54 | 业务的可支持性与持续运营

2019-11-05 许式伟

讲述：姚迪迈
时长 08:32　大小 7.82M

你好，我是七牛云许式伟。

保障业务的 7x24 小时不间断服务并不容易，所以我们花费了非常长的篇幅来探讨这件事情。我们需要系统性的、结构化的解决方案，它的涉及面非常广，需要基础架构、中间件、SRE 工作平台等多个层次、多个工种之间的紧密配合。

客户支持

但就算线上没有出问题，我们的用户仍然会遇到各种麻烦。所以大部分公司都会成立客户支持团队来服务客户。客户支持团队可能使用工单、电话或者即时通讯工具来服务客户。

对于客户支持部门，我们如何评估他们的业务价值？

很多公司会关注服务质量，简单说就是客户对某个会话的服务满意度。这当然也是重要的，但对整个客户支持部门来说，最核心的还不是这一点。

我们最核心的关注点，是如何减少客户服务的人工成本。

通常来说，客户支持团队会收到客户各式各样的问题反馈。这些反馈大体可以分这样几类：

使用姿势类；

报障类；

投诉与建议类。这个不是今天关注的内容，不再展开。

我们首先看 “使用姿势类” 。这又细分为两种情况：一种是完全不知道应该怎么用我们的产品，需要有一步步引导的向导或者示范的 DEMO。另一种是接入了我们的产品，但是发生了非预期的结果，于是客户迷茫了，需要寻求帮助。

怎么才能避免用户拿到我们的产品，完全摸不到北，不知道应该怎样才能使用起来的情况发生？在产品中植入必要的引导是非常必要而且重要的。产品帮助文档虽然应该有，但是我们应该坚信的一点是，绝大部分客户的问题应该依靠产品自身来解决，而不是依靠产品文档来解决。

要想做到更好的产品引导，需要我们代入到客户的使用场景中去，预想客户各种可能遇到的问题是什么。

当然，在逐步迭代的场景中，历史的用户行为分析也能够指导我们找到需要改善的关键问题是什么。

客户接入了，但是发生了非预期的结果。在很多时候，出错的信息往往是很难理解的。这时候客户就被迫到产品文档网站或者搜索引擎去搜索出错对应的解决方案。如果迟迟无法得到解决，客户就会怀疑我们的产品，转向其他的替换方案。

所以错误信息的呈现，也是需要非常讲究的。我们非常有必要将错误信息的表达变得更加贴近用户的语言。甚至，我们在错误提示信息中，给出我们的建议，或者建议文档的链接。

还有一些错误的使用姿势，可能并不会在存在误用的现场直接表现为错误，而是在其他的某个场景下才发生错误。

这方面一个很好的例子是 Go 语言的 map，也就是大家通常理解的哈希表。Go 语言的 map 并不是多线程安全（准确地说，在 Go 语言里面其实是多 goroutine 安全）的，如果在多个 goroutine 中共享这个 map 但是却忘记了加锁，这个时候结果就是不可预期的。

按大部分人常规的逻辑，这样的误用在 Go 的文档中给予必要的提醒就好了。从职责来说，用户误用那是用户自身的问题。

但是 Go 语言团队显然不是这么看的。它特意在代码逻辑中加入了这种使用姿势上误用的检测，检测到错误后直接抛出异常，让程序停止运行。在异常信息的提示中，它告诉你，你的代码存在了什么样的问题。

这让用户更快地找到错误的根因，并且及时去修复错误。

这个细节让我感受颇多。产品被开发出来，对于很多研发人员的认知来说，是个结束。但是从业务的持续运营角度来说，这只是开始。

持续运营中产生的各类成本，无论是 SRE 做服务保障的成本，还是客户支持服务客户的成本，还是客户为了寻找问题根源而花费的时间成本，都需要我们去认真思考怎么进行大幅度的优化。

我们再来看下 “报障类”，也就是用户认为我们服务出问题了。当然实际问题不一定真是我们的服务有问题，也有可能是用户自身的环境，比如 Wi-Fi 或者本地运营商的问题。但无论原因如何我们都需要给客户一个确切的问题说明和建议的解决方案。

为了应对这种场景，我们就需要有意识地收集用户请求的整个调用链。这个机制和我们前面介绍的 “52 | 故障排查与根因分析” 一讲中介绍的 Tracing 机制类似，都会依赖 request id 这样的东西作为线索。只不过，为了有更加完整的调用链，这个过程不只是在服务端，同时还需要应用到客户端。

有了 Tracing 机制，客户端基本上就可以做到一键报障。在报障时客户端会携带自己的 IP 和 Tracing 日志。这样客户支持人员就可以通过客户 IP 和 request id 知道这个客户最近

都发生了什么事情。

但某种程度来说，一键报障也是繁琐的。所以很多应用程序会让你签署一份用户体验改进协议，在获得你的同意后，程序就可以把 Tracing 日志主动同步给服务器。这样我们的客户支持团队就可以不是针对某个具体的客户，而是针对相对全局性的问题来进行改进。

当然，这也意味着我们需要建立合适的数据运营的体系。通过这个体系，我们可以迅速找到用户最经常遇到的问题是什么，并持续加以改进。七牛云的 Pandora 日志平台可以协助做好这方面的改进工作的开展。

有一些改进我们来不及系统化进行改进，但是我们已经形成了一些最佳实践，那么对于 VIP 类客户我们可以先主动进行最佳实践的推广。

用户报障也有可能是我们的业务真实发生了故障。这时客户报障通常会有一个突然的爆发式增长，并且往往容易让我们的客户支持团队应接不暇。

当线上发生故障的时候，什么时候对外宣布事故，什么时候主动通知客户？

首先，先宣布事故发生，随后找到一个简单解决方案，然后宣布事故结束，要比在问题已经持续几个小时之后才想起告知客户要更好。

所以针对事故，我们应当设立一个明确的宣布条件。比如，如果下面任何一条满足条件，这次事故应该被及时宣布：

是否需要引入 SRE 之外的团队来一起处理问题？

在处理了一小时后，这个问题是否依然没有得到解决？

这次事故是否正在大范围地影响了最终用户？

如果平时不经常使用，事故流程管理的可靠性萎缩得很快。怎么才能使工程师不忘记他们的流程管理的技能呢？难道一定要制造更多事故吗？

一个可能的思路是，经常针对之前发生过的灾难进行角色扮演式的演习，比如演习另外一个地区处理过的问题，以更好地熟悉事故流程管理体系。

当然，事故流程管理框架其实也往往适用于其他的跨团队的常规运维变更过程。如果我们经常使用相同的流程管理方式来处理线上的变更请求，那么在事故来临时，就可以很好地利用这些流程管理手段来管理它。

BOSS 系统

客户使用上的困扰、排错、事故告知、投诉与建议，这些是客户支持所面临的常规工作。但实际上我们在服务客户过程中，往往还有更常规的业务支持工作要做，比如客户的业务开通、财务与发票、业务管理等。这类系统我们往往把它叫 BOSS 系统。

BOSS 系统面向的用户是企业的内部员工，为员工能够正常开展业务服务。

什么样的事务，应该被加入到 BOSS 系统？一个基本的策略是越高频执行的业务动作，越应该被固化到系统中。

业务过程固化到系统会有很多好处。

首先，最常规的，是业务效率提升。员工执行业务有更高的便捷性。

其次，安全风险管理。有一些业务过程可能会有潜在的风险需要防范，通过固化业务过程，进行必要的风险检查，可以避免掉最常见的已知风险。

最后，业务过程进一步被数字化，业务行为被记录，这为系统性地进行业务优化提供了可能。

除了客户支持系统、业务 BOSS 系统之外，还有一类极其重要的业务运营需求，就是客户增长的运营了。但客户增长运营是一个复杂的话题，我们这里就不再展开。

结语

今天我们聊的主要是一个视野的问题。除了产品的功能外，实际上为了产品能够更好地服务好客户，我们需要关注售前、售后支持能力的构建。

为了更好地支持客户，以及进行后续用户体验的改善，我们往往需要将用户行为记录写到日志系统中，以便于进一步地分析和挖掘。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。下一讲我们将聊聊“云计算、容器革命与服务端的未来”。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

上一篇　53 | 过载保护与容量规划

下一篇　55 | 云计算、容器革命与服务端的未来

精选留言 (2)

写留言

Aaron Cheung
2019-11-05

七牛云的 Pandora 日志平台 老师推荐了多次 值得试试👍

1

雷霹雳的爸爸
2019-11-05

这还真是挺难的一个问题，一个是如许大所说的视野问题，另一个在于组织文化方面的水平约束了能给予技术团队从多大程度上去整合整个业务价值链条来为客户提供能够更好的支持，以及为其自身做出必要的信息收集和技术演进支撑，这不是官话，作为一个传统服务业数字化转型过程艰难爬的技术支撑小组的领班人，还是很多切肤之痛的

展开