科学的故障管理

-鼓励改进，还是处罚错误？

赵成@蘑菇街

关于我

- 11年工作经历，电信及互联网软件研发&运维
  - 2008.5-2015.1，华为
  - 2015.1-至今，蘑菇街
- 擅长领域：运维、SRE、DevOps、云计算、技术管理
- 个人荣誉：
  - SRECon19 Asia/Pacific Speaker
  - 腾讯云TVP，最具价值专家
  - 专栏作家,《进化：运维技术变革与实践探索》作者
- 当前，专注于云计算、SRE和技术运营方向

目录

为什么要科学的故障管理

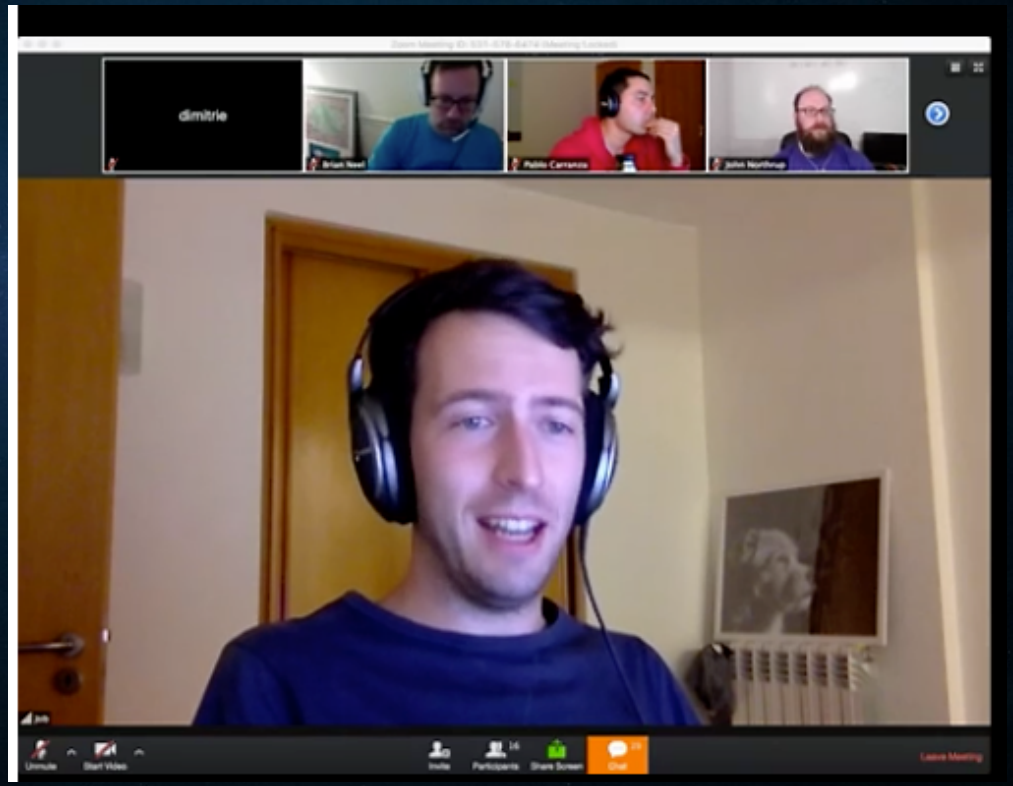
故障管理实践-定级、定性、定责

故障管理与SLO

为什么要“科学”的故障管理？

Blameless Culture

Gitlab全球直播故障恢复



如何科学地管理故障？

—故障定级、定性、定责

纠结的问题

- 问题到底严不严重？有多严重？
- 责任谁来承担？ 锅谁来背？
- 造成恶劣影响，应该怎么处罚？

故障定级-影响程度（P0-P4）

- 影响范围
  - 用户影响，持续时长，如注册、登录、IM通信
- 影响收入
  - 收入降低，GMV、支付金额
- 造成资损
  - 向商家、CP、SP赔偿损失
- 造成PR事件
  - 公司口碑、声誉受损

故障定级/升级示例

产品线	功能大类	问题描述	影响面/定级	P4	P3	P2	P1
交易	交易主流程 （按订单量下降算）	造成下单量下跌的按此计算（ipad版本不适用此升级时间）	30-100%				P1
			10-30%			P2	0.5H
			0-10%		P3	1H	2H
	交易核心功能	购物车：加入购物车、购物车列表、去结算功能不可用 下单：确定下单不可用，下单主流程无法进行；地址信息出不来；运费计算错误； 支付：去支付不可用，支付回调不可用	60-100%				P1
			10-60%			P2	1H
			0-10%		P3	2H	4H
	次核心功能	购物车：修改，凑单 订单操作：发货、确认收货、结算给商家、发起退货退款，退货退款列表，卖家同意/拒绝，买家发货，卖家确认收货 订单操作：取消订单、删除订单、发起维权、修改订单（地址、价格等）	60-100%			P2	2H
			10-60%		P3	4H	8H
			0-10%	P4	4H	8H	
	非核心功能	购物车：计数 订单列表：计数，退货退款，订单搜索不到。	60-100%		P3	4H	
			10-60%	P4	4H		
			0-10%	P4	8H		

故障定性-有效分类

- 代码质量
- 测试质量
- 流程规范
- 硬件设备
- 变更执行
- 第三方
- 产品逻辑
- 预案演练
- 。。。。。

故障定责-判定原则

- 高压线原则
- 上下游原则
- 健壮性原则
- 分段判定原则
- Case By Case原则（自由裁量）

谁来主持公道？

科学管理-“法官”判定

中立

权威

规则制定

自由裁量

普法宣传

主持公道

定级

定性

定责

要不要“处罚”？

定责 ≠ 处罚 ≠ 钱

要不要处罚？

- 慎用处罚，负作用大
- 要有标准，有理有据
- 高压线、重复错误、低级失误，要罚
- 长周期，全局审视
- 管理者要承担更大责任

故障管理与SLO

如何度量—SLO稳定性度量体系

SLI

- 度量内容：
  能够衡量服务质量的指标，如可用性，RT
- 度量范围：
  指标合理范围，如RT<200ms，http返回码为非5xx

SLO

- 全站访问请求RT<500ms的比例不低于95%
- 全站访问请求非5xx比例>99.95%

SLA

- 涉及双方商务合约，违反将触发处罚条款
- SLA = SLO + 后果

SLIs and SLOs

Category	SLI	SLO
API		
Availability	The proportion of successful requests, as measured from the load balancer metrics. Any HTTP status other than 500–599 is considered successful. `count of "api" http_requests which do not have a 5XX status code divided by count of all "api" http_requests`	97% success

Latency	The proportion of sufficiently fast requests, as measured from the load balancer metrics.
	"Sufficiently fast" is defined as < 400 ms, or < 850 ms.
	`count of "api" http_requests with a duration less than or equal to "0.4" seconds divided by count of all "api" http_requests`
	
	`count of "api" http_requests with a duration less than or equal to "0.85" seconds divided by count of all "api" http_requests`
		90% of requests < 400 ms
		99% of requests < 850 ms

SLI选择依据—VALET法则

- Volume-容量
  - ➢服务承诺的最大容量是多少？（QPS、流量、连接数、吞吐等）
- Availablity-可用性
  - 服务是否正常？（2xx的返回码的请求比例）
- Latency-时延
  - 响应是否足够快？（rt是否在规定范围内？任务执行时长是否合理？）
- Errors-错误
  - 错误率有多少？（5xx占比）
- Tickets-人工介入
  - 是否需要人工介入？（任务重跑？数据修复）

蘑菇街业务SLO大盘

2018-11-22 17:54:47 ~2018-11-29 17:54:47 | trade_buy_web | Api.UnifiedCreateService

Api.UnifiedCreateService

	VOLUME		AVAILABILITY	LATENCY		ERRORS
	Request Per Minute	Peak TPS	Overall Availability	90th Percentile(ms)	95th Percentile(ms)	Error(%)
SLOs	3000	100	99.96%	50.00	80.00	1.00
2018-11-28	1369	40	98.01%	30.21	48.35	0.07
2018-11-27	926	55	99.99%	32.27	44.41	0.02
2018-11-27	936	51	99.10%	54.24	73.89	0.12
2018-11-28	1130	36	99.98%	35.54	49.2	0.03
2018-11-28	1213	44	97.12%	61.21	81.48	2.87

CDN SLO制定

可用性
成功率

日常情况：96.97% ~ 99.67%

用户体验
请求RT

日常情况：
90% 请求RT：<150ms
95% 请求RT：< 160ms

经济性
命中率

日常情况：
命中率：76.78% ~ 94.99%
9x%占比：68%
8x%占比：30%
<80%占比：2%

综上所述，不可用时长 = (可用性<98% || 90% Rt>200ms || 命中率<80%) && 持续2min

在一月内，开始计算月度稳定性可用率时长，全月多次触发以上条件会累加计算。
即：*月度可用率 = (总时长– 不可用时长)/总时长*

SLO体系延伸

- Error Buget-错误预算
- Policy-预算策略
- 宽松和收紧SLO
- 基于SLO的告警

故障复盘—黄金三问

- 故障根因，究竟是什么？
- 我们应该怎么做，才能更快地恢复业务？
- 我们应该怎么做，才能避免再次出现类似问题？
- One more thing，我们还可以做些什么？

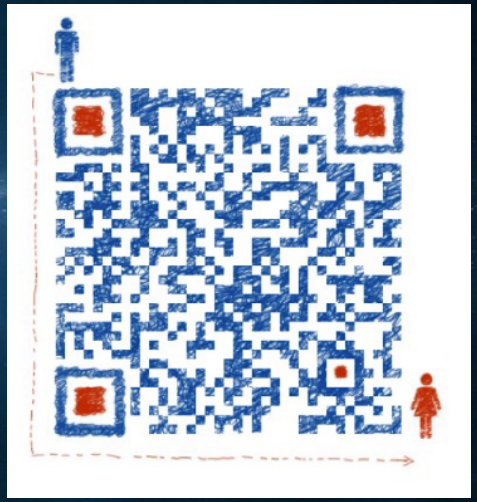
如何辩证的对待故障？

- 理解一个系统应该如何工作并不能使人成为专家，只能靠调查系统为何不能正常工作才行
- 系统故障是常态，系统正常是特例
- 故障永远是表象，背后技术和管理问题才是本质
- 从技术角度考虑解决方案，尽量避免流程和制度约束
- 管理者应该多反思

实践中，故障是暴露问题最充分的渠道

鼓励改进，而不是处罚错误。

Thanks



继续交流