我们需要收集所有必保任务，并确认任务分配和交付排期。最终，我们需要通过这个工具来跟踪所有必保任务的交付情况。 需要注意的是，这些必保任务也要和用例形成关联关系。

鉴于这个环节是标准的研发流程，我们在这里就不赘述了。

确认领域模型

同样，在这个阶段之前，你准备的领域模型也是一组输入。那么在规划确认这个环节，你就需要完成定稿。

这是一个统一语义的过程，整个架构活动只能有一个问题域模型。虽然不同的执行者可以帮你梳理领域模型，但是你必须把整个领域模型整理到同一个语义环境中。这是对于架构活动中要解决的问题的准确描述。

另外，你必须**让最终的执行者来确认**领域模型。因为之前帮你起草领域模型的，不一定是最终的执行者。

确认 API

有了用例文档、必保任务和领域模型，接下来就可以请各个团队完成 API 设计了。如果架构活动中主要使用的是 RESTful 框架，那么 **Swagger** 就是一个非常好的选项。

相比 Wiki，Swagger 的优点是：

1. 有约束性。我们刚才把架构师在这个环节的角色看作律师，那么我们就要看看执行方能提供什么样的服务。
2. 易读易用。对于大多数程序员来说，读代码要比读文档更便宜。
3. 有 Copyright 和 Ownership。研发人员非常在意自己的口碑，一般比较资深的研发都会在 API 的定义上面下很大的功夫。不像 Wiki，很容易变成一个 Group 文档。
4. 有投入度。连 Swagger 都不想写或者写不出来的人，估计你未来也很难撬得动他，甚至也用不上这样的人。所以在 Swagger 上的投入，能让你提早发现资源和能力上的问题。
5. 规范性好，很容易在团队中标准化掉。

6. 可测试性好，容易验证其完整性。
7. 长期回报大。仅架构活动本身，对定义者本人的价值就很大，可以帮助他想清楚问题。而对于依赖这个 API 的人来说，价值就更大了。他可以及早做 Mock 测试，及早给出反馈意见，避免在很晚的时候才发现集成问题。长期来看，还可以让整个团队形成好的设计习惯，从而提升整体的 API 质量。这是个典型的有复利的编程模式。

顺便提一下，测试这个职能并不是为研发人员服务的，而是为用户服务的。在企业内部，测试人员是为产品服务的，而不是为研发服务的。所以我建议你在 Swagger Review 时，要尽量动员测试人员，让他们加入进来，并及早给出修正建议。

确认消息和数据流

在确认好 API 之后，接下来就要去确认消息和数据的流转了。也就是某个角色在某个时间能为某个使用方提供某些消息和数据。

消息，是除了 API 调用外服务间最常见的通讯机制，甚至可以说是过分常见了。事实上我一直在怀疑，过去大多数利用消息解耦的场景，在今天的计算能力之下，到底还是不是一个好的设计模式。

尤其对于一个体量较小的公司来说，消息队列带来的编程、维护、状态查询和故障恢复的复杂性，一般来说不值得采用这样的设计。不过我没有在百人规模的初创公司工作过，也没有什么发言权。欢迎有经验的同学在评论区分享一下。

不过在大公司里，消息的确是一个非常不对等的合作机制。往往是某个负责核心实体的团队比较强势，通过消息机制和其他服务解耦来降低稳定性的风险。

这个团队是消息生产方，只管发送消息。而接收方呢，也自动认为消息队列永远是高可用、低延迟和不丢失的。至于怎么做到这一点，无论发送者还是消费者，似乎都不大关心。往往是消息积压之后，才会跳出来做各种清洗操作。

无论如何，消息机制在很多需要解耦、人工审核或者风控这样的异步场景中，还是很有必要的。你作为一个架构师，也应该关注消息机制的设计和使用。不过多数时候， 你只需要确认消息的生产和消费机制畅通，保证它们不影响现有的场景即可。而通过消息通讯的双

方，则需要在消息的内容、发送频次、消息的延迟要求、监控责任、幂等机制和消息的确认模式等方面达成一致。

总体而言，很多公司的消息机制缺乏规范性，治理起来又非常困难。因而我不建议你把消息机制作为一个数据传输的首选机制。

要知道，在互联网时代，数据是个核心资产。所以上线不仅仅是功能上线，还要保障监控、数字工作台、实时和离线业务分析等能力同时上线。与此同时，数据的采集、清洗、整合、加工、可视化和质量监控报警的能力，在这个阶段也要确认完成。

这是个常规工作，没太大的技术难度，难的是保障资源投入、模型质量和数据质量。所以如果在这个阶段就及早规划，会省去后续很多麻烦。

需要留心的是，在一个大型的架构活动里，往往会碰到**数据共享**这样一个比较棘手的问题：哪怕在公司内部，有时数据的拥有者也不愿意把数据分享给其他团队。有时候是出于管控和数据安全的担心，不过更多时候是为了维护团队的利益，尤其是那些喜欢赛马的企业。

因为参与赛马的团队，本质上是竞争关系。那么有先发数据优势的团队，肯定不太乐意把自己的实时数据分享出去。但是有些架构活动就要求团队之间做数据打通，那么到底打通什么呢？就必须在这个阶段讲得明明白白。不过你作为一个架构师，在数据分享的内容和范围这件事情上，是没有决策权的。所以最好由决策者亲自拍板，然后再走正式的邮件审批流程。

如果说数据的分享机制还要通过一个数据中台或者数据仓库，那么数仓的同学也必须参与到这个合同确认的环节里来，确保数据生产方、管理方和数据使用方能够在各个方面达成一致。包括埋点要求、指标定义、必须支持的分析场景、模型定义、数据的及时性和准确性、数据清洗的分工和数据权限管理等。

在互联网时代，数据资产和商业分析团队永远是核心场景。如果做错了，未来还是要改。所以越早请相关方参与到设计的确认中，对数据服务的质量提升就越大。

确认强依赖任务的交付节奏

最终，你其实真正想得到的，就是所有执行者都能说这么一句话：

> 我承诺，只要依赖方能在 XX 年 X 月 X 日前，按质量完成所承诺的 API 和数据流。那么我也以我的口碑、年终奖和晋升担保，我会在 XX 年 X 月 X 日，按质量交付我的所有 API 和数据流。

这样的承诺肯定是不太现实的，不过过程管理工具会帮助你完成依赖梳理、交付项描述、时间承诺的梳理和跟进。

事实上，这句承诺也侧面反映了项目规划中的一个常见风险点，即**强依赖任务的交付**。 虽然通过依赖解耦和 Swagger 的应用，能让你并行处理一些工作。但是真正的集成，以及对异常情况的处理，还是需要完成强依赖任务后才能进行。所以确认强依赖任务的交付节奏，是你这个架构师、项目经理和执行者在各个用例层级上都要进行的任务。

确认整个架构规划的完整性

整个架构规划的完整性确认，需要测试和相关团队核心人员的介入，从而确保核心场景的核心用例能被现有的功能所覆盖。同时也要确认 API、消息、数据是完整和兼容的，整体集成风险是可控的。

在这个环节，我一般不太关注边界条件的梳理，主要是担心大家会把过多的注意力分散在较小，甚至是比较难的异常情况梳理上，而在核心场景和强依赖任务的梳理上投入得不够。

我一般会让团队内部来梳理边界条件，一旦他们发现了影响其他团队的重大风险存在，就可以拿到整个项目组的层面上来讨论了。

我在这个环节反倒不太去关注重大风险了。原因有两点：

大多数工作都在规划确认这个环节之前完成了。

架构师的注意力无法关注到领域内，所以领域内的风险，需要交给各个执行方去解除。

最终我们要达到的结果是**有人有图有承诺**，这才算是合同签署完毕。需要注意的是，这张图要尽量完整，每个模块都有一个 Owner，也就是我们在边界划分里确认的执行者。每一条边，不论是 API 调用、数据流，还是消息机制。都会有一个人承诺：

> 我会在某年某月的某一天，完整交付你可以依赖的完整的一条边。

小结

到了这里你可能已经能总结出来了，规划确认环节的王道，就是**通过精细规划来控制风险，保障全面启动前交付风险的最小化**。

那么该怎么做呢？答案是靠正确的取舍，将注意力放在核心场景、强依赖和交付主链路上。同时，也要尽量将这个确认环节，变成一个**分布式的并行确认**的过程。一来可以争取时间，二来可以提升参与者的担当和投入度。

远大规划是由目标来决定的。在架构规划中，你这个架构师的责任不是让目标更伟大，而是要确保交付风险的最小化。我见过太多激进的架构师了，他们不但不控制项目的范围，反而会在规划环节持续放大范围。这么做，既不符合互联网时代小步迭代的原则，也不符合系统论中复杂度控制的原则。

有些大项目是很空的。当我们没办法把一个大项目拆分成一组小项目时，就搞了一个大项目出来。而一个项目里，最可怕的就是架构师把项目规划得越来越大，恨不得让公司所有研发人员都参与进来。你千万不要这么做！

思考题

三个思考题，选择你认为最有价值的一道来分享一下吧。

1. 作为架构师，在冒险精神的价值观下，你认为最小可用的规划是什么？你会选择忽略哪个环节呢？为什么？
2. 作为执行者，你最反感什么样的规划？
3. 计划永远赶不上变化，哪怕是再完美的规划也敌不过变化。你见到最夸张的变化是什么？这种变化可以通过规划来应对吗？为什么？

如果这节课对你有帮助，欢迎你把课程转发给你的同事或朋友。顺便打个小广告，我刚开了个人抖音号，我会定期发表一些比较新、但是不一定那么成熟的观点。欢迎在抖音上搜索“郭东白”并关注，也欢迎你的批评指正。我们下节课再见！

上一篇 27 | 节点四：架构规划之如何划分任务边界？

下一篇 对话于冰（上）| 怎样成为那个有准备的人？

精选留言 (6)

写留言

罗杰

2022-04-07

对于我现在的公司，没有运维，个位数的开发，想要去学人家大公司搞 K8S，服务风格这种东西，真是想都不要想了。利润是开发团队成本的十倍到二十倍，就这也无法说服给团队成员购买正版的开发工具。

1

peter
2022-04-07

请教老师几个问题啊：
Q1：与参与者签合同，具体是怎么做的？纸上签字吗？还是口头承诺？
Q2：消息机制不是数据传输的首选机制，那首选机制是什么？
Q3：多大规模的公司适合用消息？比如极客时间适合用消息吗？
Q4：老师一般用什么过程管理工具？

展开

聪明的傻孩子
2022-04-07

3.系统性风险最难避开，这种很难通过规划来避免，只能避开；比如最近教育部对于课外补课的打击，造成教育行业的大规模裁员和业务下线

罗均 - Jun
2022-04-05

再次感谢老师非同凡响的精彩课程：
1. 精炼清晰的用例文档。
2. 任务边界清晰的分工与计划。
3. 产品、交付与运营都达成共识（或许三方不同的语义环境）的领域模型。
4. 最爱的swagger。...

展开

spark
2022-04-05

郭老师, take away~~~商业、场景、用户体验，产品经理首先需要成为架构师才行~~~商业的阶段、增长路径、战略意图，目标确认，洞察力，少就是多，问题的定义，这些要素是确认架构的前置条件。举个例子，我是如何确认架构规划完整性的？一个项目，我写了将近20个PPT，每个PPT我只讲第一页，不超过3分钟。我们公司总裁拿着我的PPT都要讲4个小时以上，消化和确认我写的内容~~~

展开

墨白™
2022-04-05

在26和27小节，部分内容我没有看懂，应该是跟接触的少有关系。这一节结合我的工作来看，就明白的很多。东白老师的这篇专栏，真的质量很高