

从经典软工到开源开发

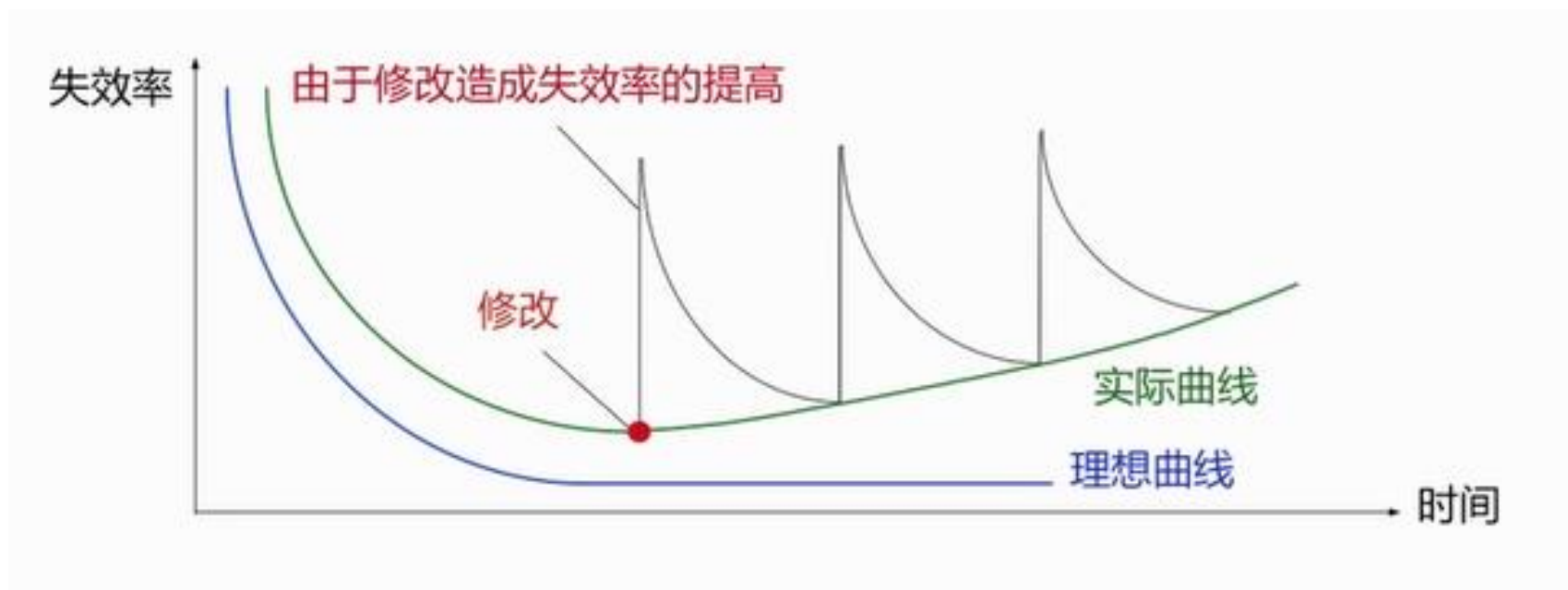
周明辉

zhmh@pku.edu.cn

北京大学

软件行业的演变：从制造产业到服务产业

- 软件工程伊始，软件开发被认为是制造业，但是，难以用销售价值决定软件价值，因为75%的资源都消耗在软件维护上。
- 因为软件维护的开销巨大，软件产业逐渐被认为是服务产业。



经典软件工程

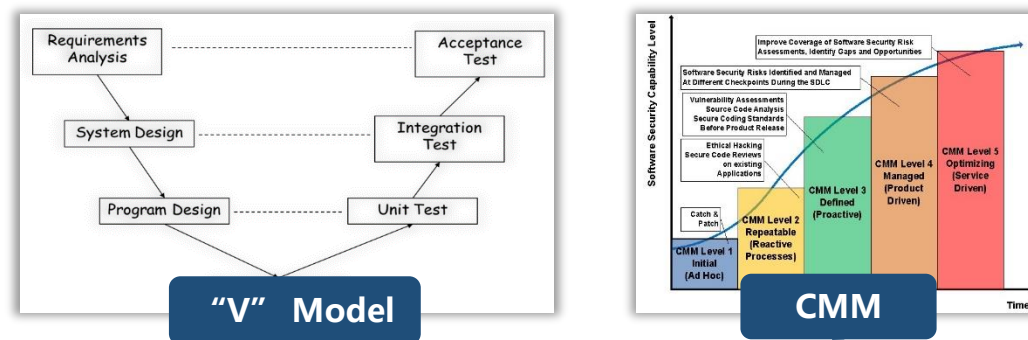
制造产业思想

“制造产业”的软件工程



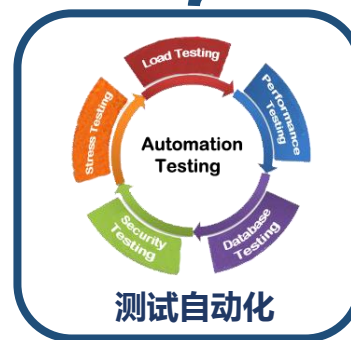
NATO Software Engineering Conference 1968

向经典的工业生产学习
软件工程三要素：方法、工具、过程



研究过程管理方法：组织性

研究软件生产工具：自动化



测试自动化



编程与修复自动化



验证自动化

工程化开发方法：自上而下、逐步求精

元素		工程化开发方法 自上而下，逐步求精
理念	需求	有明确用户和明确一致用户需求描述，这是软件开发的前提
	质量	源代码满足需求描述的程度
	效率	开发满足需求软件的时间和成本
方法	过程模型	瀑布模型、敏捷模型、能力成熟度模型
	支持工具	软件产品线工具集
	计算环境	面向计算机环境

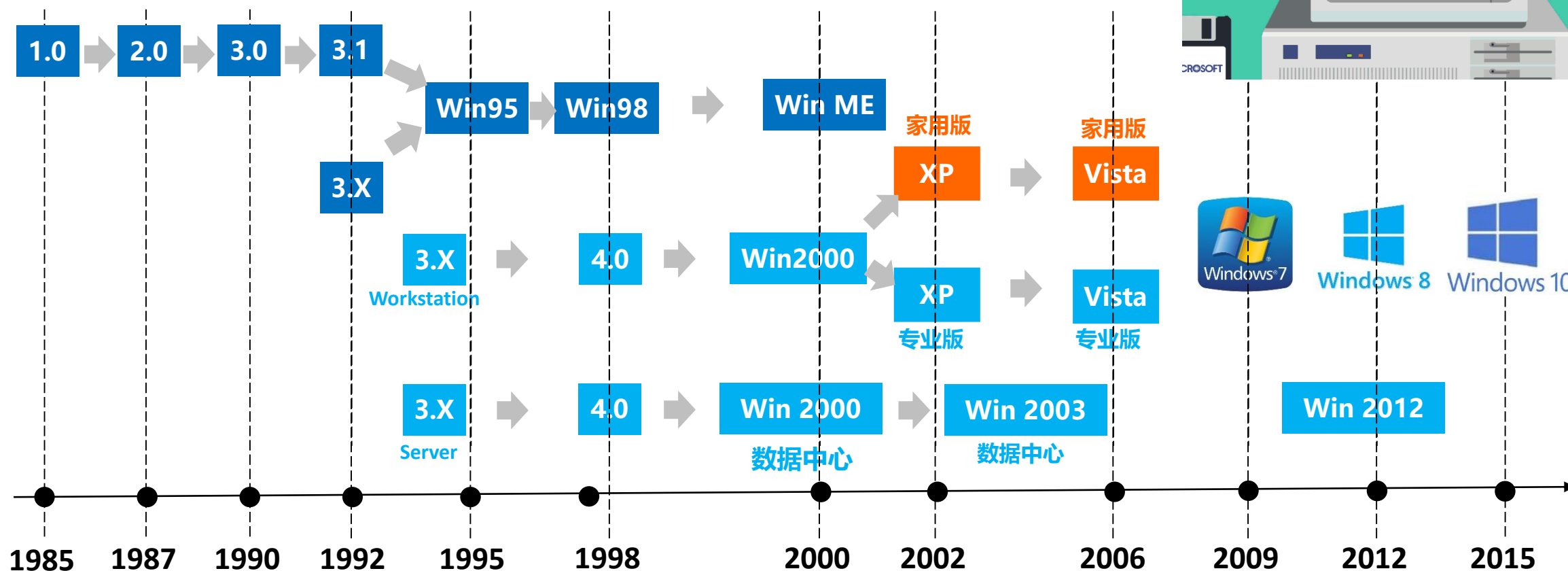
- 用户需求是确定的
- 可被理解的
- 可被表述的
- 可被线性分解还原的
- 严格定义的流程和时间

工程化的经典：Windows 演化史

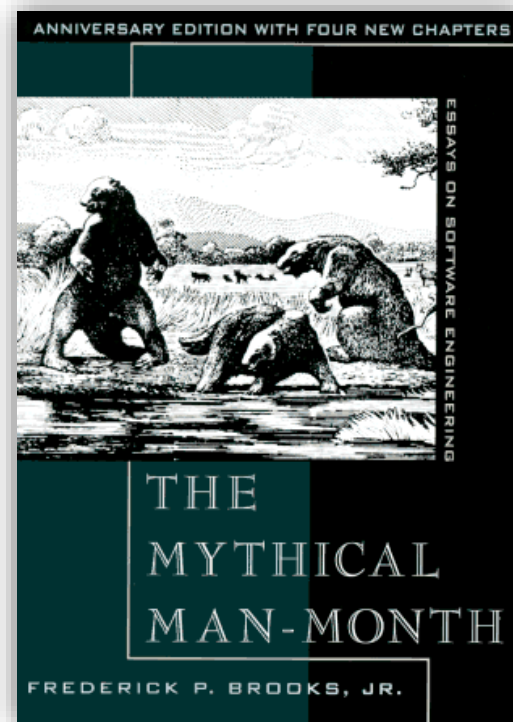
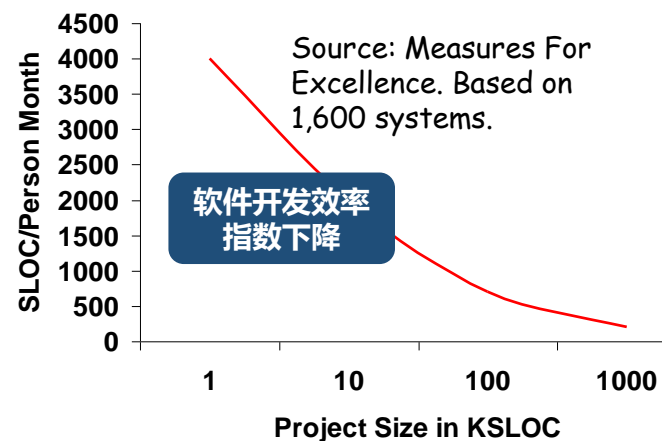
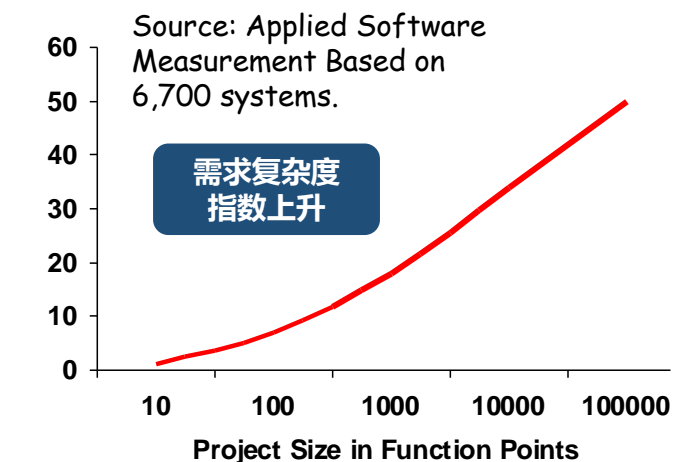
严格控制软件出现分叉版本

MS-DOS版本

NT版本



工程化开发方法的协同瓶颈



“No Silver Bullet”

软件开发管理如何突破群体协同生产的天花板？

工程化开发方法的领域瓶颈

复杂软件系统

人与系统的边界模糊了

分散控制

固有的冲突、不可知、
多样需求

失效是常态

持续演化和部署

异构的、不一致的和变
化的元素

经典方法的前提

人仅仅是系统的使用者，人的群体行为不为系统所关注，系统不涉及社会性的交互

求解过程是集中统一控制的、**自上而下的分解**过程

需求是能够被提前获知的，所有的冲突必须被解决，其变化是缓慢的，决策是稳定的

缺陷能够被剔除，失效应该极少产生

系统的改进是阶段性的

变化的结果是可以被充分预见的，部署配置信息是精确的，可被严格控制的，构件与人员是完全同质的

开源开发

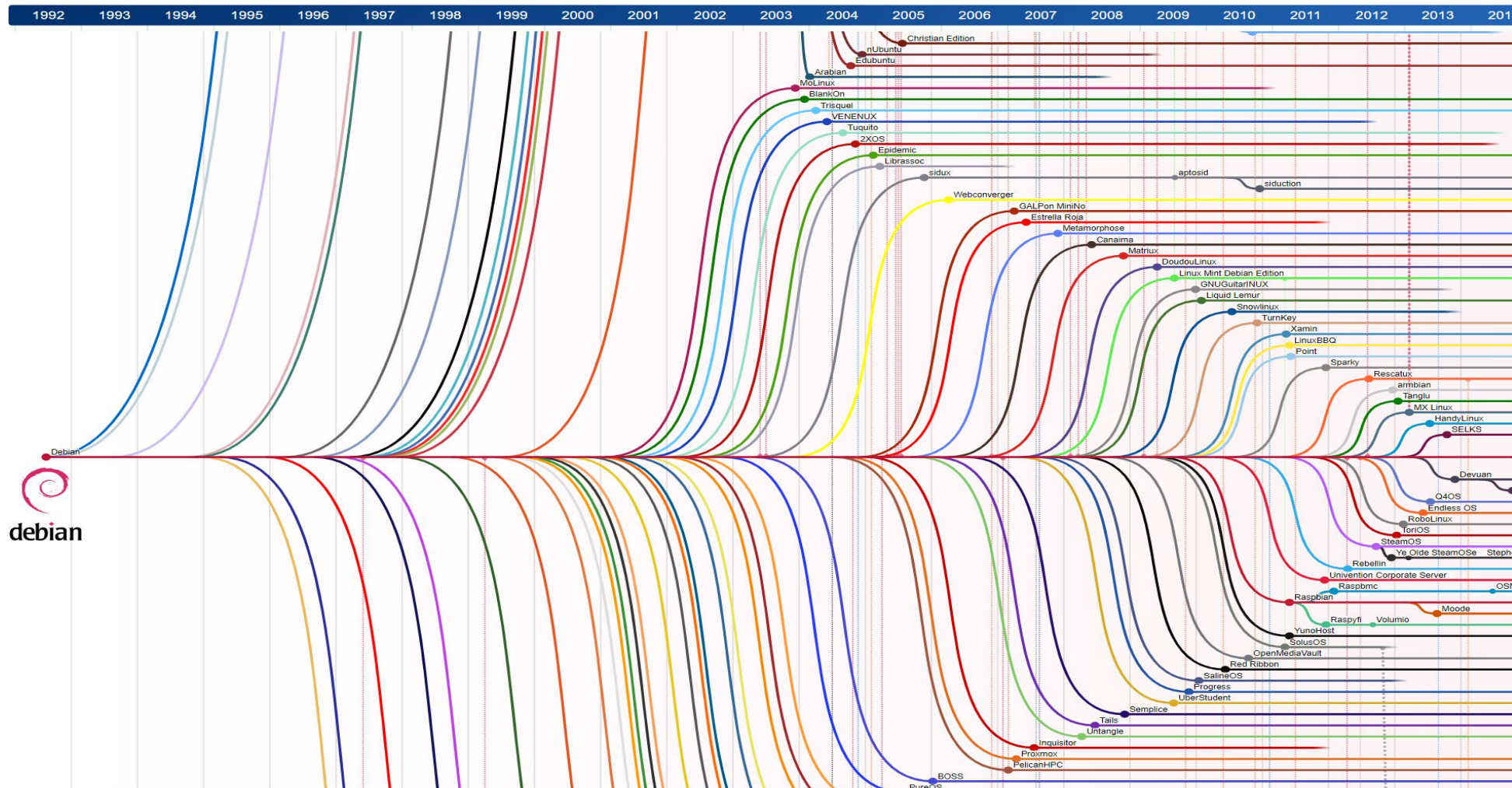
Linux的成功：偶然与必然



Linus Torvalds
(1969-)

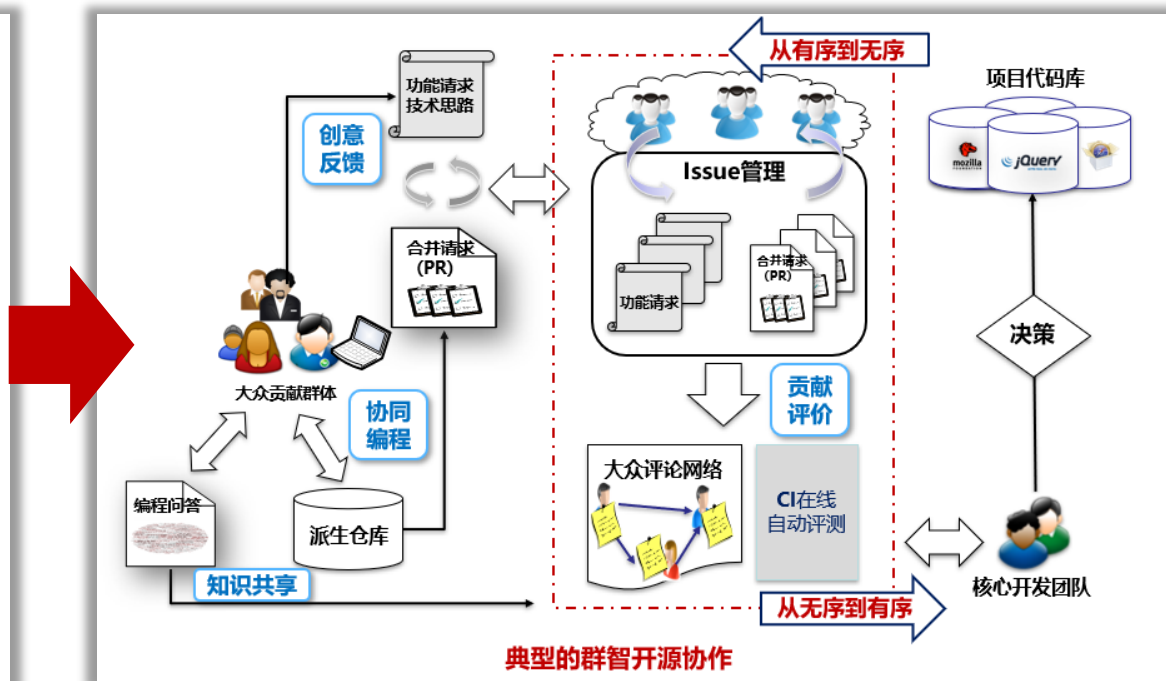
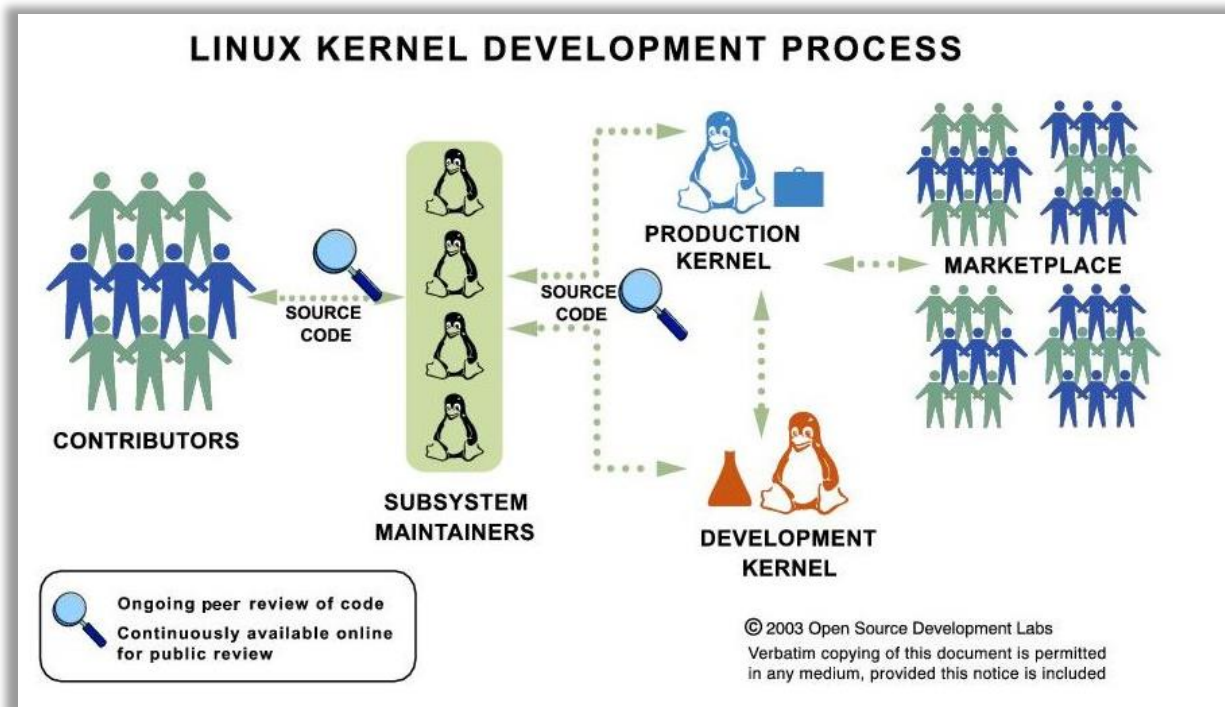


1991年8月发起



Linux的成功：偶然与必然

- 从被组织的大规模协同生产活动到自组织的超大规模协同开发活动
- 第一个大规模松散自由组织的成功案例：超过1900个公司21000个开发者，贡献代码超过2700万行



开源模式的本质特征

- 用户需求驱动： scratch my own itch
- 开放透明的同行评审： 同行评审的范本
- 人类最佳协作模式： 精英自由主义者们的一次成功的协作行为艺术的展现

大教堂与集市

THE CATHEDRAL & THE BAZAAR

WISDOMS ON LINUX AND OPEN SOURCE BY RICHARD STALLMAN

开源参与者技能 – 文化和技术



“Publicly making fun of people is half the fun of open source programming.

本质：公开透明同行评审

In fact the main reason to eschew programming in closed environments is that you can't embarrass people in public.”

– Linus Torvalds



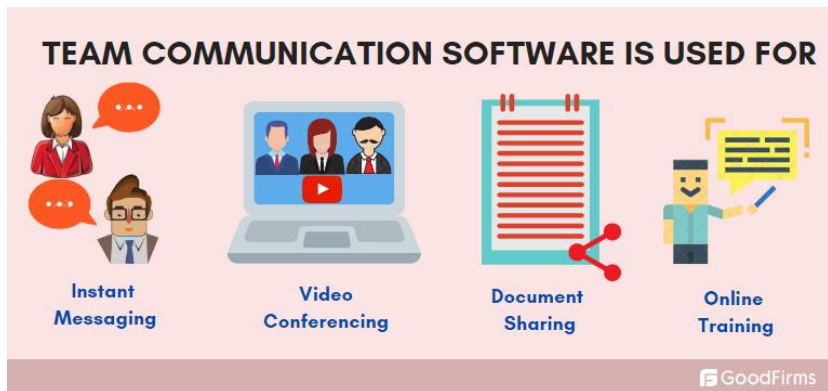
Linux 内核稳定分支(-stable)的维护者

你最喜爱的软件开发工具是什么？你通常使用什么？你在桌面一般运行哪些程序？

我重度依赖 Email 客户端 mutt 和 编辑器 vim，它们是我生存的必需品。其他我在日常工作中会用到的工具有：内核开发中使用 git 和 quilt；浏览器用 Chrome 和 Firefox；irc 通讯使用 irssi；桌面环境选择 GNOME 3，有时候用烦了也会退回到 OpenBox 或 i3m；每过一段时间我也会测试下 KDE，仅仅为了确保能够跟进开发进展。

开源参与者技能 – 沟通和协作

全球分布式开发，沟通工具灵活多样



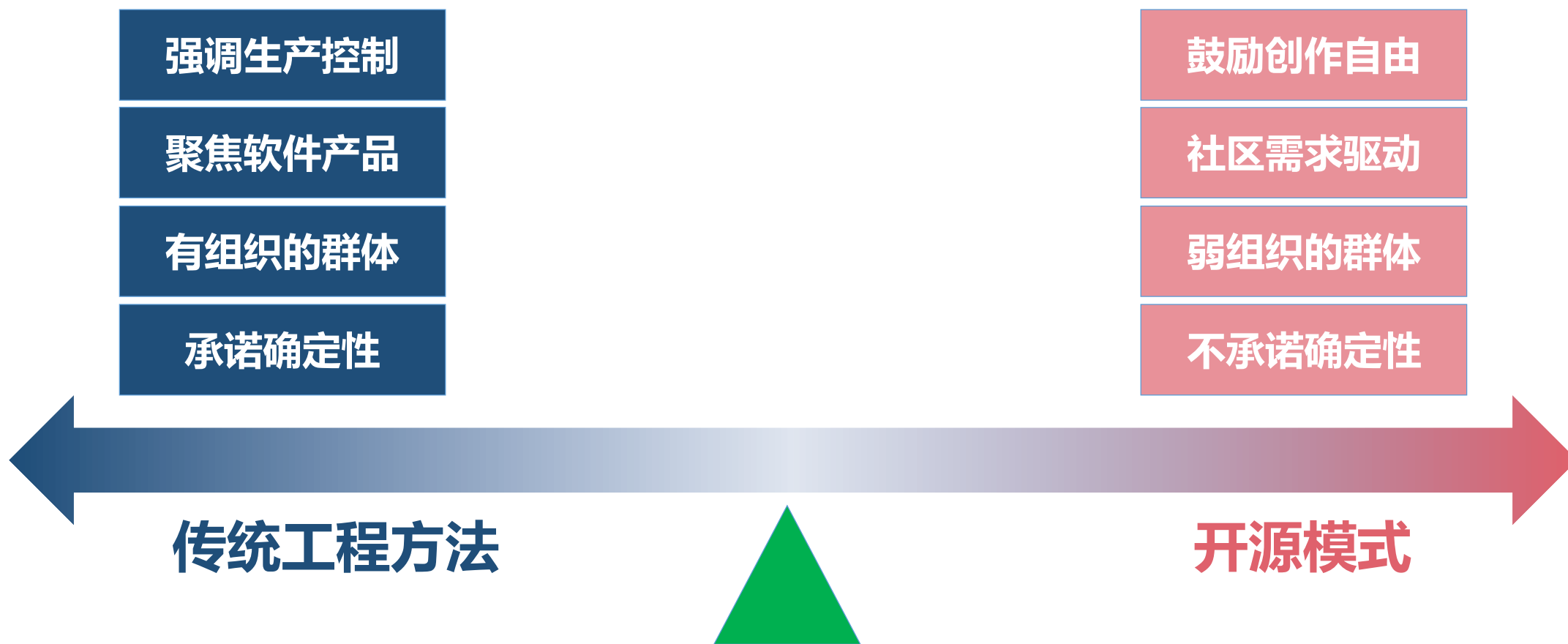
- 软件开发的复杂性： $n*(n-1)$ 的沟通路径， n 是开发者数量
- 开源开发：



开源模式：自下而上、开放协作

元素		工程化方法 自上而下，逐步求精	开源模式 自下而上，开放(黑客)合作
理念	需求	有明确用户和明确一致用户需求描述，这是软件开发的前提	没有明确的需求，开发者自我驱动开发软件特征
	质量	源代码满足需求描述的程度	黑客的自我要求及社区口碑
	效率	开发满足需求软件的时间和成本	行业最高水准的人们的自我驱动的效率
方法	过程模型	瀑布模型、敏捷模型、能力成熟度模型	开发者社区的自组织模式
	支持工具	软件产品线工具集	版本控制工具、缺陷追踪工具
	计算环境	面向计算机环境	面向互联网环境

传统工程化方法 vs. 开源模式



开源模式将能够做到的软件工程化以一种最佳模式呈现了出来。

公地悲剧

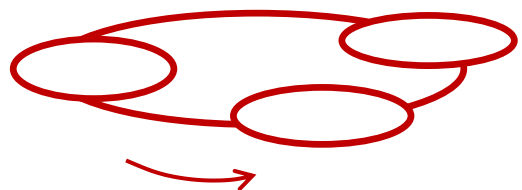


- 亚里斯多德：“那由最大人数所共享的事物，却只得到最少的照顾。”
- 公共草地上有一群牧羊人，每一个牧羊人都想要多获利一些，所以某个牧羊人就带了大量的羊来放牧，虽然他知道过度放牧，草地可能会承受不住。但他依然获利了，而后所有的牧羊人都跟进，所以草地牧草耗竭，悲剧因而发生了。
- 这个案例揭示有限的资源注定因自由使用和不受限的要求而被过度剥削。这样的情况之所以会发生源自于每一个个体都企求扩大自身可使用的资源，然而资源耗损的代价却转嫁所有可使用资源的人们。（可使用资源的群体数目可能远大于夺取资源的数目）

开源软件开发会发生公地悲剧吗？

- 开源软件是公共财产
- 开源软件开发跟公地悲剧的区别
 - 从使用上来说：开源软件的广泛使用呈现网络效应，不会因使用而发生损耗和枯竭。
 - 从供应上来说：开源软件的供应如何解决？如何保障无偿开发的人们的动力？

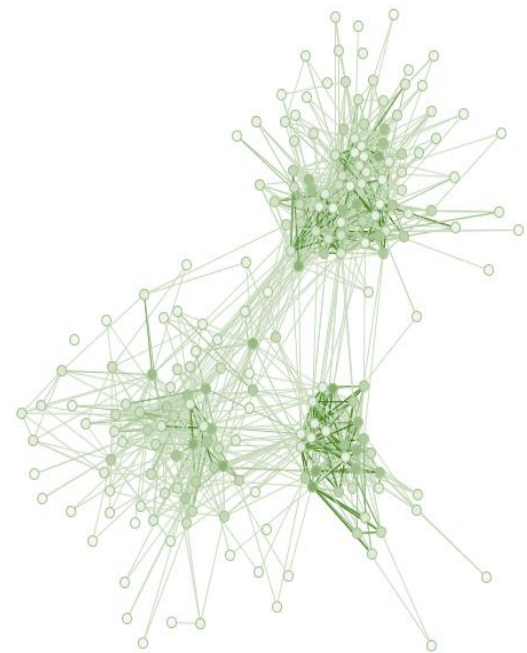
开源开发的核心挑战



问题1：群体如何激发？（人们为什么无偿做贡献？）

问题2：群体如何组织？（当企业加入，开源模式是否变成传统工程化模式？）

问题3：群智平台与生态？（如何提供自动化技术和工具支持群体开发、如何形成生态？）



鸣谢
王涛

TBC

- 经典开发过程和开源开发过程的实际案例？
 - Call for contribution: 企业内部发布一个产品版本的全过程, 和开源社区发布一个产品版本的全过程