开源软件与软件工程

方真

2023-12-05

目录

什么是开源软件

开源软件是指源代码对公众开放，允许自由使用、复制、修改和分发的软件

- 开放源代码：软件的源代码对任何人都是可用的，可以被查看和修改
- 透明性与可验证性
- 开放的设计与开发过程
- 遵循开源协议

OSI 对开源软件定义

- 自由分发
- 源代码开放
- 允许修改和派生
- 作者源代码的完整性
- 不歧视任何个人或团体
- 不歧视任何特定用途
- 许可协议的分发：无需额外许可即可使用
- 许可协议不局限于某个产品
- 许可协议不得限制其他软件
- 许可协议必须保持技术中立

FSF 对自由软件的定义

- 基于任何目的使用该软件的自由
- 研究软件如何工作，修改软件的自由
- 重分发该软件的自由
- 重分发派生版本的自由

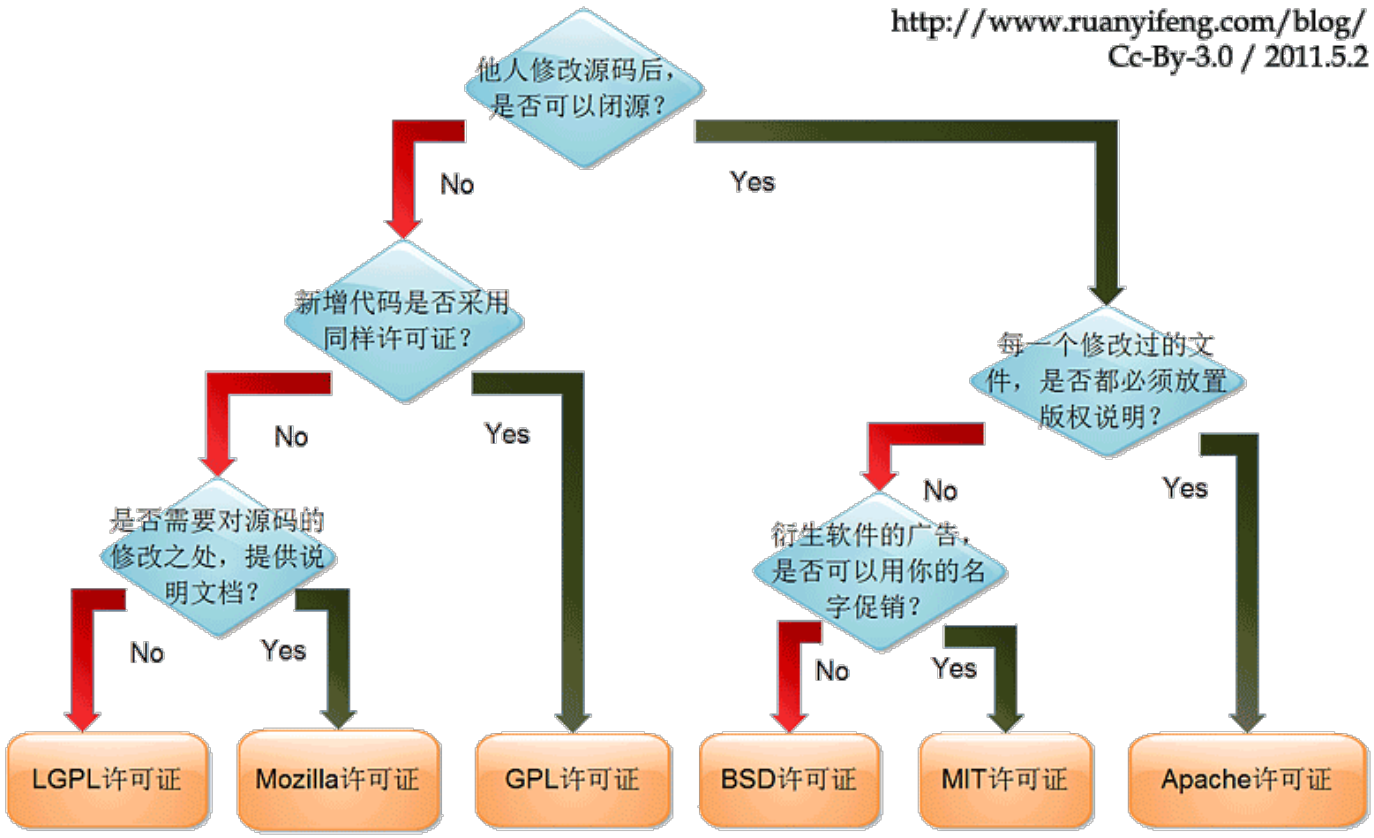
开源软件简史

- 早期软件著作权从无到有
- Unix 与 C 语言的诞生：60 年代末到 70 年代初
  - 源码可近乎免费获得，可用于非商业用途
- 70 到 80 年代：越来越多的公司将软件作为财产，源码受保护，无法免费获取
  - 1976 比尔盖茨《致爱好者的公开信》
- 80 到 90 年代，随着 AT&T 对 SystemV 商业版收费和限制，BSD Unix 逐步发展起来
  - 至今 OpenBSD/NetBSD/FreeBSD 依然在开发
- 1983 年，Richard Stallman 发起了 GNU 计划
- 1991 年，Linus 发布第一版 Linux 内核。GNU/Linux 成为了一个完全自由的开源操作系统
- 1998 年，Eric Raymond 和 Bruce Perens 成立了开源促进组织 (Open Source Initiative)。
- 2004 年，中国开源软件推进联盟成立
- 2020 年，开放原子开源基金会成立，是中国内地首个开源领域的基金会

开源许可证

开源许可证可以粗略地分为两大类：

- 著佐权许可证 ("Copyleft license")
  - 在软件被修改并再发行时，仍然强制要求公开源代码
- 宽松自由软件许可协议 ("Permissive free software licence")
  - 衍生软件可以变为专有软件



开源软件的例子

- 操作系统内核：Linux、BSD、AOSP
- 浏览器：Firefox、Chromium
- 数据库：Mariadb、PostgreSQL
- 云计算：Openstack、Kubernetes
- 虚拟化：Qemu、Bochs
- 编程语言：Java、Python、Go、Rust
- 编译器：GCC、LLVM
- Web 服务器：Httpd、Nginx
- 开发工具：Git、Eclipse、Emacs、Vi
- Web & 桌面：Angular、Vue.js、Flutter
- AI 框架：TensorFlow、Pytorch
- 多媒体：FFmpeg、VLC
- 科学计算：NumPy

目录

开源软件工程实践的特点

总的来说，开源项目中的软件工程实践强调了社区参与、透明度、协作和持续交付，这些特点使得开源项目具有更强的创新能力和灵活性。

- 分散的开发者群体
- 透明度和公开性
- 社区参与和治理
- 持续集成和持续交付（CI/CD）
- 开放式问题跟踪和协作
- 文档的重要性
- 代码评审和协作

案例：Openstack 项目

OpenStack 是一个开源的云计算平台，旨在提供基础设施即服务（IaaS）和平台即服务（PaaS）解决方案。由 Open Infrastructure Foundation 负责运营。许可协议采用 Apache 2.0。

治理与组织结构

- 董事会对 OpenStack 基金会以及基金会所保护的资产 (如 OpenStack 商标) 进行监督。由赞助商指定以及选举产生。
- 技术委员会 (TC) OpenStack 项目的最高技术决策机构。TC 成员由选举产生，负责项目技术方向、标准、项目治理规则等决策。
- 用户委员会用户委员会代表用户利益，与其他方进行合作，确保 Openstack 项目方向符合用户需求。
- 项目团队
  - OpenStack 项目被组织成一系列的项目组，每个项目组负责一个或多个相关的项目。
  - 每个项目组都有一个项目组长 (Project Team Lead，PTL) 负责组织和协调项目组的活动。
  - 每个项目组都有多个 Core Reviewer

项目管理

- Openstack 项目是一直发展的，从最初的 Nova 到现在几十个项目。
- 新项目的准入是由 TC 来评估和决定；同时项目开发者可以获得 TC 的投票权。
- 必须满足 Openstack 要求 (4 Opens):
  - 开放源码
  - 开放社区
  - 开放开发
  - 开放设计
- Openstack 的项目管理机制几经变化，目前流程有所简化。

Feature 管理

- Blueprint 在 Openstack 项目中用来追踪重大特性的实现。
  - 包含了详细规划和设计文档。
  - 由社区成员创建，并经过讨论、审查和批准。
- Blueprint 的生命周期:
  - 提出与创建，上传设计文档到代码库；
  - Blueprint 被批准，其中会经过讨论与反馈，修改与评审；
  - 由提出者或其他人实现，并保持进度更新；
  - 需求实现，状态变成完成

Bug 追踪系统

Openstack 项目使用 launchpad 来进行 bug 与任务追踪。

- 通常来说，Bug 要有以下几个信息：
  - Bug 基本信息：现象、触发条件等
  - 状态
  - 优先级
  - 报告人和负责人
  - 目标版本，受影响版本
  - 其他标签
- Bug 的主要生命周期
  - 报告
  - 确认优先级
  - 修复方案的实现
  - 完成

沟通与文档

开源项目的协作模式决定了它不同于商业软件的沟通方式。沟通主要发生在:

- 开发者和社区内部
- 外部用户与开发者

沟通方式:

- 各种需求管理，任务追踪系统
- 即时通信
- 邮件列表
- 文档
  - 文档在开源项目中处于核心地位
  - 高质量的文档对于开源项目有巨大的助益

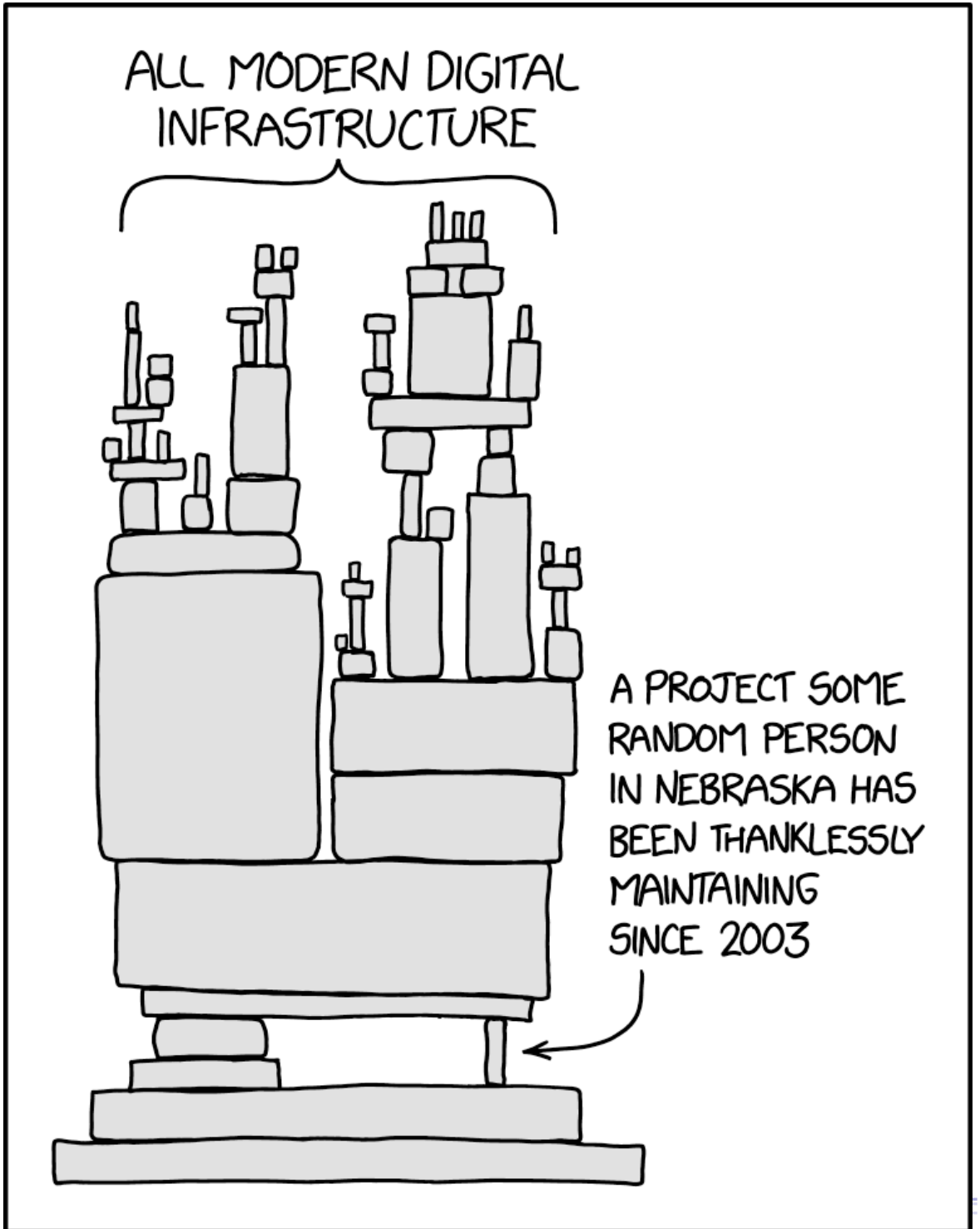
代码托管与评审

- Openstack 项目采用 Gerrit 来管理代码。
  - Gerrit 是一个基于 Git 的代码评审和管理系统
  - 一切皆可代码化
- 代码都需要经过评审才能进入代码库
  - 每个 patch 提交之后都会自动执行自动化测试
  - 贡献者可以邀请其他人参与评审
  - 项目的 Core Reviewer 需要同意
- 分支模型

生态

- 赞助商：Openstack 赞助商分为白金赞助商，黄金赞助商，白银赞助商
- 发行版：Redhat、Canonical、华为等
- OpenInfra 峰会
- COA 认证与培训
- 用户：2022 年数据，全球 300 个公有云数据中心，4000 万 CPU Core 的部署规模

目录


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

实例：OpenSSL heartbleed 漏洞

Heartbleed 是 OpenSSL 的一个严重漏洞，它允许攻击者在正常情况下窃取本应受 SSL 协议加密保护的信息。

- Heartbleed 是 OpenSSL 在心跳机制的代码实现中产生的漏洞，并非 SSL 协议中的设计缺陷。
- OpenSSL 可能是使用最广泛的 SSL/TLS 实现：
  - nginx、apache httpd 都使用 OpenSSL，两者合计占有一半以上的 Web server 市场
  - 众多 Linux 发行版和 BSD 发行版都包含 OpenSSL
- 漏洞 2012 年引入，2014 年 4 月公开。期间可能有未被披漏的利用。

类似问题

- log4j 漏洞：CVE-2021-44228
- core-js 维护问题：
  `https://github.com/zloirock/core-js/blob/master/docs/2023-02-14-so-whats-next.md`

挑战：项目本身

- 项目开发过程
  - 代码风格与质量
  - 核心开发者的开放性
- 资源有限
- 项目运营
  - 成功的项目需要重视代码之外的建设
- 问题修复和通知的挑战

挑战：项目之外

- 广泛影响
- 关注依赖链的复杂性
- 及时关注并修复安全漏洞
- 选取开源项目时的评估
- 赞助开源项目，促进良性发展

没有银弹

开源软件有虽然诸多优势，但并不能解决软件开发的所有问题。

- 项目可持续性
- 安全风险
- 许可问题
  - Redis、Mongo 许可变更
- 版本兼容性
  - 开源项目对兼容性的哲学与商业目标不一定一致
- 社区支持有限
  - 当缺乏足够的技能解决开源项目的问题时，无法像商业软件一样寻求支持
- 过时的版本
  - 91%的商业软件包含过时或废弃的开源组件
  - 升级难度
- 社区分裂
  - MariDB vs. MySQL
  - 派生版本与主线开源版本分裂

业界方案

开源生态产品化：将开源软件或技术整合到一个完整的产品或解决方案中，并通过商业化的方式提供给最终用户或企业。

- 商业支持和服务
- 可扩展性和定制性
- 安全性和合规性
- 用户友好的界面

软件工程在开源生态产品化中发挥着关键作用

- 通过软件工程的系统性思维来解决产品化过程中的问题
- 着眼于整个产品和方案，而不只是具体的代码实现
- 可维护性是软件生命周期的一个重要而关键的阶段

目录

参与理由

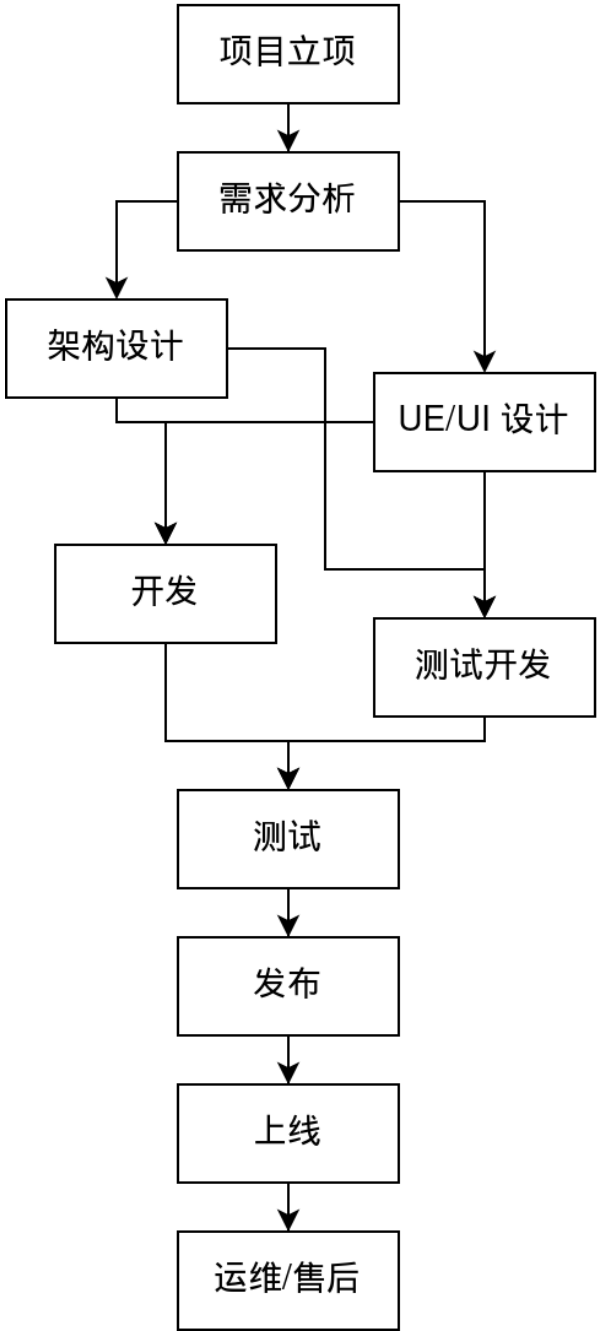
参与开源项目是学习和实践软件工程的绝佳选择

- 获得实际项目经验
  - 了解真实世界的软件开发挑战和流程
  - 比教科书学习更加深入的体验
  - 可以实践软件工程的方法学
- 锻炼协同合作的能力
  - 能够与来自不同背景和地区的开发者合作
- 提升技术能力
  - 养成良好的设计和编程习惯
  - 学习新技术

几点建议

- 保持平常心
- 了解并融入社区文化和技术风格
- 选择感兴趣的项目
- 动手而不是观望
- 多样化贡献
  - 编码、文档、测试、基础设施、提交反馈
- 参与面向学生的开源活动
  - 如开源之夏：中科院软件所发起并支持

真实世界的软件开发流程



市场调研，公司战略考量

具体需求调研与细化，还要考虑竞品和客户

架构设计考虑技术选型，组件间依赖与接口等

设计产品交互，观感

模块内详细设计与编码开发

测试用例，自动化测试等

不仅要考虑代码功能，还要考虑运行环境，部署等

根据实际情况，可能要发布alpha、beta等版本

上线到测试环境或生产环境

可能发生硬件、软件故障；用户不会用等各种情况

Thank You!

Q&A