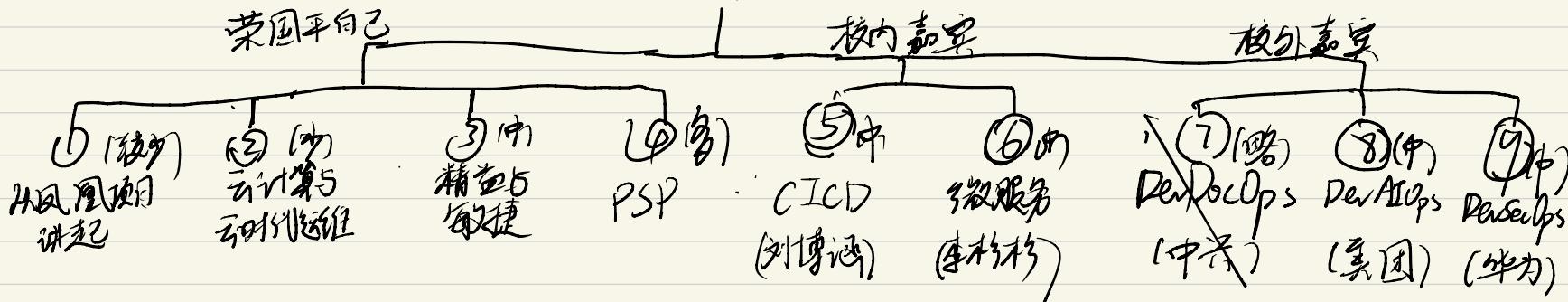


DevOps 2022



十华为 凤凰商城

MOOC 腾一眼

① 凤凰项目 措施

- 变更可视化
- 解放资源约束
- 安全审计
- 远维自动化

业务项目

- IT 内部项目
- 变更
- 计划外工作

精益

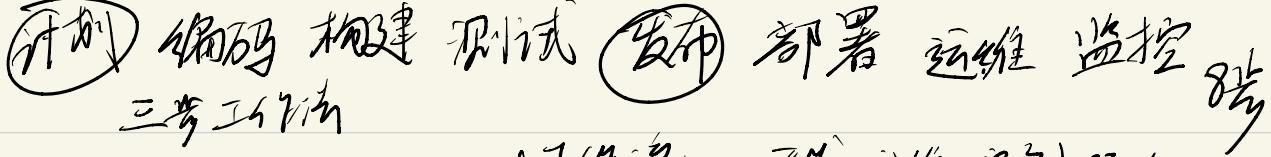
(合) 混合流
看板 (看板)

持续集成
Integration

持续交付
Delivery

降低批量规模
减少半成品
缩短并增强反馈回路
(交付时间)
(部署频率)

持续部署
Deployment



① 打通价值流

Rev → Ops

工作流
增量交付
反馈环

从 Dev -> 运维 -> 客户
CI / CD
持续环境
限 WIP
精益组织
看板

② 强调反馈循环

Rev → Ops

避免问题两次发回
以源来保证质量

适时停止并持续改进
自动化测试监控与优化

③ 持续交付与学习驱动

Rev → Ops

敢于 & 敢而为
25% 投入 NFR

价值观 原则 方法 实践 工具

通过自动化积累经验并容纳变更的过程，使得构建、测试、发布执行能更快速地响应变化与需求

部署频率 变更时间 平均恢复时间 MTTR 变更失败率

② 云计算与云时代运维

定义：1.0版 SaaS PaaS IaaS

云计算 → 边缘计算 → 网计算
云中心 边缘 + 网上边缘
层级增强

CMMI-SVC

1-5 评级
开发运维一体化
成熟度模型国家标准

ITIL

Information
Technology
Infrastructure
Library

战略设计、转换、
运营、改进

④ 互联网式运维

快速迭代
快速发布

敏捷发布
安全

③ 云计算下的运维

自身平台

编译器

④ 大规模下的运维

标准

自动

⑤ 提前运维

监控运维

预防设计

ISO 20000

ITSS

P D C A
策 审 檢 訂
劃 實 施 查 置

成熟度和综合
配套的信息
服务标准库

运维主要指软件系统测试交付后的发布和管理工作，其核心目标是将交付的业务软件和硬件基础设施进行高效合理的整合，转换为可持续提供高质量服务的产品，同时最大限度降低服务运行成本，保障服务运行的安全。

③ 从传统到敏捷

反摩尔定律 → 更早更快交付价值
灵活地响应变化

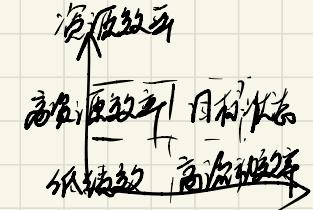
SCRUM, XP, DSDM

- 敏捷宣言
- ① 个体和互动胜过流程和工具
 - ② 可工作的软件胜过详尽的文档
 - ③ 客户合作胜过合同谈判
 - ④ 响应变化胜过遵循计划

精益原则

全局优化、消灭浪费、品质为先、不断学习、尽快交付、从参与、不断提高

从资源效率到流动效率



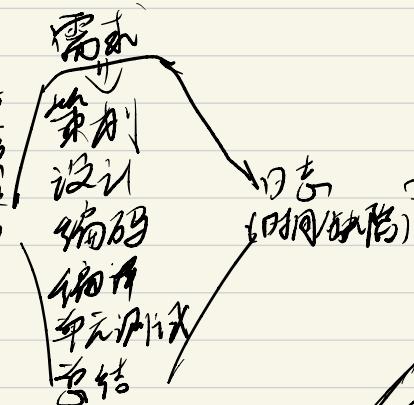
看板、控制库存、加速流动、灵活响应、提高质量

- 可视化的板 /
信号卡
- ① 可视化价值流动
 - ② 立式化流程规则
 - ③ 控制在制品数量
 - ④ 管理价值流动
 - ⑤ 反馈与持续改进

④ 个体软件过程 - 整合

- ① 个人级管理实践 过程估算与计划
- ② 反馈和迭代循环
- ③ 理解和改进
- ④ 工具评估的移与规范
- ⑤ 客观决策的数据

过程操作指南

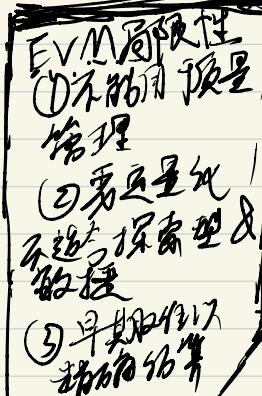


水桶原理

PSP0
执行进度
时间缺陷
缺陷类型标准

PSP0.1
缺陷登记
进度沟通记录
编码执行规范

+ 日志 &
规范



(3) 早期时间轴
缺陷统计

PSP2
代码审查
设计审查

PSP1
PROBE估算
缺陷报告

PSP2.1
设计模板
复用模板

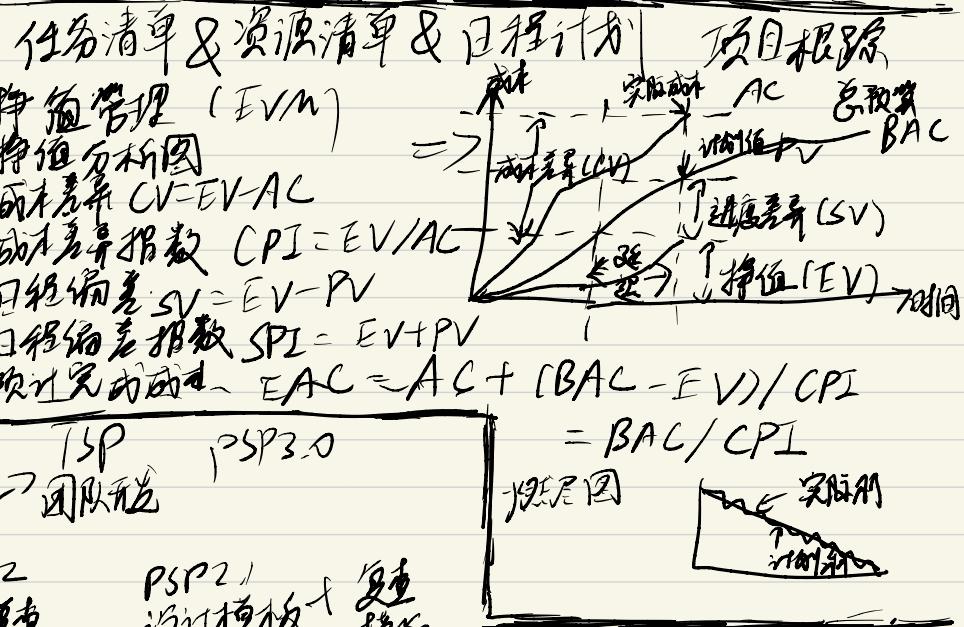
PSP1.1
缺陷计划
日志+计划

+ 日志

TSP
团队形成

PSP3.0

估算方法
LOC FP 不同早期
缺陷报告 功能点
日志管理见上方
时间轴: 开始 结束 中断
缺陷日志
发现日期 进入阶段 消除阶段
清除时间 关联缺陷 处理描述



(5) 个体软件过程 PROBE (PROxy Based Estimation)

概要设计

代理识别与代理规模
估算并调整规模
计算预测区间

→
调整资源
计算预测区间

① 确定不同大小的组件单个成本
VS S M L VL

然后个数 × 尺寸得结果 最后结性圆周
出预测圆周曲线、置信区间与方差

② 从历史数据得到构件大小
简单分布 正态分布 对数正态分布
指数分布 均匀分布 大部分小概率
分布 S 方差 S 埋入缺陷

VS M S = M - 3
最大 VL VS = M - 3
选择中值 M M = M
VS S M 的值为 S L = M + 10
VL S M 的值为 L VL = M + 20 → VS 为负数吗?

③ 从预测大小到实际大小

$$\beta_{size} = \frac{\left(\sum_{i=1}^n x_i y_i \right) - (n \bar{x} \bar{y})}{\left(\sum_{i=1}^n x_i^2 \right) - (n \bar{x}^2)}$$

$$E_{size} = \bar{y} - \beta_{size} \bar{x}$$

最好方法是
上加征高/对数

Range = (这个给了公式也算不了)

$$\text{方差: } \sigma^2 = \frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y})^2$$

③ 局限与度量

④ 有足够的历史数据, 质量不高

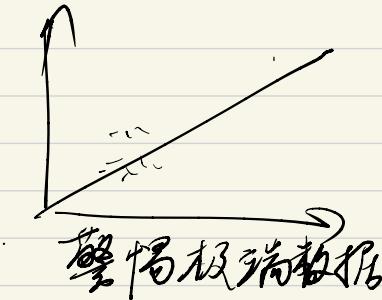
相关性 > 0.7

$$r_{x,y} = \frac{n \left(\sum_{i=1}^n x_i y_i \right) - \left(\sum_{i=1}^n x_i \right) \left(\sum_{i=1}^n y_i \right)}{\sqrt{\left[n \left(\sum_{i=1}^n x_i^2 \right) - \left(\sum_{i=1}^n x_i \right)^2 \right] \left[n \left(\sum_{i=1}^n y_i^2 \right) - \left(\sum_{i=1}^n y_i \right)^2 \right]}}$$

显著性 SC0.05

$$\sqrt{\left[n \left(\sum_{i=1}^n x_i^2 \right) - \left(\sum_{i=1}^n x_i \right)^2 \right] \left[n \left(\sum_{i=1}^n y_i^2 \right) - \left(\sum_{i=1}^n y_i \right)^2 \right]}$$

	个体软件过程	数据需求 $r \geq 0.7$	计算方法 如直角坐标
A	数据需求 3组以上代理规模 + 实际规模	$s \leq 0.05$	
B	3组以上代理规模 + 实际规模	$\beta_0 \leq -0.25y$ $0.5 \leq \beta_1 \leq 2$	
C	有历史数据	无	预测的调整
D	没历史数据	无	猜



估算准确性

估计划：设备技能

条件外事件 变更实现错误

详尽计划

-识别公认的方法

-基于项目的经验估算

-对底下的进行预测判断

整合估算

整合一个项目的多次估算

- ①累积各部分估算
- ②进行次级性回归计算
- ③计算一个预测区间

整合多个开发人员

①单独线性回归预测

②将计划规模或时间相加

③将个人范围平方和相加

再对其中平均数是预测区间

估算精度 $\pm 50\%$ 1000 h 工时 ($500h \sim 1500h$)

$\Rightarrow 25$ 个部分 $\pm 50\%$ 平均每个 $40h$ 每个方差 $20^2 = 400$
方差和 $10000 \Rightarrow$ 标准差 100 估算范围 $900 \sim 1100 h$

⑤ CI/CD 持续集成

是 DevOps 的重要实践
是实现高效的、可靠的
价值闭环的基础

减少风险
减少重复过程
增强项目的可见性

配置管理是
自动化是基础
反馈是目的

物理前提

依赖

配置项

环境

编译

集成

测试

部署

触发 -> 审核

SCCS

- ① 动态中滚动部署
- ② 得到干净环境
- ③ 取得软件资产生成物的标签
- ④ 极发布、生产 -> 测试
- ⑤ 构建反馈报告
- ⑥ 和上游系统集成

Travis CI + GitHub

私密快、运维稳

单元集成 系统功能

速度↑	一月多次
可见性↓	每日一次
缺陷 服务者 → 用户	每周一次
耗时↑ 频率↓	每版一次

测试中的权衡 (成本)
以及触发机制

审查

测试

工具 vs 人

便宜	工具规则
常见	(工具局限性)
彻底	(灵活)

风格	缺陷
漏洞	检测
设计	(增强)
静态	动态

开发环境 预发布环境 生产环境

持续集成	持续交付	持续部署
√	√	√
持续交付	持续部署	√

DCI/CD 开源 变革成功

构建 打包部署 启动 运行

现实阻碍 ③老系统的迁徙

④首次集成过耗时

持续数据驱动过程

GitLab GitLab Runner ...

私密快、运维稳

增加抽象人共性

减少具体人个体

④ 微服务

什么是软件架构设计模式

- 组织类
 - ④ 与环境 + 设计深灰原理 - IEEE
 - ⑤ 外部可见属性 - Len Bass
- 设计类, 模式 相互看

快速 颗粒度 可靠

单体 \rightarrow 分布式 \rightarrow SOA \rightarrow 微服务 \Rightarrow 2011
2016

(组件
部署)

简单

弱测试

不割裂 I/O

不可拆分

\downarrow SOA
模块化组合
强依赖
缺乏细颗粒度
不关心细节
ESB

解耦

契约

封装

相互通用 路由转换

自治

重用

组合

无状态

分布式管理

多语言
国际部署能力
构建共享

云大特点

① 通过服务组件化

② 围绕业务能力组织

③ 内聚 解耦

④ 去中心化

⑤ 基础设施自动

⑥ 服务设计 \leftarrow 高可用性
 \downarrow 变更与演化

技术债、过高复杂性

组件扩展、组件耦合

日依赖

对团队的资源 跨功能

提瓶颈 等待？ 一致面对 远程团队

重构「根本 新办法，旧人旧办法」

提取 直接动手拆

脆弱泛用设计

\Rightarrow API 网关 增强器（弹性设计）

① 通信协议

② 中国旅行社

③ 弱测试

④ 没有统一的服务治理

⑤ 复杂性↑

坏：① 内聚低耦合 ② 良好互通性

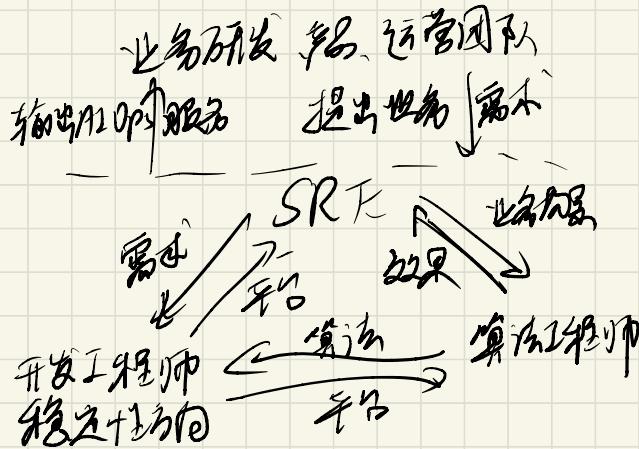
③ 语义理解困难

⑦ Dev Doc Ops (四名)

内容敏捷 内容管理 内容交付 编写 审核 快速 准确 口碑

⑧ AI Ops

Algorithm IT Operations \rightarrow AI for IT Operations



缺失值填充
标准化
降维处理
模型训练

如果异常样本太多，那就人造
自动 + 人工筛选

特征物品优化
① 极端期 + 窗口

② 节假日 + 前后对比、节假日经
济影响
③ 楼层分布，人群 遵循 (④ 极端波动)

故障发现 \rightarrow 故障确认 \rightarrow 故障定位 \rightarrow 故障恢复

故障/平稳灰度
自动检测
基线识别
(日历分析)

统一告警
降噪与聚合
(规则挖掘)

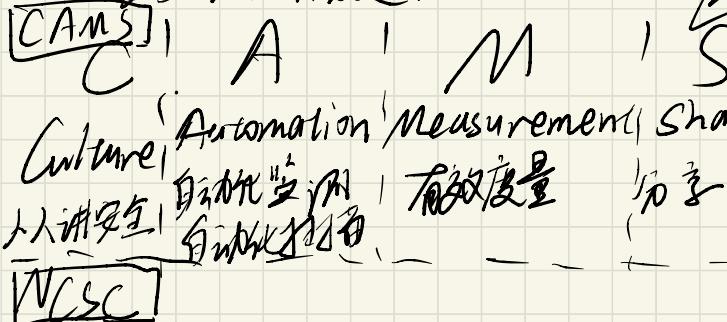
智能定位
实际模拟
知识图谱
演绎止损

服务降级
故障自愈
常态化演练

量子化设计 模型编排
离线训练 实时检测

⑨ DevSecOps

安全贯穿开发运维全过程



- ① 安全开发是每个人的责任
- ② 保持安全知识体系化
- ③ 支付整洁和可维护的代码
- ④ 保障开发环境安全
- ⑤ 保护代码库
- ⑥ 保障构建和部署流水线安全
- ⑦ 持续训练安全性
- ⑧ 为安全缺陷制定计划

CAMS & NCSC

操作实践

ITIL
构建
验证
预防
发布
预防
检测
响应
预防
调整

培训 负责人 责任/工具共享
循环 事件/变更管理

例：① 提交问题 ② 变更 ③ 流程组织

目标：① 建立敏捷与安全
② 构建 DevOps 专家
③ Dev SecOps 安全流程
④ 强化自动化工具
⑤ 起步成本

人、流程、工具、成本