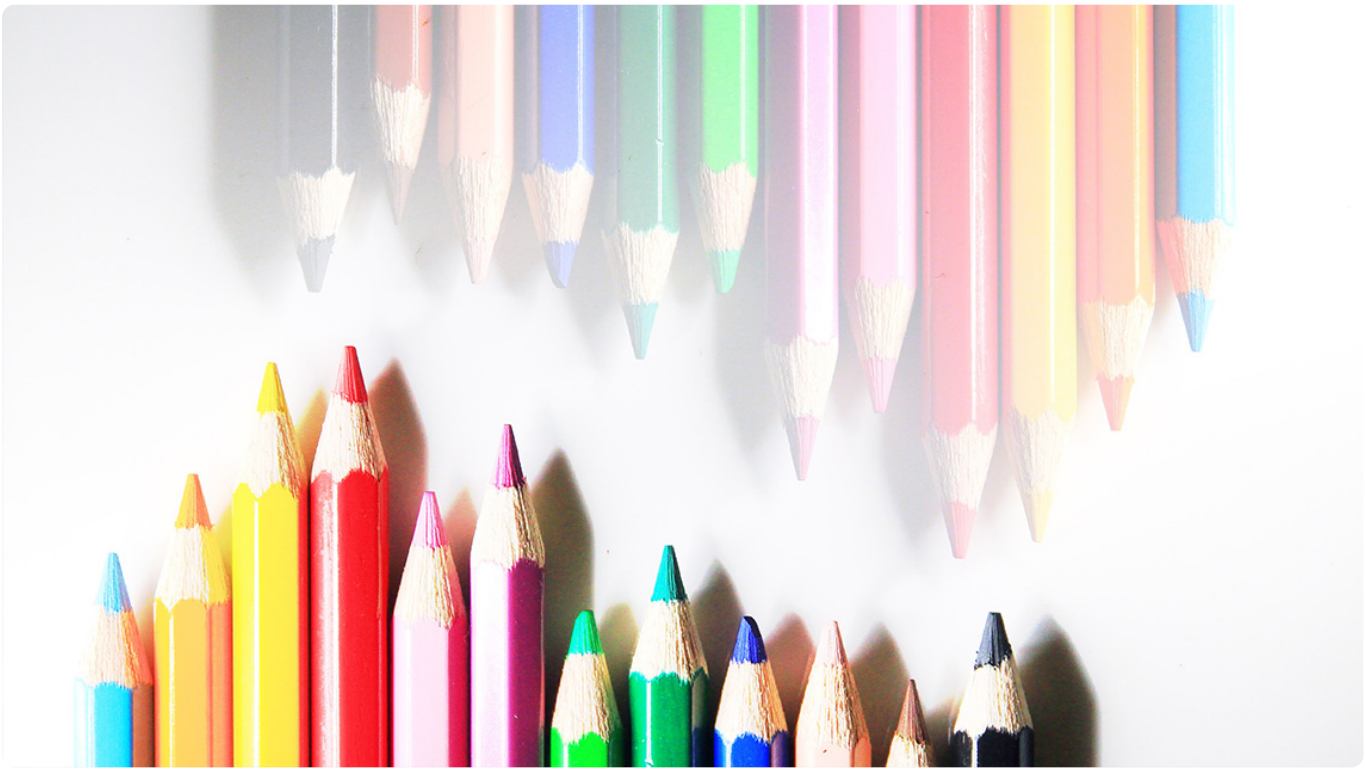
22 | 如何为项目做好技术选型?

2019-04-18 宝玉

软件工程之美 进入课程 >



讲述：宝玉
时长 13:29 大小 12.36M

你好，我是宝玉，我今天分享的主题是：如何为项目做好技术选型?

在架构设计过程中，肯定绕不开技术选型这个话题，大到架构、框架、语言选择，小到用什么组件、设计模式。

这也是最容易引起争议的话题，无论是现实中还是网上，到处有各种语言、框架的争论：Java 好还是 C# 好？前端框架是 Vue 好还是 React 好？跨平台手机开发，该选 React Native 还是 Flutter……

虽然这种争论从来没什么结果，但当你做技术选型时，却很容易受到这些信息的干扰，尤其是你身边有几个某种语言或者框架的狂热粉丝的话，他们会不停地在你旁边吹风，说他喜欢的语言或框架的各种好处。

包括我们自己做技术选型时，也会有很多个人偏好在里面。比如我以前对微软技术栈特别熟悉，也特别喜欢，做技术方案就会偏向微软技术栈；我喜欢 React，做前端技术选型，也会偏向 React 的方案。

通过上一篇架构设计的学习，我们知道，**架构设计的主要目标，是要能低成本地满足需求和需求变化，低成本地保障软件运行。**然而对技术的个人偏好，很可能让你在技术选型时，忽略架构设计的目标，导致满足需求的成本变高，或者运行成本居高不下。

所以今天，我们一起来探讨一下，在软件工程中，怎么样才能避免这种选型的倾向性，科学客观地做好技术选型。

技术选型就是项目决策

技术选型，就是在两个或者多个技术方案中选择适合当时项目情况的方案。**技术选型看起来是个技术的选择，但其实是一个和项目情况密切相关的项目决策。**

在项目中，除了技术上的选型，类似的选择也有很多，比如说产品设计中：某个功能该不该加？该选哪种动画效果？比如制定测试方案的时候，选择哪一种压力测试工具？选择哪个测试框架？这些选择，本质上就是一种项目决策。

要做好技术选型，就是要做好项目决策。那么怎样从做项目决策的角度来选择合适的技术选型呢？

受制于时间、范围和成本的约束

我们在《08 | 怎样平衡软件质量与时间成本范围的关系？》中学习了项目金三角的理论，也就是项目受制于三个因素：时间、范围和成本。

技术决策作为一种项目决策，也要受制于时间、范围和成本，在决策时不能超出这三者的边界。

比如说在项目时间紧时，决策上就要偏向能提升开发速度的技术；在成本吃紧的情况下，要多用成熟的免费的框架、工具，避免用贵的商业软件或者自己造轮子提升成本；在范围大、需求多的情况下，架构就要考虑如何能简单快速完成需求。

还要注意一个问题就是随着项目的推进，其实制约项目的三个因素一直在动态变化，需要及时根据情况调整技术决策。

举个例子来说，2004 年飞信 PC 客户端做第一个版本的时候，那时候主要约束是成本，只有一个 C++ 程序员，这个程序员会用什么技术就用什么技术，谈不上选型。

到 2005 年做第二版本时，有了几个人，但是时间上要求快，所以就选择了能提升开发速度的 C# Winform 技术方案。到 2008 年做第三版时，人手充裕了，也没有进度上的压力，这时候主要就追求用户体验、性能，所以又选择了 C++ 的技术方案重新开发。

要分析可行性和风险

我们在专栏前面的内容中学习了可行性研究和风险管理的知识。如果在项目决策时，不考虑可行性，不预估风险，就极有可能导致决策失败。

就像在《09 | 可行性研究：一个从一开始就注定失败的跨平台项目》那篇文章中的案例，技术选型时，没有考虑到 License 的法律问题，导致项目失败。还有在《14 | 风险管理：不能盲目乐观，凡事都应该有 B 计划》那篇文章中的案例，选择 React 时，没有考虑到可能导致的风险，导致项目延迟。

当然，换个角度说，如果在项目中，选择新技术的风险可以接受，也能满足时间、成本和范围的约束，还可以达到丰富团队技术栈的目的，那也是可以的。

要考虑利益相关人

在做项目的决策时，如果决策时没有人代表利益相关的人，就可能会做出不考虑他们利益的决策。

选择适合的技术选型时，也要考虑到这一点。比如说光顾着选用新酷的技术，而没有考虑客户的利益，导致成本增加，进度延迟；比如在选择 UI 组件时，只想着哪个调用方便，而不考虑产品经理的利益，导致产品体验不好。

项目决策中常见的坑

无论是技术选型也好，还是其他项目决策，经常会遇到一些坑，一不小心就会踩上去。

把听到的观点当事实

现在网上充斥着各种观点：一个 React 的粉丝会给你描述 React 的各种优点，而不会告诉你学习曲线有多陡峭；一个不喜欢微软技术的程序员会把.Net 贬低的一文不值；一篇鼓吹 Mongodb 多好的文章可能是收了钱的软文。

每个人都有自己的观点没有问题，但是不能把观点当成事实，尤其是在做决策之前，至少需要验证一下。

> 先入为主，有了结论再找证据

在做技术选型或者项目决策时，还有一个问题就是可能心中已经有了答案，后面所谓的决策，不过是寻找有利于自己答案的证据。比如说我特别喜欢 React，在做技术选型时，就会拼命寻找对 React 有利的数据作为证据，这其实可能会导致结论并不客观。

所以当你选择技术选型的时候，要像做项目决策一样思考分析。要想你的决策能正确，就要注意项目中范围、时间和成本的约束，要分析可行性和风险，要考虑利益相关人，最后还得要避开常见的一些坑。

如何做好技术选型?

现在我们知道了要像做项目决策一样，去选择适合自己项目的技术选型，那么具体该怎样做呢?

我们在《02 | 工程思维：把每件事都当作一个项目来推进》中学习了工程思维和工程方法，在《12 | 流程和规范：红绿灯不是约束，而是用来提高效率》中学习了流程规范。对于技术选型问题，我们一样也可以考虑借鉴工程方法设计一套流程，基于流程去做技术选型或项目决策，来保证整个过程能科学可行，充分考虑项目决策的特点，避开常见的坑。

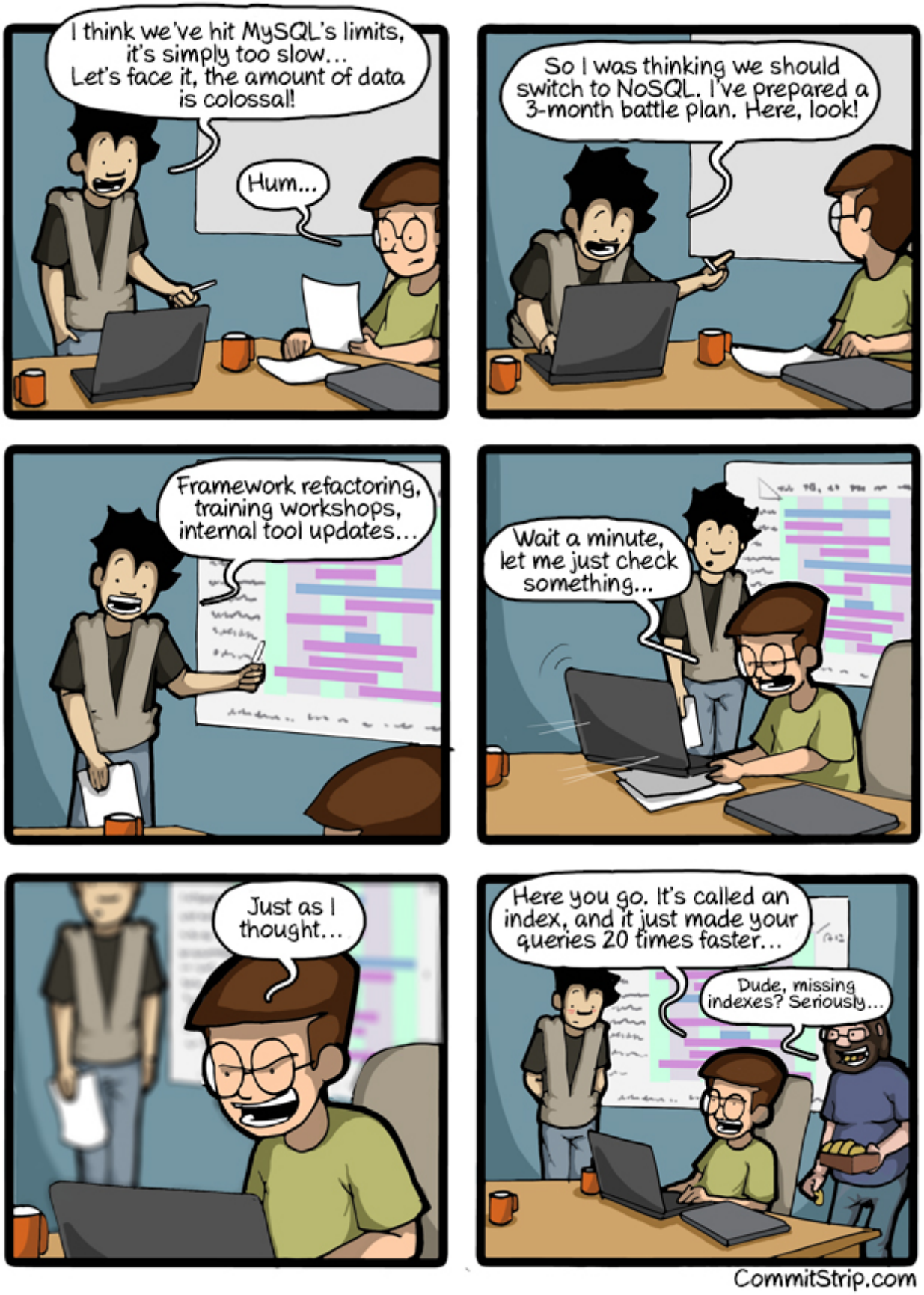
对于技术选型包括项目决策类的问题，我们可以分成：问题定义、调研、验证、决策这几个阶段。

问题定义

在问题定义阶段，需要搞清楚两个问题：为什么需要技术选型？技术选型的目标是什么？

以前看过一个技术漫画《The problem is not the tool itself》。

“我想我们已经达到 MySql 的极限，非常慢...我们得面对数据巨大的事实”，“我们应该迁移到 NoSQL，我已经准备了一个 3 个月的战斗方案，瞧瞧这”，“框架重构、培训、内部工具升级”，“等等，让我检查一下”，“你忘了建索引，加上索引可以让速度提升 20 倍”，“忘记索引？你可真行。”



（图片来源：Commitstrip）

这种事情在软件项目中可不少见，很多时候为了解决问题引入一个新技术，然而真的需要吗？也许我们可以基于现有技术方案进行优化，根本就不需要引入一个新的技术或新的框架。

还有一个就是技术选型的目标需要明确，你的技术选型目标是为了使用新酷技术呢？还是为了提升开发效率？还是为了降低开发成本？

只有明确了技术选型的目标，才能有一个标准可以来评判该选择哪一个方案。

就像上面例子中提到的第二版的飞信 PC 客户端，目标就是要提升开发速度，所以就选开发效率高的 C#。

调研

在明确技术选型的目标后，就可以去调研，看有哪些技术选型可以满足目标，包括开源的方案和商业的方案。

在调研时，可以参考前面“项目决策的特点”中的内容，从几个方面去分析：

满足技术选型目标吗？

满足范围、时间和成本的约束吗？

是不是可行？

有什么样的风险？风险是不是可控？

优缺点是什么？

在调研结束后，可以筛选掉明显不合适的，最终保留 2-3 种方案留待验证。必要的话，可以一起讨论，最终确认。

验证

一个技术是不是合适，如果不够了解，没有应用过的话，实际用一下是很有必要的。可以通过一个小型的快速原型项目，用候选的技术方案快速做一个原型出来，做的过程中才能知

道，你选择的技术选型是不是真的能满足技术选型的目标。

就像前面举的飞信 PC 客户端的例子，在决定第二版本是否使用 C# 开发时，其实做了大量验证工作。当时.Net Framework 还不普及，要打包整个.Net Framework 到安装包里面，这体积就太大了，这是一个很大的问题。

后来发现有一个产品叫 Salamander，它可以只打包程序所需要的 dll 库文件，这样体积就可以控制在 20mb 以内，最后在制作安装包时，用 7zip 压缩，就可以让安装包控制在 10mb 左右。在验证阶段，证明了安装包体积是可以缩小的，基于 C# 开发是可行的，才最终选定了 C# 的技术方案。

决策

在调研和验证完成后，就可以召集所有利益相关人一起，就选择的方案有一个调研结果评审的会议，让大家提出自己的意见，做出最终的决策。

必须要承认，对于技术选型来说，是有不确定性的。即使通过上面的流程，也一样可能会做出错误的决策。但有一个科学的流程，至少可以保证提升做出正确决策的概率。

如果遇到很纠结的情况，就需要负责决策的人来拍板了，这时候其实并不一定有对错，重点的就是做出一个选择，然后按照选择去执行。有时候迟迟不选择、不拍板才是最坏的结果。

在项目结束后，也要对之前技术选型和项目决策做总结，不断的完善技术选型和项目决策的机制，帮助未来更好的进行决策。

总结

今天，我带你一起探讨了技术选型的问题。技术选型，本质上是项目决策的一种，也符合项目决策的一些特点。也就是说，技术选型的选择要受制于范围、时间和成本的约束，要分析可行性和风险，要考虑利益相关人。还有一些坑要小心避开，比如要避免把听到的观点当事实，要验证；要避免先入为主，不要有了结论再找证据。

要做好技术选项，要有一个科学的流程，首先要明确技术选型的目标，避免没必要的引入新技术；然后要充分调研；还要对备选的方案进行验证；最终和利益相关人一起决策。

技术选型，也不要太过于纠结，要勇于决策，选定了就坚定的去执行。

课后思考

你所在项目中是如何做技术选型的？有哪些不错的方法和原则？你在技术选型时，通常是选择保守的还是新酷的？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

上一篇　21 | 架构设计：普通程序员也能实现复杂系统？

下一篇　23 | 架构师：不想当架构师的程序员不是好程序员

精选留言 (16)

写留言

bearlu
2019-04-18

5

主管会C++，然后什么都是C++😂

展开

作者回复: 这确实是个事实，很多技术选型就是技术负责人擅长什么、喜欢什么而决定的。

但很多时候这也是有原因的，先抛开个人喜好的因素不说，初期的时候团队人少，没有什么好选择的，只能是负责人会什么就是什么。后面团队虽然壮大了，但是更换语言或者技术平台成本就高了。

这篇关于Youtube选型Python的文章就很有意思，也是类似的情况。
《在大型项目上，Python 是个烂语言吗？ - 布丁的回答 - 知乎》
https://www.zhihu.com/question/21017354/answer/652602653

小伟
2019-04-18

4

老师后面会有介绍文档编写吗？我们组现在有mrd/prd/概要设计/接口设计/详细设计。

我的疑惑是其他团队不是这个样子的，我想问下，比较规范的文档有哪些，他们功能分别是什么？

展开

作者回复: 很抱歉我们专栏后面没有介绍文档编写的文章了。

对于瀑布模型，每个阶段结束后，都有相应的验收文档，而敏捷开发则没有那么多硬性的要求，而是根据项目需要，写必要的文档。

你说的这些文档已经算比较全面了，有些团队对于测试阶段，会有测试用例文档、测试验收报告，发布前还会有部署文档、维护手册，但现在这类文档基本上被测试工具、部署脚本替代了，也没有什么存在必要。

我觉得项目中必要的文档，主要包括这几类：

1. 设计类文档
这类文档主要用来说明、讨论需求设计、架构设计，可以用来了解、讨论和评审，以及记录后续结果。

2. 说明类文档
这类文档用来对规范、API、配置、操作等做说明，便于规范和统一

3. 报告类文档

对事情结果的报告和说明，比如说验收报告、故障报告、调研等。

而这些文档的价值，在于帮助成员了解设计、参与讨论，记录项目成果，减少沟通成本。重要的不是文档多丰富，而是这些文档有没有价值，你能不能及时通过这些文档得到想要的答案。所以你也可以对照一下你的项目中，现在的文档中，哪些是可以简化的，哪些地方是要增强的。

比如说概要设计/接口设计/详细设计是不是可以适当合并，减轻文档工作？PRD如果是不是够详细？会不会引起歧义不容易理解，要不要增加原型设计文档辅助？

陈珙 3

2019-04-18

在没有特殊要求的情况下，项目中更加倾向选择更为熟悉的技术，因为我们需要对项目的质量与交付时间负责，可以做到可控的。而新技术有着新的设计思想与强大的功能，同时也伴随着无法预知的”坑”。在后续产品迭代的时间里，有针对性的升级或者选择更换同类技术里更优的。

作者回复: 是的，正式项目应该尽可能选择熟悉的、成熟的技术👍

kirogiyi 3

2019-04-18

技术选型中，不考虑金三角理论的情况下，我会倾向于选择新酷的技术，这样有利于自己全面了解技术的发展趋势，更新自己的技术思维，感受技术革新带来的乐趣。另外，在维持项目稳定推进的状态下，尽量尝试使用新酷的技术，毕竟作为技术人总是淘汰于技术，更是成长于技术。

...

展开

作者回复: 👍谢谢补充分享。

关于极客时间架构，我是从池建强那篇文章中推测的，这个我确实不了解他们选型的目的和原因。

如果是我的话，现阶段也未必会选择微服务，微服务对运维要求更高、对架构要求更高，像极客时间现阶段的重点应该是功能和稳定性，直接上微服务，必然要花大量精力在架构的重构和运维上面。

不如先用熟悉的、稳定的技术，满足好现阶段业务增长，等到业务稳定，团队规模、用户量上来以后，再考虑拆分微服务不迟。

Y024
2019-04-21

2

Appfuse（一个 Web 开发基础平台）的作者 Matt Raible 曾总结了选择 Web 框架的 20 条标准：
http://static.raibledesigns.com/repository/presentations/Comparing_JVM_Web_Frar
同时，他也整理了一份表格，你可以根据自己的权重进行调整，产生自己的分析：
https://docs.google.com/spreadsheets/d/12l0sZNVSnwxcxs3CtcjeaCcI8rXjTrrldfT...
展开

作者回复: 非常有价值的分享！

我觉得在纠结的时候，遵循“经济适用原则”蛮好的，不会犯大错误，老老实实选熟悉的成熟稳重的把需求满足好才是王道。

"make it Work Make It Right Make It Fast"

Aaron Che...
2019-04-18

2

楼上说的 深以为然
展开

作者回复: 已回复楼上：）

刘晓林
2019-04-21

1

个人感悟:先明确自己的需求，对业务做领域建模，然后参考对应领域中主流的解决方案，这样即能控制尝新带来的风险，又不至于技术上掉队，导致代码一写出来就过时了。然后还要考虑时间、成本、范围的约束。
展开

作者回复: 谢谢总结分享

邢爱明
2019-04-20

1

项目团队的开发人员，基本都是从外包公司临时找的，水平参差不齐，稳定性差，因此技术选型更多的考虑技术的普及度和容易学习掌握，从这方面看基本不太可能选择比较小众、但在特定领域很高效的技术。
加上是企业内部管理的系统，数据量和用户数量可控，因此存在技术瓶颈的可能性很小，综合下来看就最好的选择就是最成熟和通用的技术，比如说选择java技术栈，web开发的...
展开

作者回复: 我觉得团队的技术提升和项目的技术选型要分开，不要总想着两个都兼顾，优先保证好项目稳定、低成本运行。

技术提升这种事，需要让一部分人先成长起来，然后带动其他人。

我自己工作之外会做一些业余项目，然后在这些项目中体验新的技术，体会其中优缺点，然后再逐步应用到工作的项目中，在传授给同事们。

我也鼓励其他同事这么做，去做一点自己的项目。但工作中的项目，我是很保守的。

小先生
2019-04-19

1

目标，调研，验证，决策。

要和利益相关人一起讨论，不要先入为主。

可以引入金三角理论来分析调研。...
展开

作者回复: 赞总结

Charles
2019-04-18

1

三四线城市，技术选型前期主要考虑：

当地市场什么人才比较充足，比如后端PHP人多，那就PHP，学习成本也低，几人团队协作起来也不是大问题，而且前期扩充人员也比较好招人，另外前期应该也不会在语言层面出现性能问题...

展开

作者回复: 我觉得你的选型思路在项目发展阶段，包括没有很大规模之前都是没有问题的。选最熟悉的、流行的往往也是风险比较低的。包括如果就一个PC站也不做SPA（单页应用），也没有必要前后端分离。还是看是不是能低成本满足好项目需求和业务发展。

有一点补充的，就是前端除了MVVM，像React的Flux和Redux的架构模式，也是一种很好的架构模式，但在非Rect/Vue的项目中应用不多。

---

alva_xu
2019-04-18

1

另外，对于开源技术方面，老师有没有什么经验来指导选型？

展开

作者回复: 开源技术选型，我的经验一般是这样的：
1. 先找朋友推荐，少走一点弯路
2. 没有推荐的话，就去网上搜索，找几个满足需求的备选。
3. 对比以下几个指标：
- 代码质量、有无测试
- 文档健全度
- 看Issue处理情况、最后更新时间（无人维护的项目后续恐怕有问题都没法解决）
- 看Star数量，通过Google和StackOverflow看使用情况
4. 自己按照说明试试看

---

alva_xu
2019-04-18

1

技术选型，确实就是项目决策。考虑的角度应该可以参考“09可行性研究"中的技术可行性部分。就是从人员、技术、风险三个角度来考虑。技术有大有小，比如传统框架和微服务框架的选择，就是一个大选择。需要选型的技术在项目和产品中的重要性决定了选型时的

风险偏好。团队对技术的熟悉程度，或者学习曲线，也是需要考虑的。另外，我觉得还可以用“天时、地利、人和”这三个角度来进行技术选型。...

展开

作者回复: 我认为在满足设计目标的前提下，大的原则还是在于项目约束，尤其是成本和时间，然后就是技术可行性和风险是不是可控，其他看团队风格，有的偏保守有的追新。

比如说我自己的原则：
1. 成熟的好过新酷的
2. 流行的好过小众的
3. 团队熟悉的好过陌生的
4. 简单的好过复杂的
5. 开源的好过商业的（有时候也视情况而定）

仅供参考

dancer
2019-04-18

1

首先确认当前技术选型要达成的主要目标，个人感觉可以分为提升开发效率、解决性能瓶颈、提升部署和运维效率、解耦强依赖的结构关系等。但是真的做选型时，往往还是会青睐自己喜欢和熟悉的技术，真的有点当局者迷的味道

展开

作者回复: “做选型时，往往还是会青睐自己喜欢和熟悉的技术”

完全能理解，所以需要配合一些流程规范来尽可能避免，这就是为什么需要有技术评审这样的环节，要调研要试点。

sotey
2019-04-18

1

我们是调研还没做完就直接决策了

展开

作者回复: 建议以后可以更慎重一点，这样可以避免因为决策不慎重导致的返工、技术债务等。

Felix

2019-04-20

现学现卖：走快的唯一方法是先走好

展开

作者回复: 欲速则不达

小先生

2019-04-19

定目标，

展开