CodeWisdom软件供应链系列学术报告

开源软件供应链管理

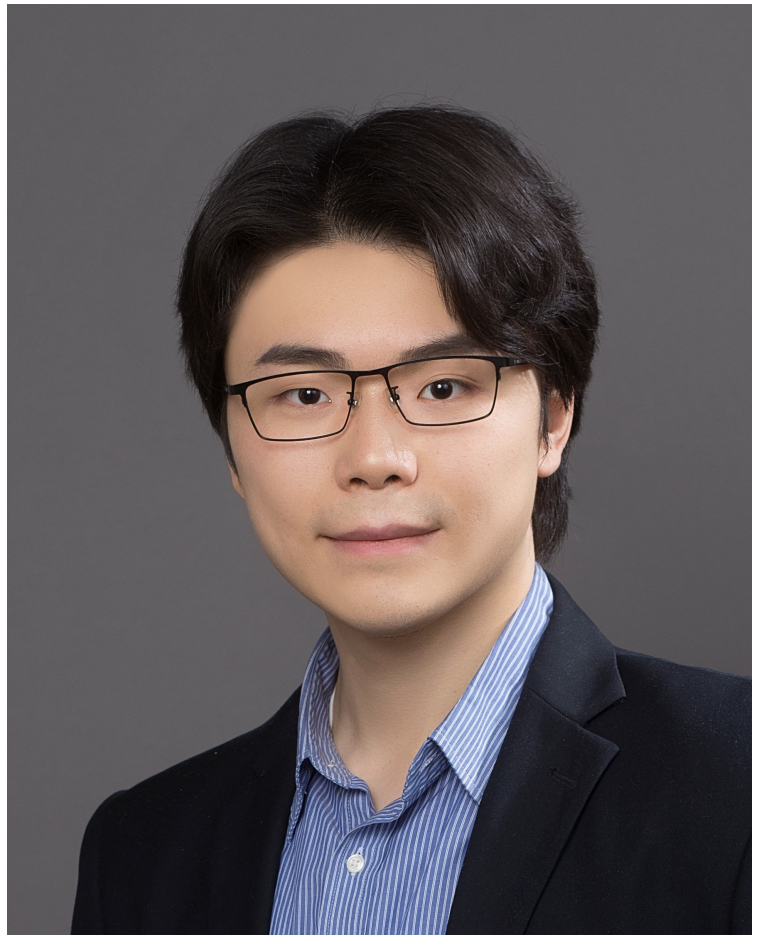
何　昊

北京大学 计算机学院

heh@pku.edu.cn

自我介绍

- 北京大学计算机学院博士研究生
- **个人主页：**https://hehao98.github.io/
- **实验室主页：**https://osslab-pku.github.io/
- **导师：**周明辉 教授 https://minghuizhou.github.io/
- **研究方向：**软件工程
  - 开源软件供应链管理
    - 开源三方库迁移 [SANER'21][ICSE'21][ESEC/FSE'21]
    - 依赖管理Bot [arXiv'22]
    - 版本发布说明 [ICPC'22]
    - API不兼容变更
  - 开源社区新人引导机制
    - Good First Issue推荐 [ICSE'22]
    - 开源教育



Disclaimer

- 主题宏大，仅代表本人视角和观点，抛砖引玉，如有不足请多海涵
- 偏向**软件工程领域**的相关研究科普 + 综述 + 介绍本人研究
  - **可能缺乏：**安全领域视角、业界视角
- 欢迎随时打断，提问讨论
- 联系方式 heh@pku.edu.cn
- PPT下载地址
  - https://hehao98.github.io/files/2022.7.21-开源软件供应链管理-CodeWisdom.pdf

报告概览

第一部分 **开源软件供应链的定义**

第二部分 开源软件供应链的现状

第三部分 开源软件供应链管理的框架和重要问题

第四部分 相关研究和本组工作

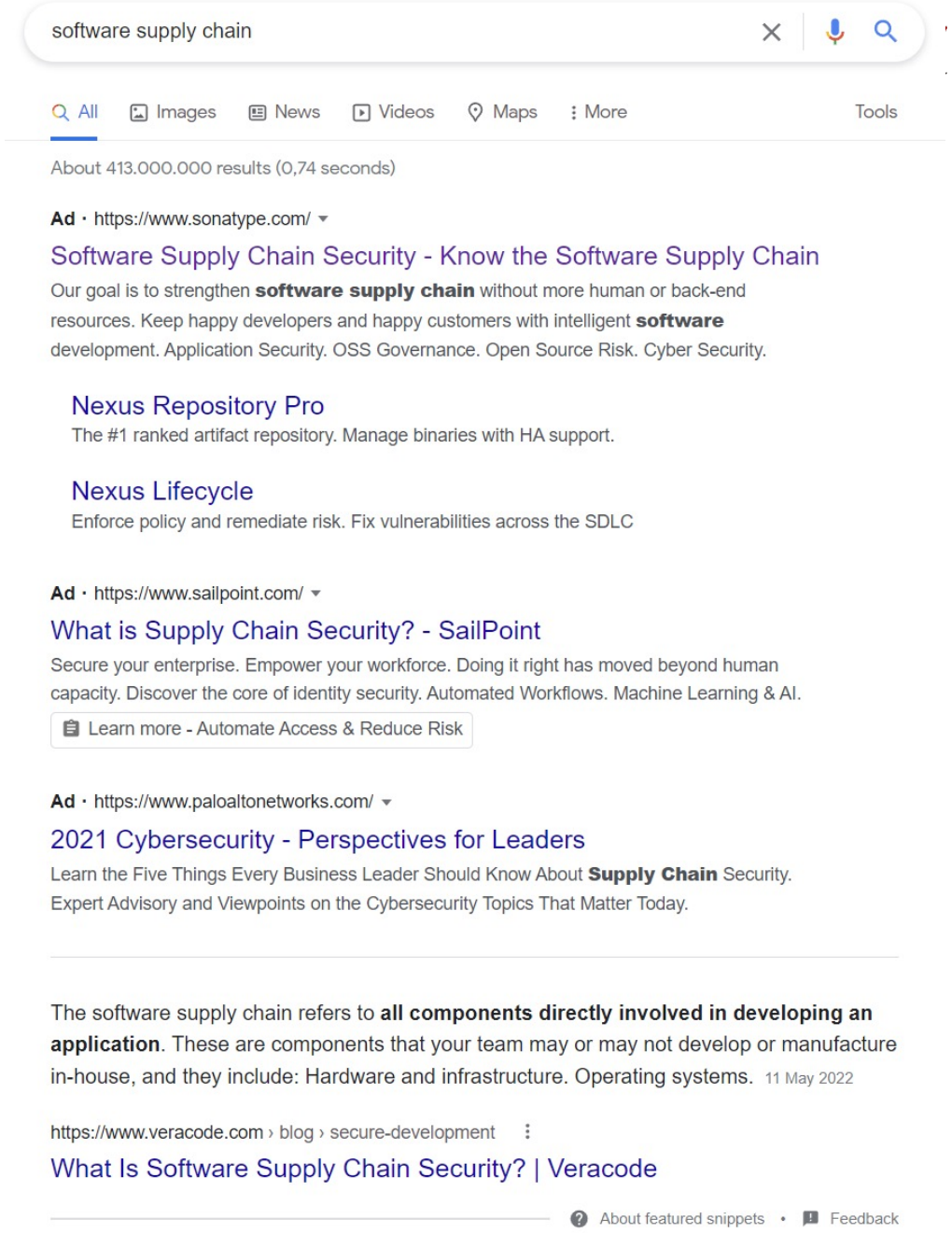
为什么要先下定义？

- 软件供应链更像一个 “buzzword”
  - 缺乏公认的/权威的定义

- 软件工程的论文较少使用 “软件供应链” 一词
  - package ecosystem, 3rd-party library, API…

  ⇒ 需要一个定义涵盖已有研究

- 本报告将会尝试给出/引用若干定义
  - 不保证完备性
  - 欢迎指出不足



什么是软件供应链？——广义的定义

WIKIPEDIA
The Free Encyclopedia

A **software supply chain** is composed of the components, libraries, tools, and processes used to develop, build, and publish a software artifact. (Wikipedia)

+ maintain and operate

什么是软件供应链？——例子

开发所需的软件：SDK、IDE、版本管理、代码托管、...

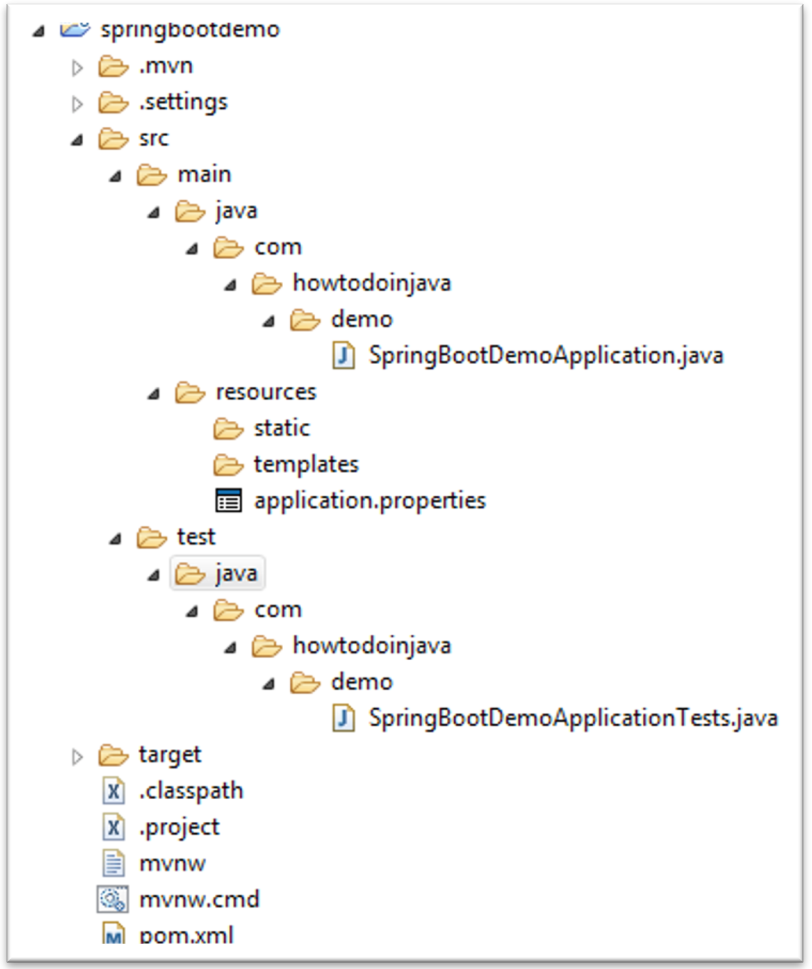
构建所需的软件：编译器、包管理器、其他组件（库）

运行所需的软件：编程语言运行环境、操作系统、数据库...

供应

供应

供应



一个基于Spring Boot的Java后端项目

为什么叫开源软件供应链？

开发所需的软件：SDK、IDE、版本管理、代码托管、...

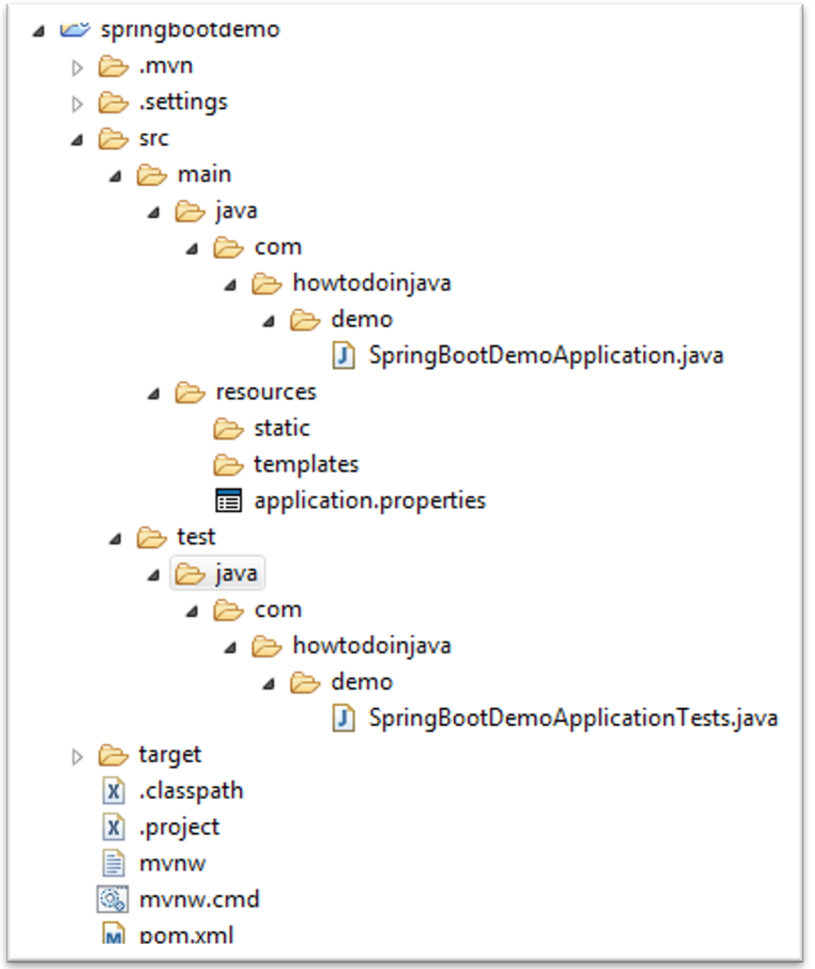
构建所需的软件：编译器、包管理器、其他组件（库）

运行所需的软件：编程语言运行环境、操作系统、数据库...

供应

供应

供应



一个基于Spring Boot的Java后端项目

什么是软件供应链？—— 一些相关的术语

- 软件材料清单（SBOM, Software Bill of Materials）
  ⇒ 软件供应链中包含的软件

- 软件生态系统（Software Ecosystem）
  - A collection of software products that have some given degree of symbiotic relationships (Messerschmitt et al., 2003)
  - Software ecosystems are sets of software solutions functioning as a unit, enabling actors to automate activities and transactions…A software ecosystem constitutes the interactions of a set of actors on top of a common technological platform that results in a number of software solutions or services. Each actor is motivated by a set of interests or business models and connected to the rest of the actors and the ecosystem as a whole with symbiotic relationships, while, the technological platform is structured in a way that allows the involvement and contribution of the different actors (Manikas et al., 2012)

- **简单理解：**软件供应链 ≈ SBOM + 依赖关系，软件供应链 ⊆ 软件生态系统

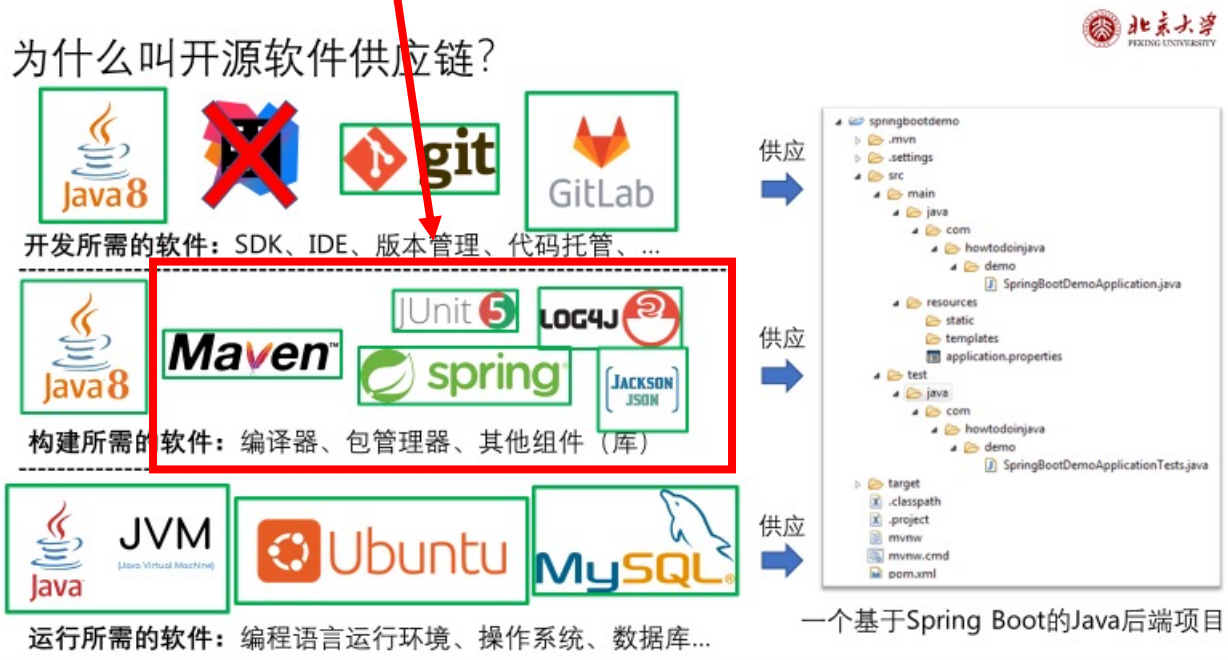
什么是软件供应链？——一些相关的术语

- 软件包（Software Package）
  - 面向复用的某个软件都可以叫软件包，更多指上传到包管理平台上的软件
  - **常用近义词：**包（package)、库（library）、组件（component）、框架（framework）、...
- 包托管平台（Package Hosting Platforms）
  - 公开托管软件包便于其他项目复用的平台
  - JS/npm、Java/Maven、Python/PyPI、...
- 包管理器（Package Managers）
  - 从包托管平台获取软件包供当前项目使用的开发工具
  - npm、Maven、pip/poetry、...

什么是软件供应链？——便于研究操作的定义

- 相当多的软件工程研究可以用如下定义覆盖

一个软件项目的**开源软件供应链**包括：通过这个项目的包管理器，从开源包托管平台所获取的所有软件包以及软件包之间的依赖关系



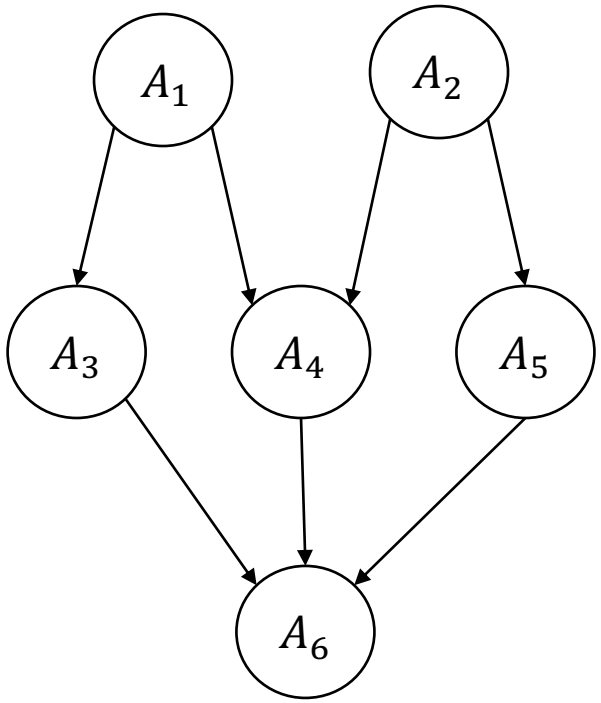
- 为何只研究这部分软件供应链？
  - 易于精确还原
    => 可自动化分析
  - 包数量庞大、包内容高度多样化
    => 研究结果的通用性较好
  - 包来源复杂、质量参差不齐、问题更加突出
    => 需要供应链管理
  - 存在极端复杂的依赖关系
    => 新的研究问题

什么是软件供应链？——形式化的视角

Software Supply Chain: A directed graph $G = \langle V_{up}, V_{down}, E \rangle$

- V_{up} are upstream software artifacts,
- V_{down} are downstream software artifacts that depend on software artifacts in V_{up},
- and $E \subseteq V_{up} \times V_{down}$ are dependencies between upstream and downstream software artifacts.

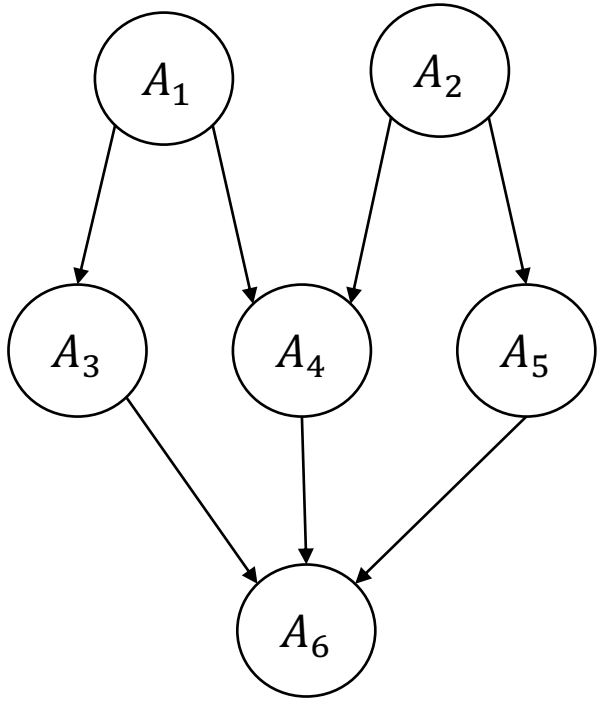
- **Software Artifacts**: modules, packages, binaries, etc.
- ≈ software dependency graph



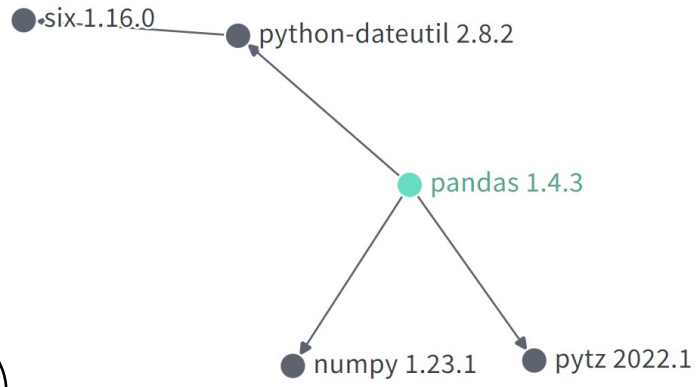
$V_{up} = \{A_1, A_2, A_3, A_4, A_5\}$
$V_{down} = \{A_3, A_4, A_5, A_6\}$

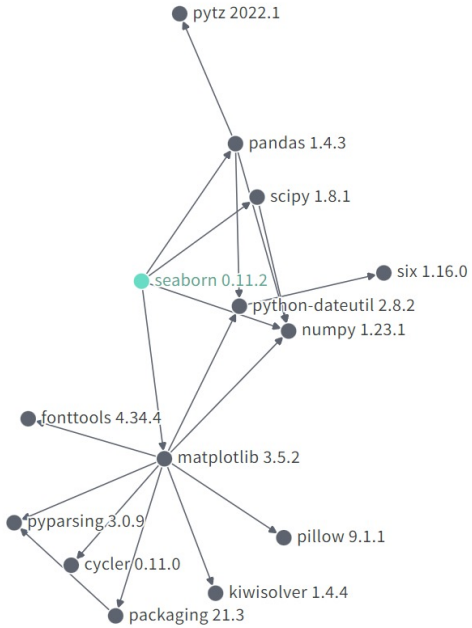
Tan, Xin, Kai Gao, Minghui Zhou, and Li Zhang. “An Exploratory Study of Deep Learning Supply Chain.” In *Proceedings of the 44th International Conference on Software Engineering (ICSE 2022)*.

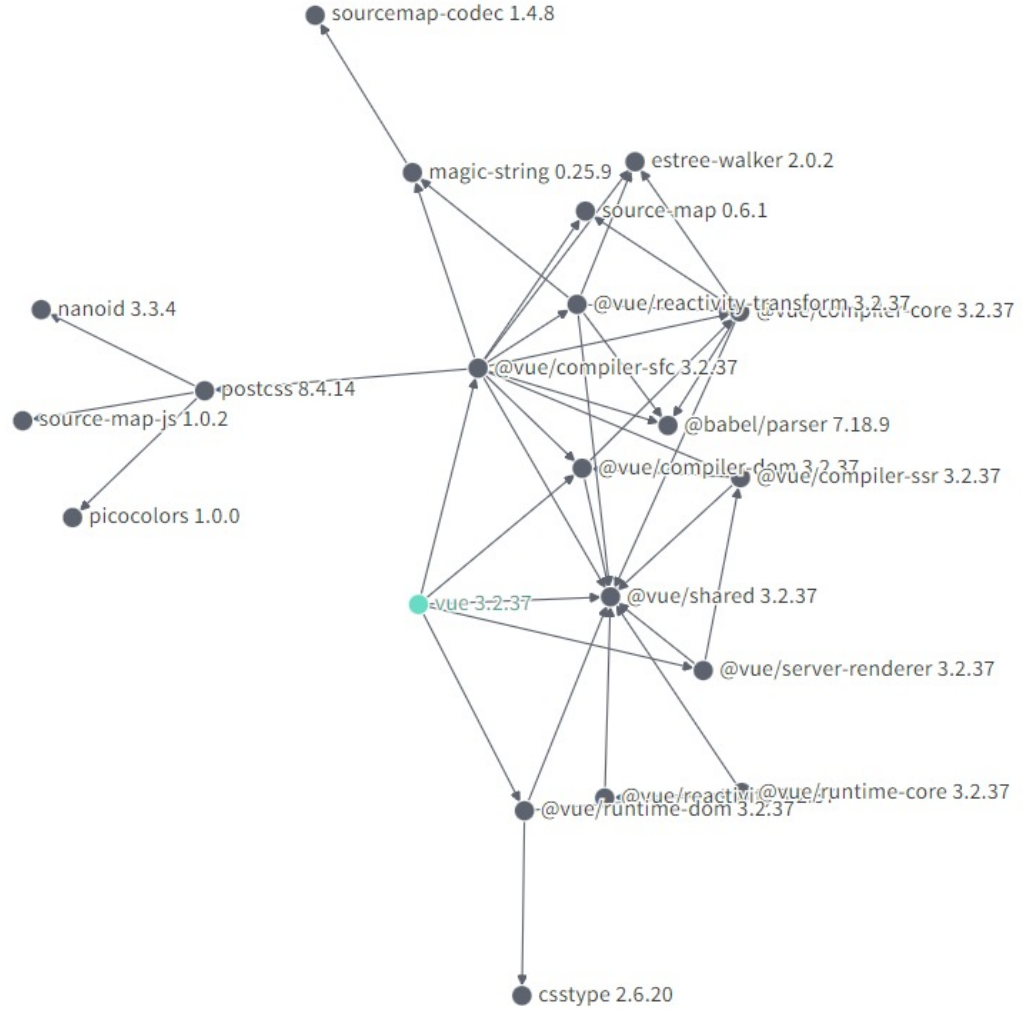
什么是软件供应链？——形式化的视角



$$V_{up} = \{A_1, A_2, A_3, A_4, A_5\}$$
$$V_{down} = \{A_3, A_4, A_5, A_6\}$$

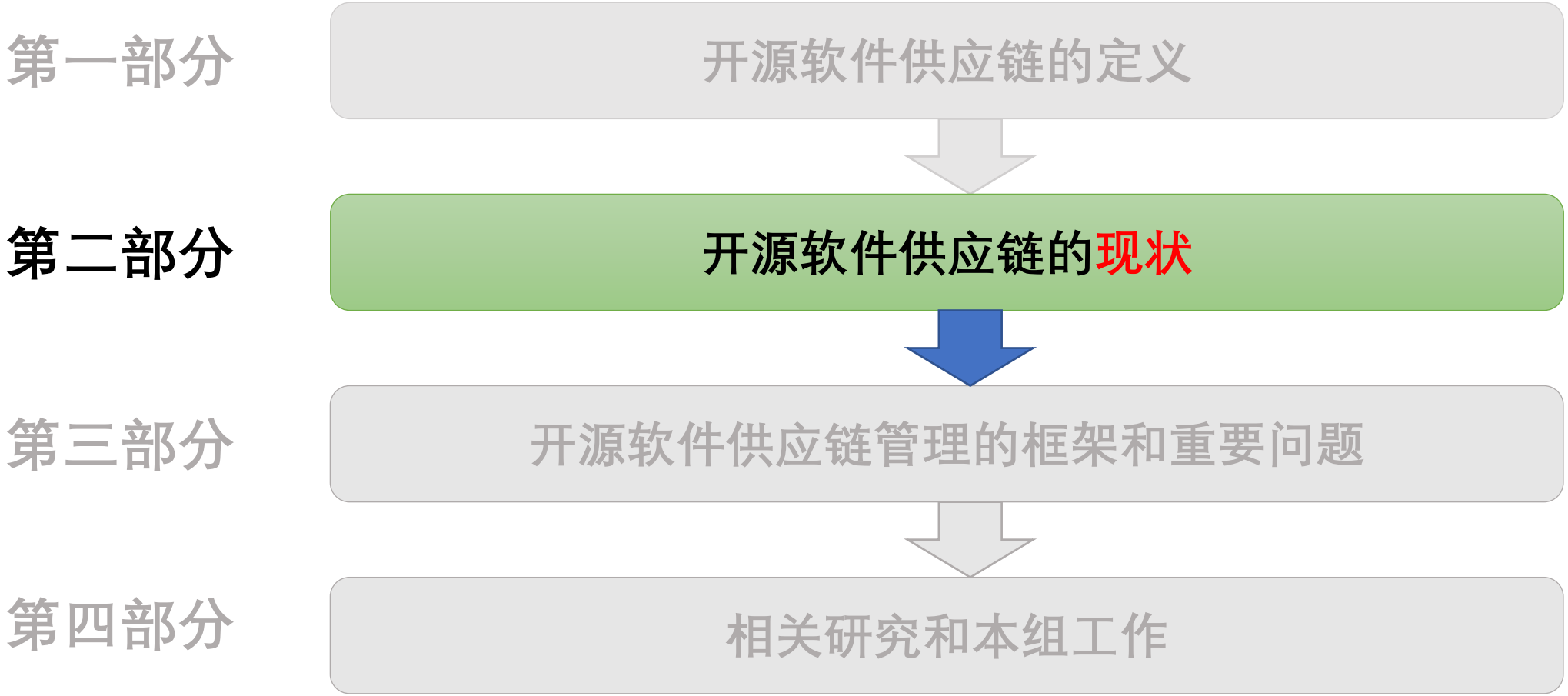






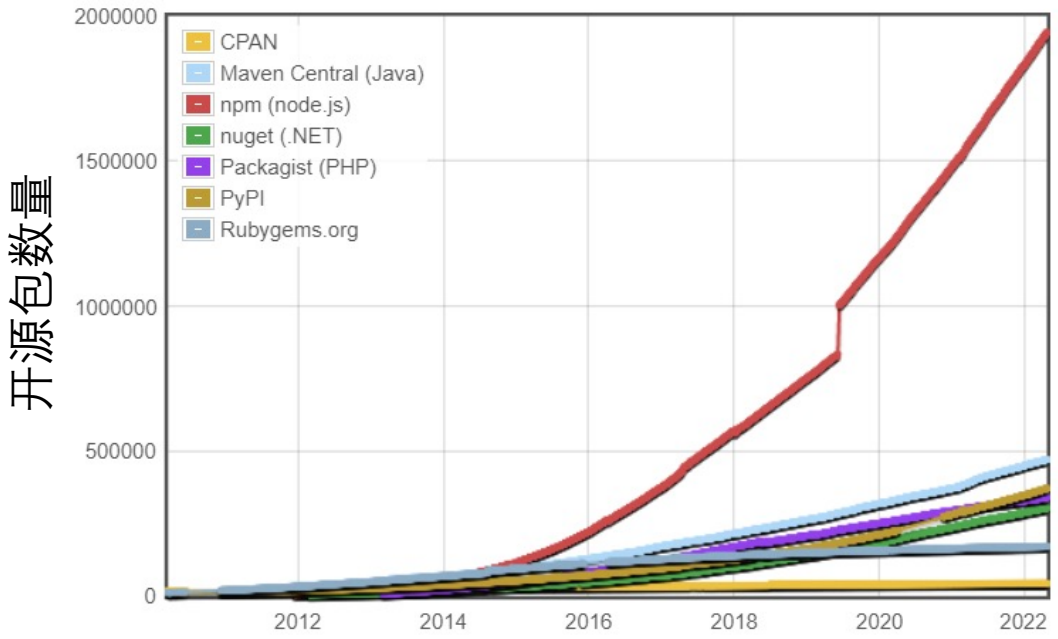
https://deps.dev/

报告概览

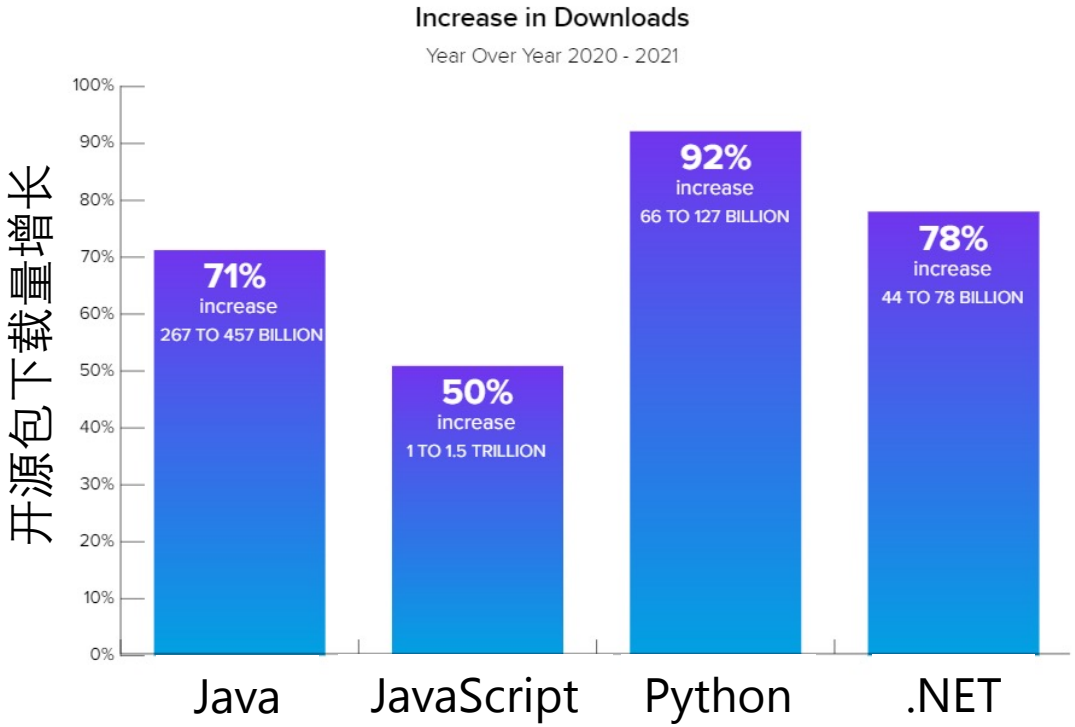

第一部分
开源软件供应链的定义
第二部分
开源软件供应链的现状
第三部分
开源软件供应链管理的框架和重要问题
第四部分
相关研究和本组工作

开源软件供应链的现状 – 高速增长

- **供给端：**软件包**数量**高速增长
- **需求端：**各个包托管平台的软件包**下载量**高速增长



http://www.modulecounts.com/



https://www.sonatype.com/resources/state-of-the-software-supply-chain-2021

开源软件供应链的现状 – 事实标准

- **面向开发者 ≈ 开源模式**
  - 工具、框架、运行平台、数据库、...

- **最佳实践 ≈ 依赖开源软件供应链**
  - **机器学习：**numpy、scikit-learn、PyTorch、...
  - **Web开发：**angular/react/vue、eslint、...
  - **其他各种领域...**

开源软件供应链的现状 – 质量问题

- **（初级）开发者：**开源的，质量一定很高！

OSS Devs: You can you up 

- **真实情况：**没有测试、缺乏维护、永远0.x版本、API不稳定、总能找出bug…

Contrary to developers' beliefs, only around 28% of npm and 49% PyPI trivial packages have tests…(they) appear to be "deployment tested" (Abdalkareem et al., EMSE, 2020)

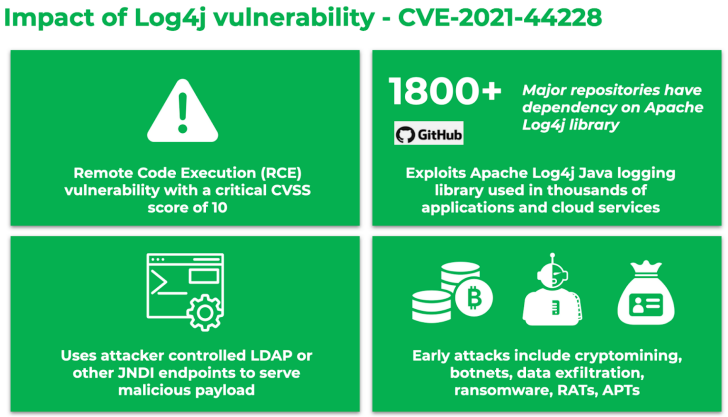
The number of downloads of a package does not associate with development activity (Saini et al., J. of Sys. Soft., 2020)

The version number does not provide a good indication of the maturity of a package release (Decan and Mens, Sci. of Comp. Prog., 2021)

We find that around one third of all releases (in Maven) introduce at least one breaking change, and that this figure is the same for minor and major releases, indicating that version numbers do not provide developers with information in stability of interfaces (Raemaekers et al., SCAM 2014)

开源软件供应链的现状 – 安全漏洞

- 开源包中存在有可能被恶意利用的Bug



2021年12月曝光的Log4j**远程代码执行**漏洞(CVE-2021-44228)是当之无愧的IT界 “地震”

https://www.zscaler.jp/blogs/security-research/weekly-roundup-what-weve-learned-about-log4j-vulnerability

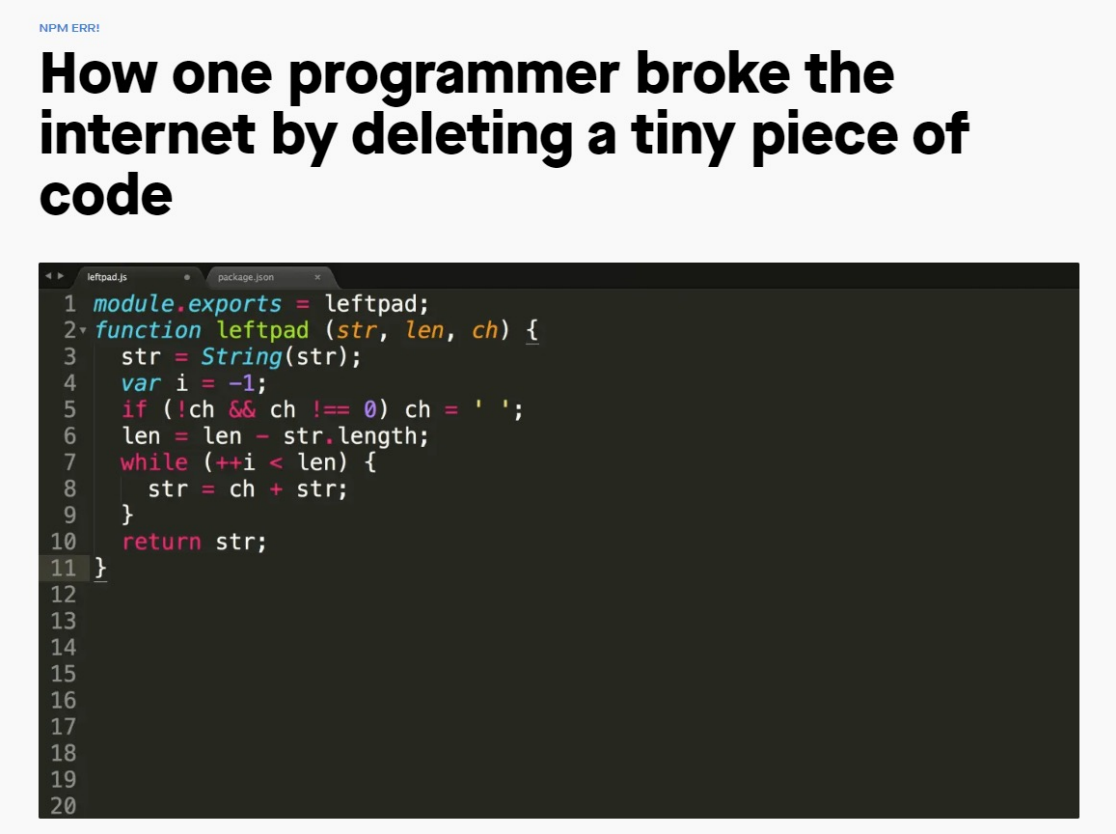
…it often takes a long time to discover vulnerabilities since their introduction (Decan et al., MSR 2018)

We statistically prove that vulnerabilities widely exist in the dependencies of NPM packages…Known vulnerabilities are causing a larger impact over time (Liu et al., ICSE 2022)

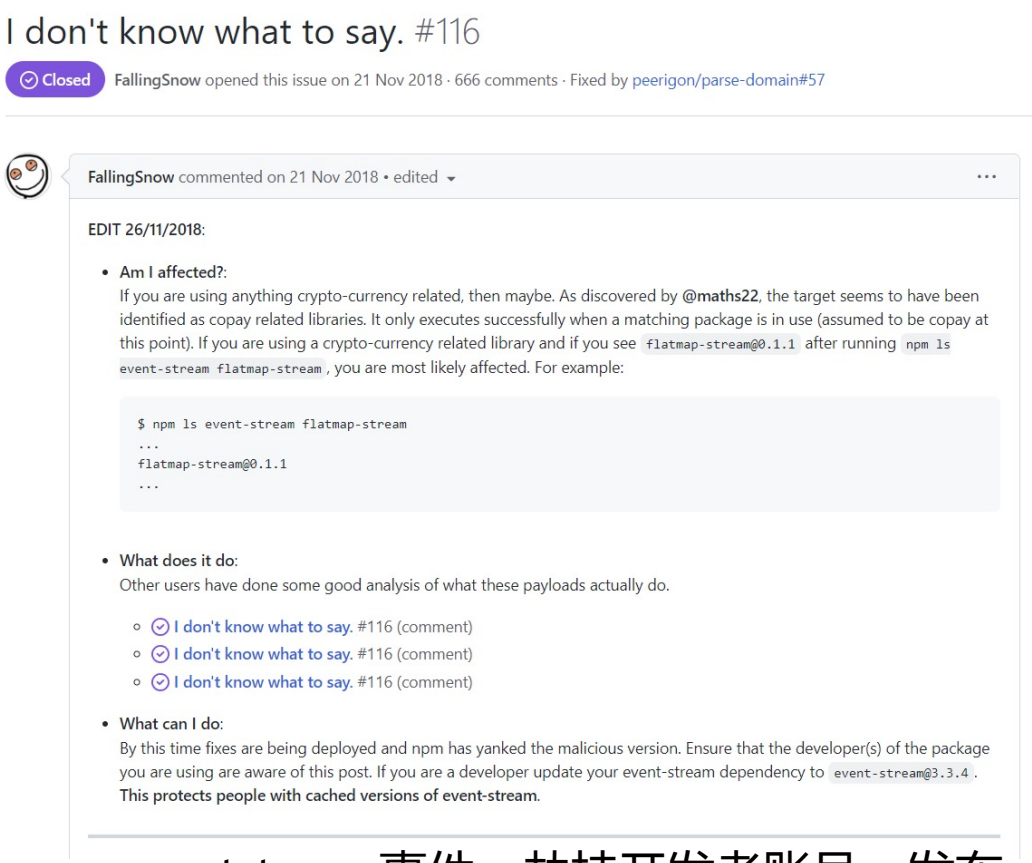
The discovered vulnerabilities in Python packages are increasing over time, and they take more than 3 years to be discovered. The majority of these vulnerabilities (50.55%) are only fixed after being publicly announced, giving ample time for attackers exploitation (Alfadel et al., SANER 2021)

开源软件供应链的现状 – 供应链攻击

- 出于有意或者无意对大量软件项目的供应链造成破坏的事件

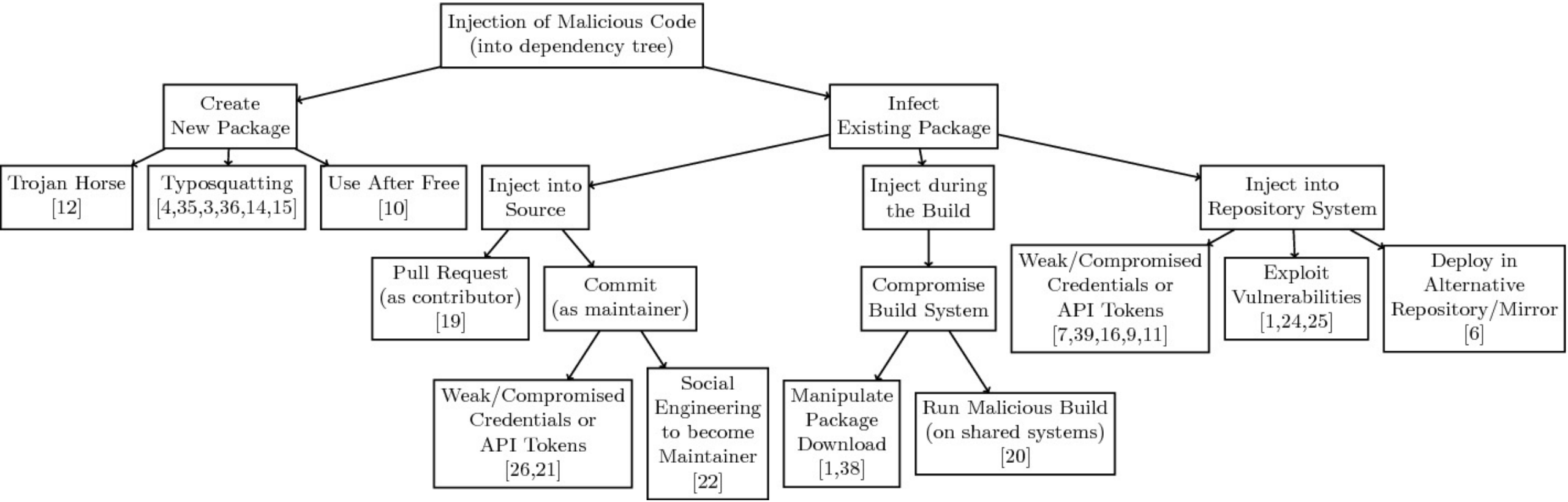


Left-Pad事件：删自己的包，带崩整个互联网



eventstream事件：劫持开发者账号，发布植入挖矿病毒的新版本

开源软件供应链的现状 – 供应链攻击



Ohm, Marc, et al. "Backstabber's knife collection: A review of open source software supply chain attacks." *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, Cham, 2020.

开源软件供应链的现状 – 可持续性

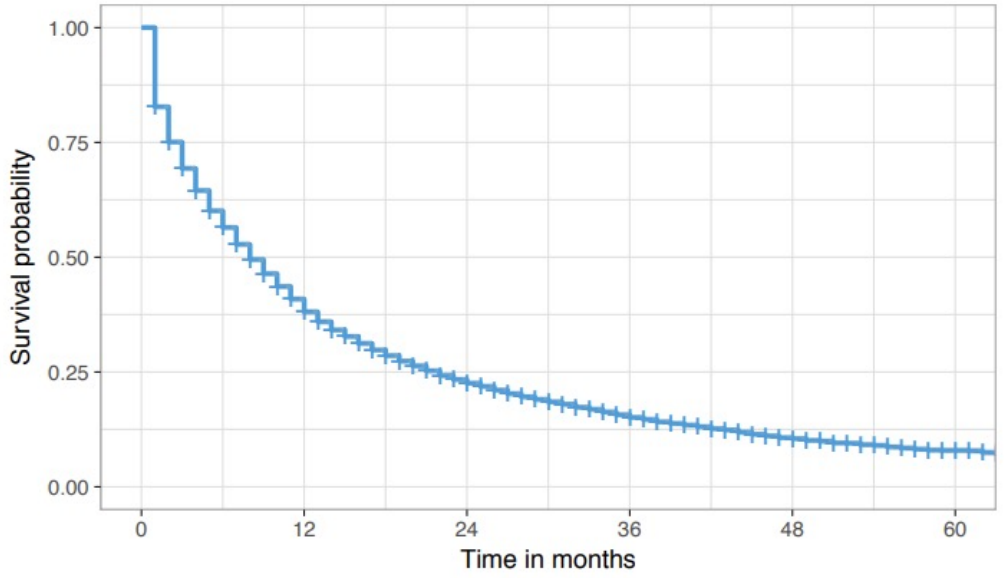
…significant number of (GitHub) projects die in the first year of existence with the survival rate decreasing year after year (Ait et al., MSR 2022)

Table 2: Why open source projects fail?

Reasons	Group	Projects
Usurped by competitor	Environment	27
Obsolete	Project	20
Lack of time	Team	18
Lack of interest	Team	18
Outdated technologies	Project	14
Low maintainability	Project	7
Conflicts among developers	Team	3
Legal problems	Environment	2
Acquisition	Environment	1

Coelho, Jailton, and Marco Tulio Valente. "Why modern open source projects fail." *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 2017.

Python包的生存曲线



Valiev, Marat, Bogdan Vasilescu, and James Herbsleb. "Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem." *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2018.

开源软件供应链的现状 – 可持续性

Marak/faker.js

#1046 **No more free work from Marak - Pay Me or Fork This**

98 comments

 **Marak** opened on November 8, 2020

https://www.reddit.com/r/javascript/comments/jquo97/faker_js_no_more_free_work_from_marak_pay_me_or/

We find a long-tail effect in the act of sponsorship, with most maintainers' expectations remaining unmet, and sponsorship has only a short-term, slightly positive impact on development activity but is not sustainable (Zhang et al., CHI 2022)

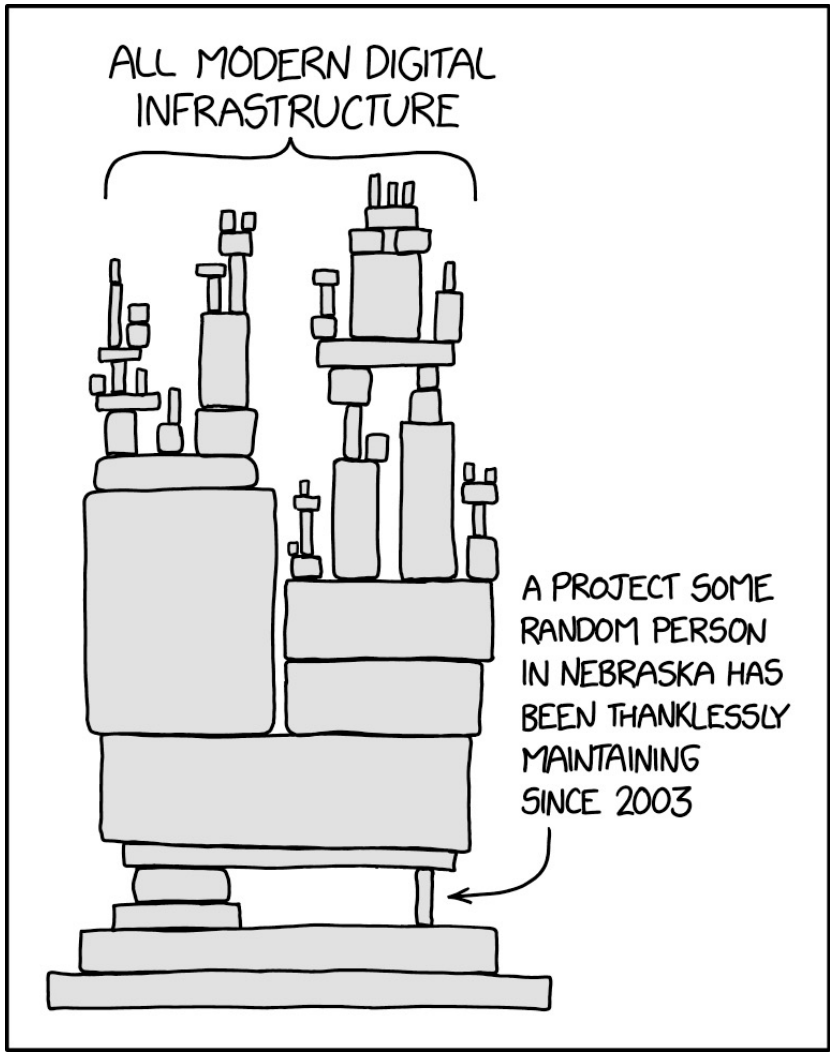
开源软件供应链的现状 – 结构脆弱

- 少量软件包被大量软件包依赖
  ⇒出现问题则影响巨大
  - 不兼容变更、安全漏洞、供应链攻击、...

Kikas, Riivo, et al. "Structure and evolution of package dependency networks." *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017.

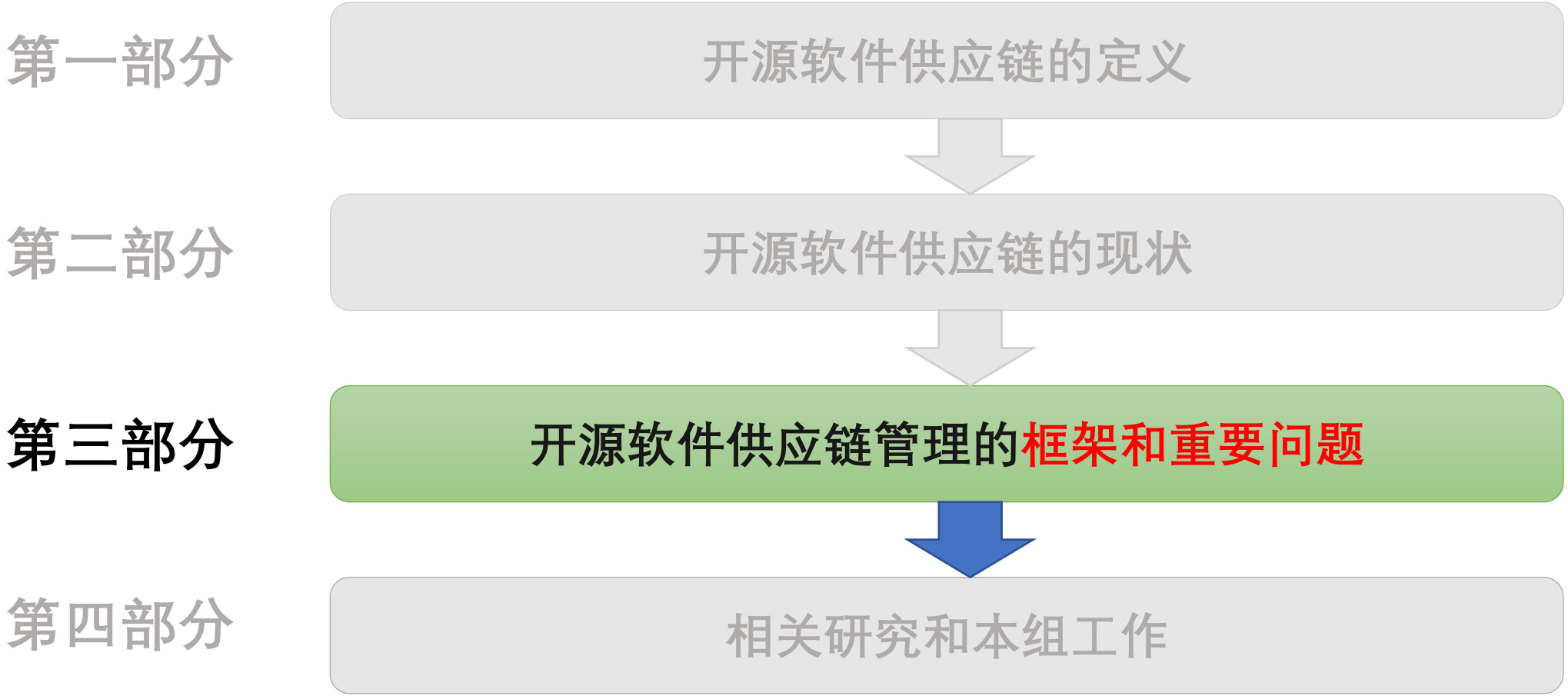
Decan, Alexandre, Tom Mens, and Philippe Grosjean. "An empirical comparison of dependency network evolution in seven software packaging ecosystems." *Empirical Software Engineering* 24.1 (2019): 381-416.

Zimmermann, Markus, et al. "Small world with high risks: A study of security threats in the npm ecosystem." *28th USENIX Security Symposium (USENIX Security 19)*. 2019.

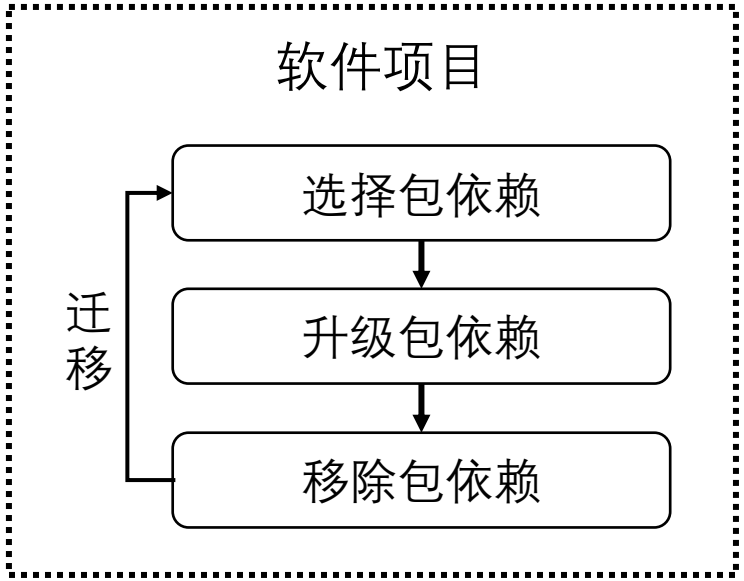


https://xkcd.com/2347/

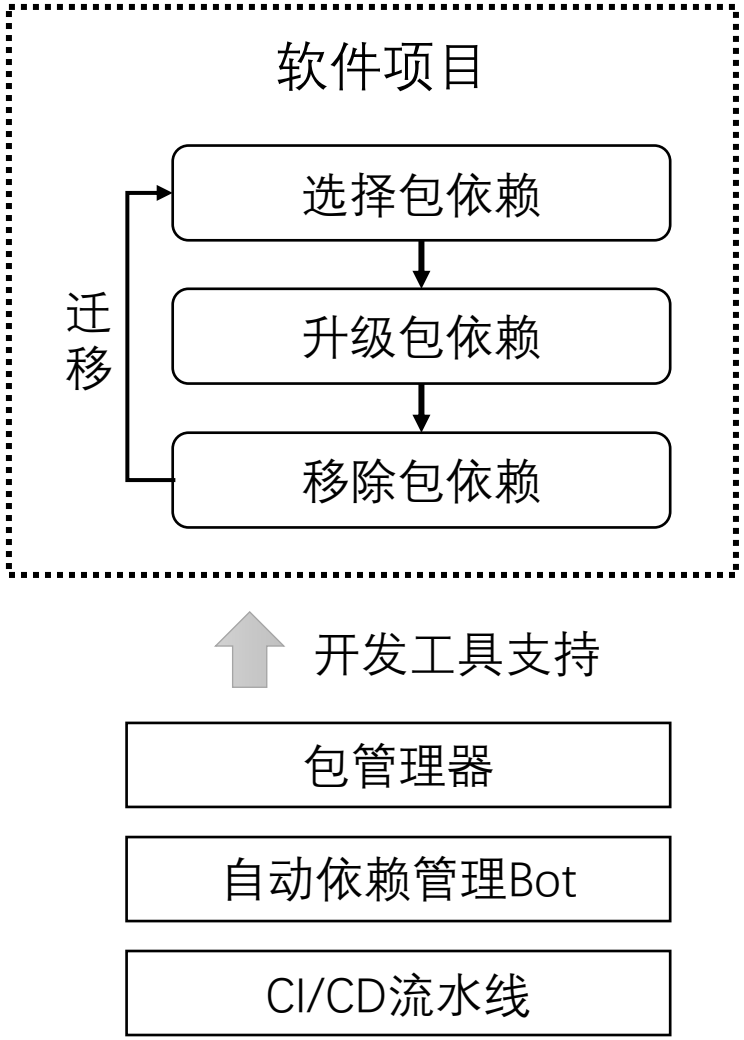
报告概览


第一部分
开源软件供应链的定义
第二部分
开源软件供应链的现状
第三部分
开源软件供应链管理的框架和重要问题
第四部分
相关研究和本组工作

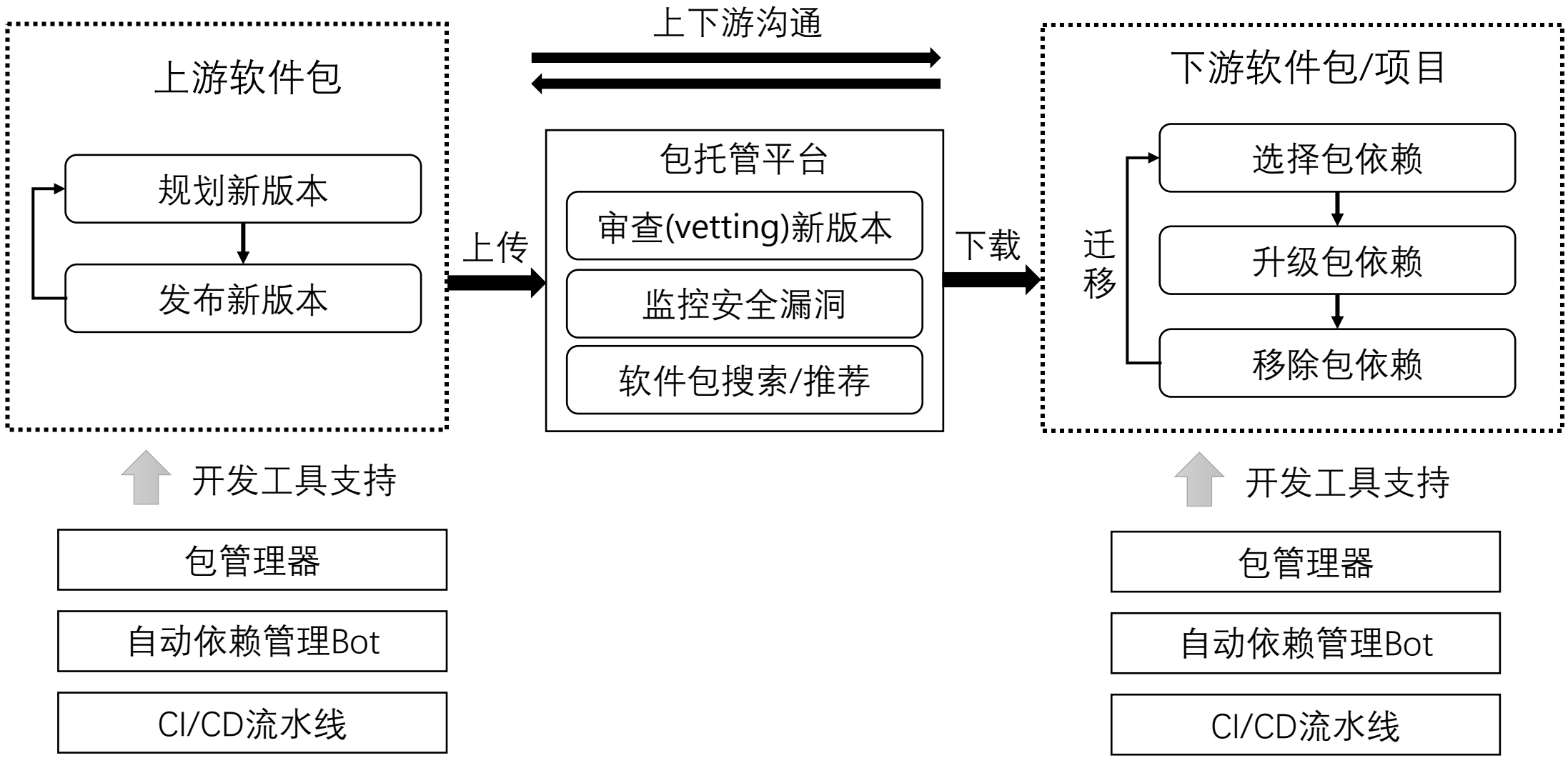
开源软件供应链管理



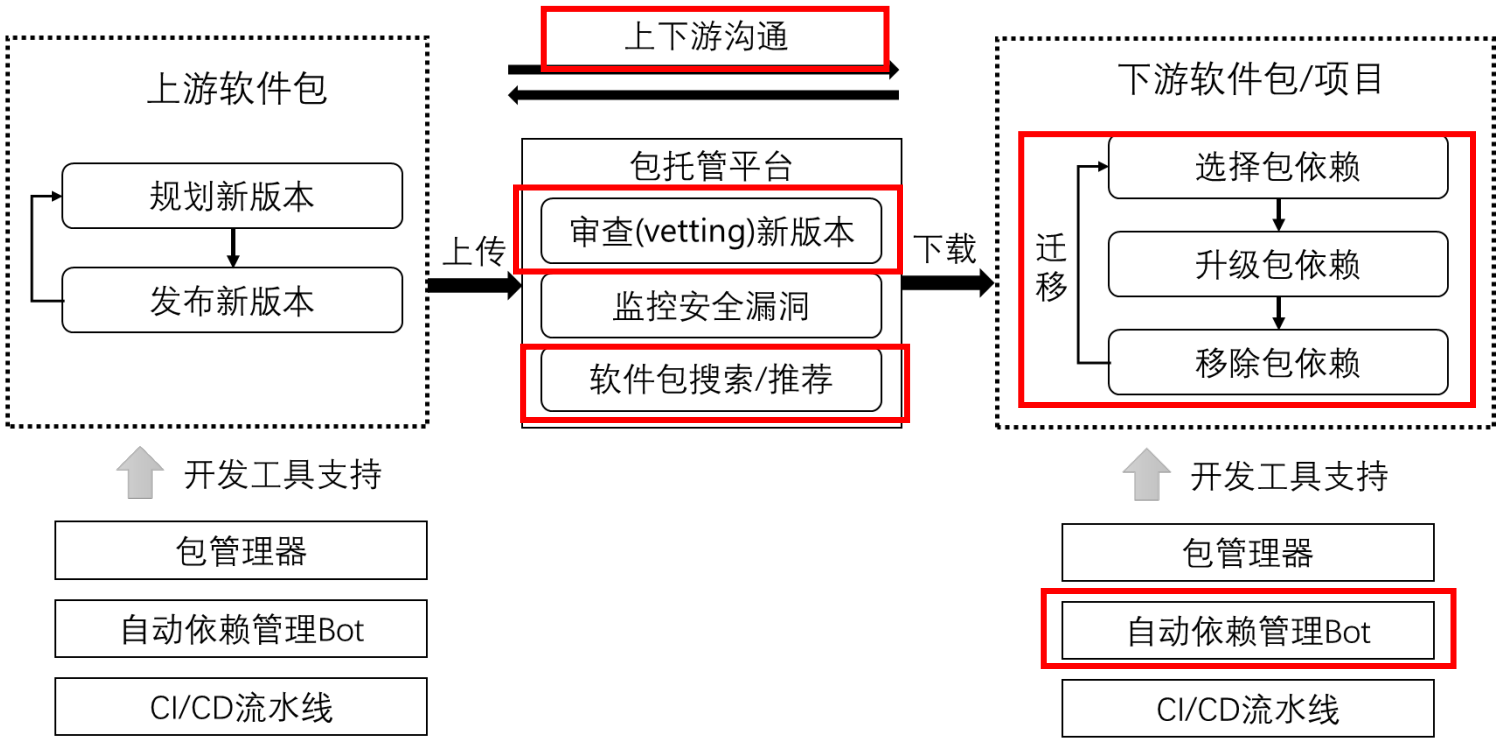
开源软件供应链管理



开源软件供应链管理


上游软件包
规划新版本
发布新版本
上下游沟通
上传
包托管平台
审查(vetting)新版本
监控安全漏洞
软件包搜索/推荐
下载
下游软件包/项目
迁移
选择包依赖
升级包依赖
移除包依赖
开发工具支持
包管理器
自动依赖管理Bot
CI/CD流水线
开发工具支持
包管理器
自动依赖管理Bot
CI/CD流水线

我认为有研究价值的若干问题



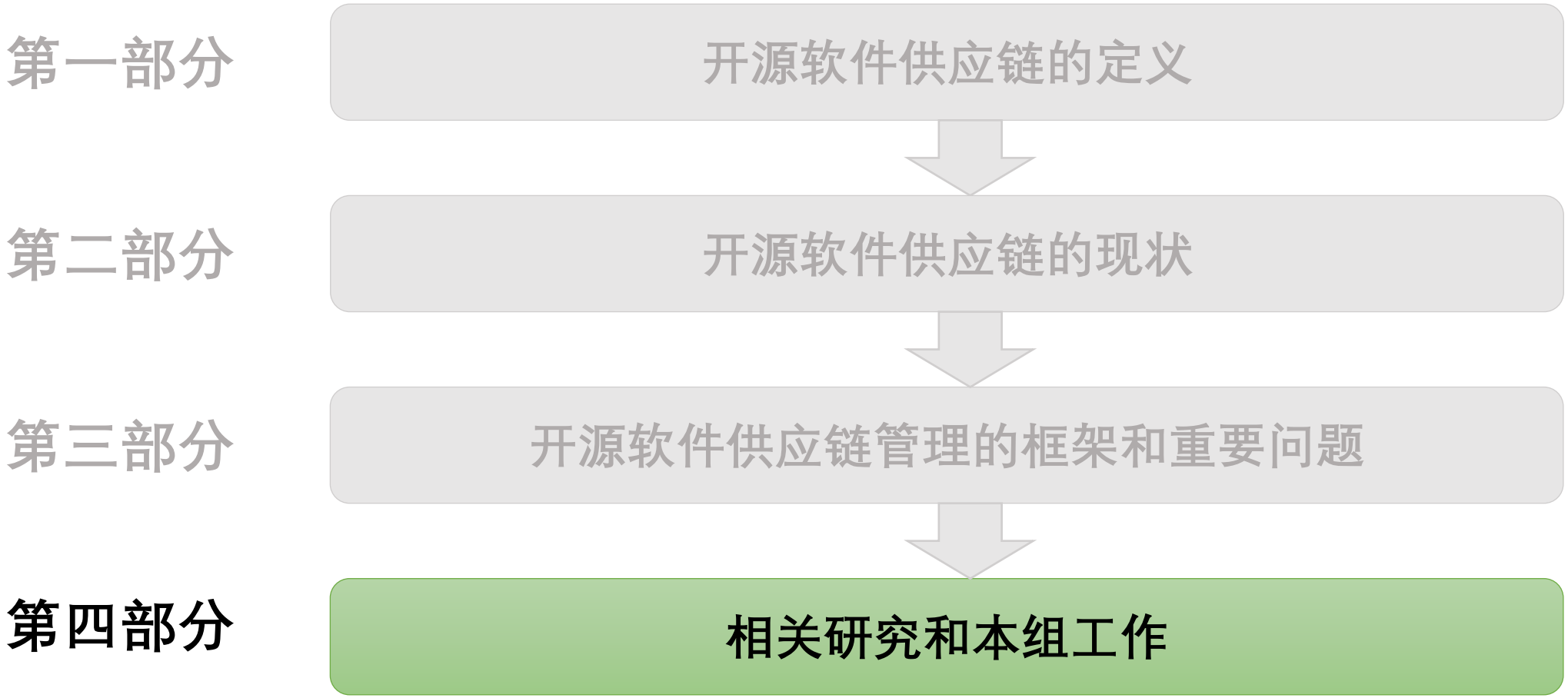
- 上下游沟通机制
  - Release Note
- 包管理平台
  - 确保新版本里没有恶意代码？
  - 构建机制推荐/选择软件包？
- 下游软件包/项目
  - 如何选择包依赖？
    - 理论
    - 推荐工具
  - 如何升级包依赖？
    - 检测不兼容变更
    - 适配不兼容变更
  - 对依赖的移除/迁移
    - 理论 & 推荐工具
- 自动依赖管理Bot
  - 交互设计
- 供应链&开源可持续性

相关研究涉及的研究方法

- Knowledge-Seeking Research
  - seek findings from observations (i.e., data)
  - empirical, positivism
    - data science
    - survey, interview, grounded theory, …
  - builds, extends, or verifies theories
    - design science (how to do sth)
    - co-variance (A correlates B / A causes B)
    - taxonomy / process theory
    - …
  - discover new hypotheses
    - solving problem A brings value B
    - solution C may solve problem A
    - …
  - findings + theories → knowledge
- Solution-Driven Research
  - propose & justifies problem A
  - propose solution B to solve problem A
    - algorithms
    - models
    - tools
    - systems
    - methods
    - …
  - evaluate B empirically
    - simulation
    - field study
    - …

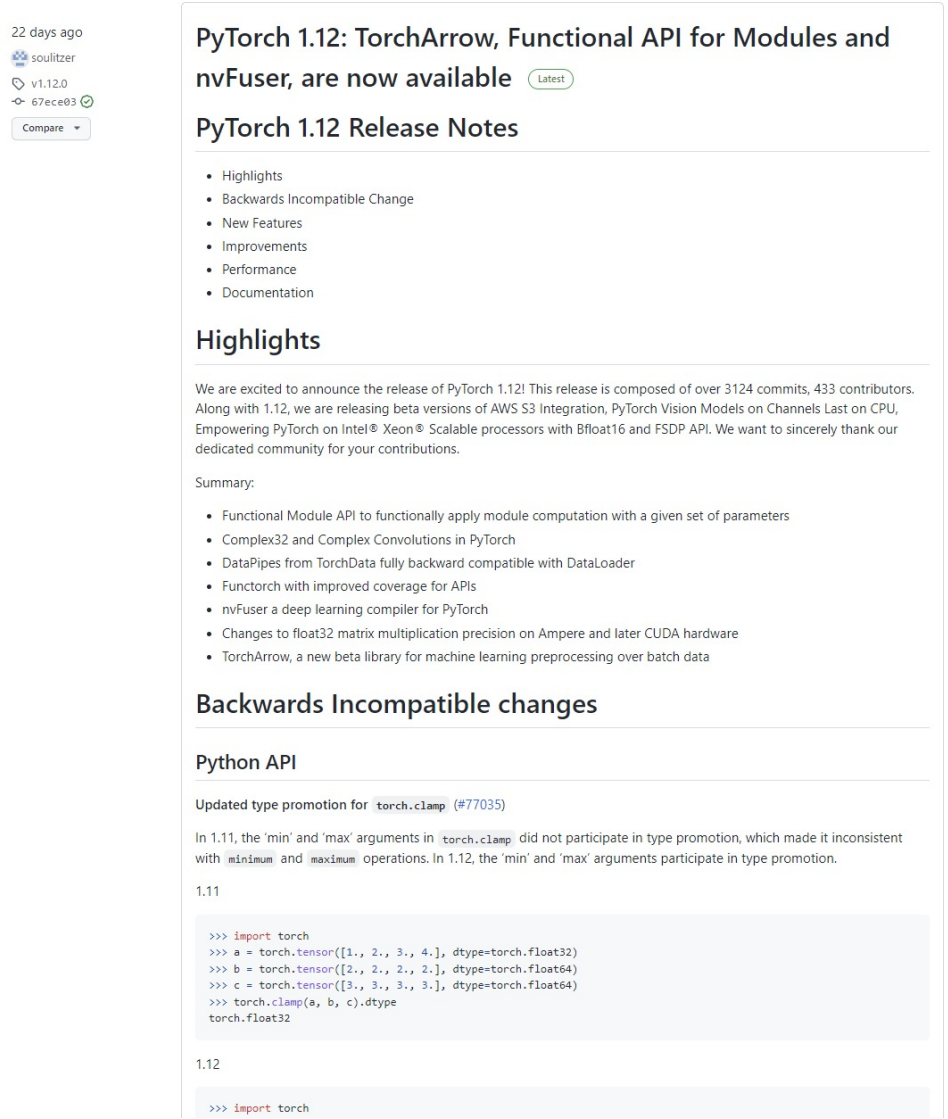
Open-Source Software Supply Chain Research:
knowledge => problem => solution => knowledge

报告概览


第一部分
开源软件供应链的定义
第二部分
开源软件供应链的现状
第三部分
开源软件供应链管理的框架和重要问题
第四部分
相关研究和本组工作

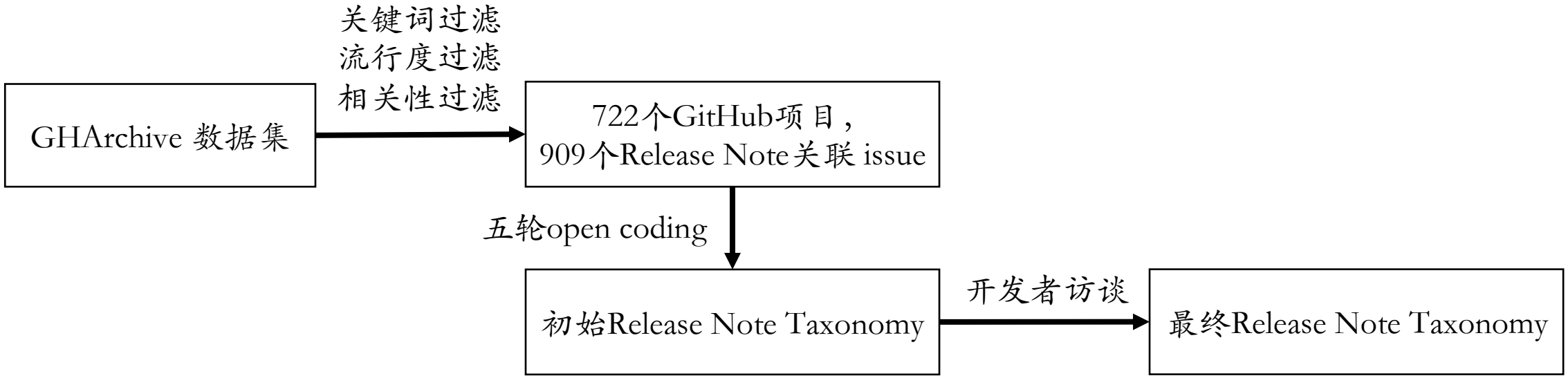
供应链上下游沟通的桥梁——Release Note

- **告诉下游开发者，软件包变更了什么**
- 已有研究主要关注Release Note内容分类
  - Abebe et al., EMSE 2016
  - Bi et al., TSE 2020
  - Yang et al., EMSE 2022
- Release Note当前实践存在哪些问题？
  - No empirical evidence
- 如何改进Release Note自动化工具？
  - ARENA (Morena et al., TSE 2017)
  - Semantic Release, Release Drafter, …
  - GitHub Release Note
    - https://docs.github.com/en/repositories/releasing-projects-on-github/automatically-generated-release-notes



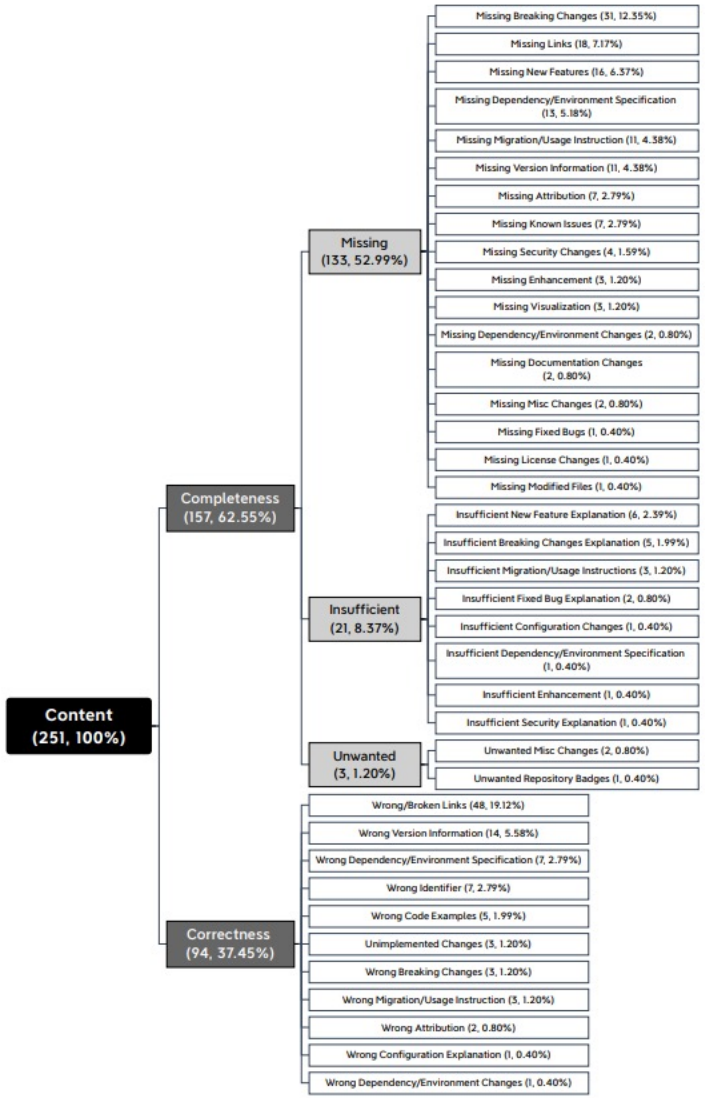
我们的研究：Release Note Issues on GitHub

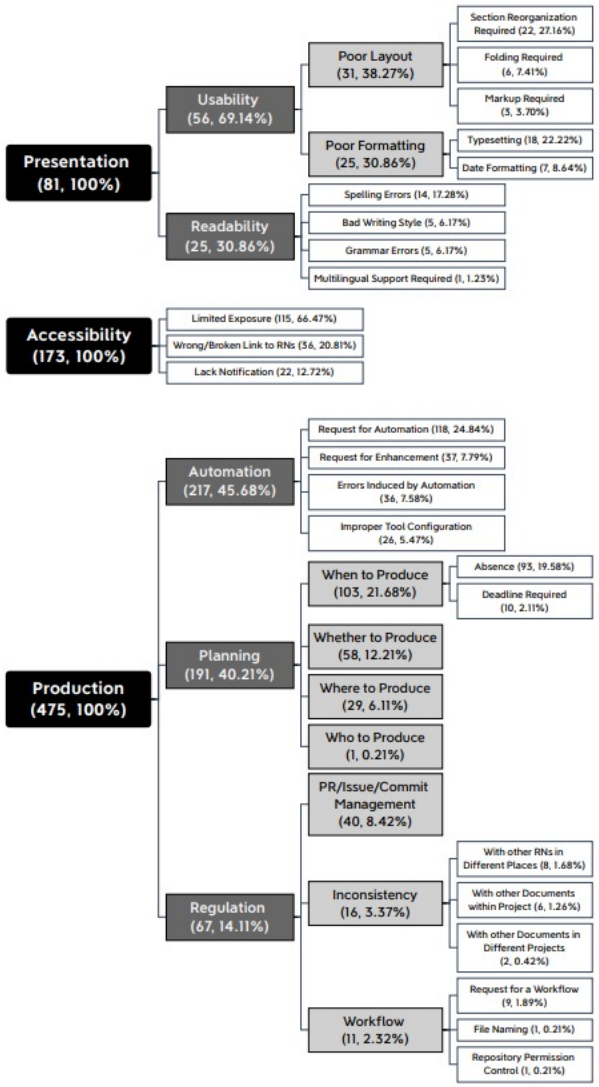
- **RQ:** What are the release note issues faced by developers on GitHub?
- **方法：**



Wu, Jianyu, et al. “Demystifying Software Release Note Issues on GitHub.” *2022 IEEE/ACM 30th International Conference on Program Comprehension (ICPC).* ***Distinguished Paper Award!***

我们的研究：Release Note Issues on GitHub





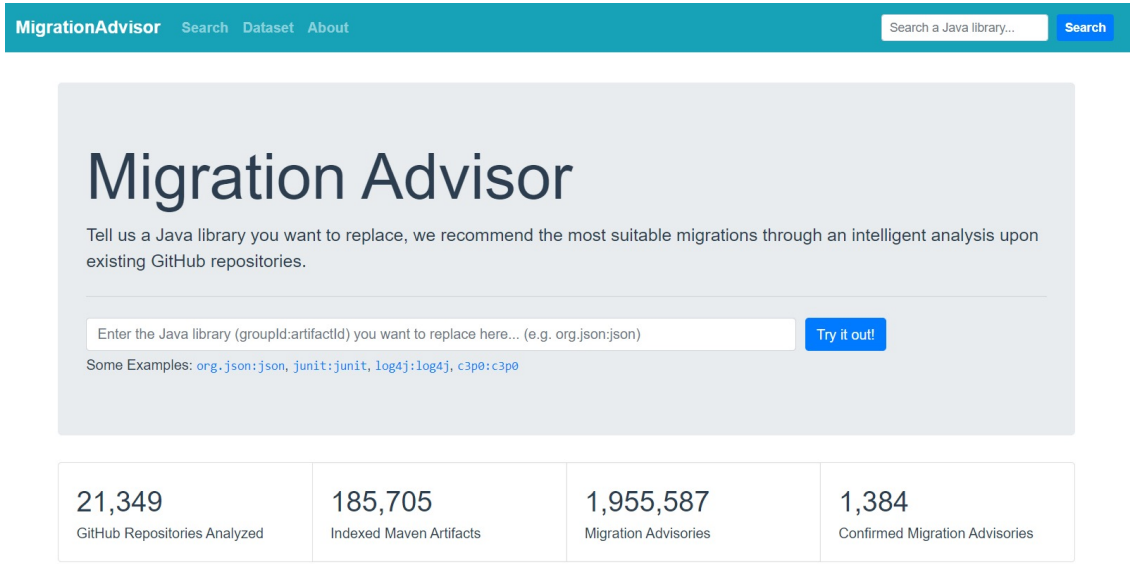
- **发现：**
  - 实证证明Release Note问题很常见
    - 遗漏、错误、风格...
  - 实证证明开发者需要自动生成
- **核心贡献一：**Release Note指南
  - https://hehao98.github.io/posts/2022/3/8/release-note
- **核心贡献二：**未来研究方向
  - Release Note生成
    - Commit/Software Change分类
    - Breaking Change检测
  - Release Note测试/Linting
    - 不一致
    - 语言风格

包依赖的选择 & 替换

- 常见的指标、例如流行度，并不能很好地反映库的质量
  - Mora and Nadi, PROMISE 2018
  - Zerouali et al. SANER 2019; Saini et al., JSEP 2020
- 开发者在选择软件包时需要考量大量技术和非技术因素，且需要开发经验
  - JavaScript Frameworks (Pano et al., EMSE 2018)
  - Reuse vs. reimplementation (Xu et al., EMSE 2020)
  - Industry practice in general (Larios Vargas et al., ESEC/FSE 2020)
- 软件包推荐？
  - **路线一：**基于已经被使用的包推荐其他包
    - Thung et al., WCRE 2013; Ouni et al., IST 2017; Nguyen et al., JSS 2020; …
  - **路线二：**无监督地对包进行理解
    - Chen and Xing, SANER 2016; Theeten et al., MSR 2019; …
- 软件包的选择和替换过程刻画？

我们的研究：Java项目的软件包替换过程 (i.e., 库迁移)

- **数据集：**半自动构建
  - 多指标挖掘 + 人工验证
- **发现一：**相似的软件包之间的替换很常见
  - 8% ~ 25%发生概率
  - 个别项目发生多次
- **发现二：**单向性
  - 从若干竞争者 => 汇聚到一个胜利者
- **发现三：**替换原因
  - 项目集成（~30%）
  - 软件包不被维护（~18%）
  - 可用性问题（~18%）
  - 缺乏功能（~13%）
  - 安全漏洞（~6.2%）



开源工具： http://migration-helper.net/#/

He, Hao, et al. "A multi-metric ranking approach for library migration recommendations." *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021.

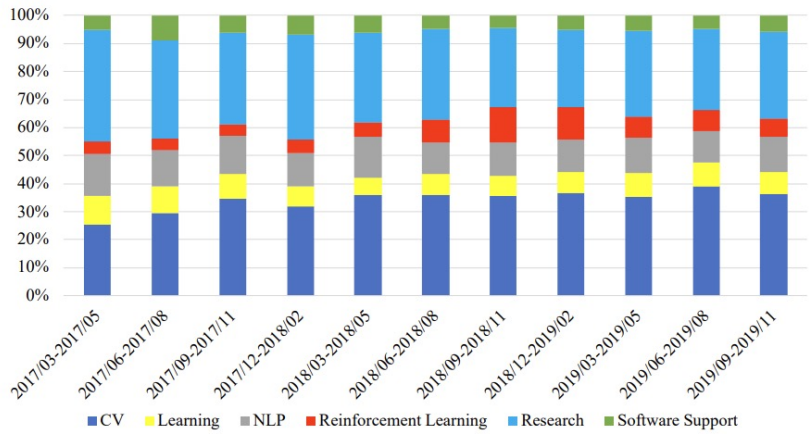
He, Hao, et al. "A large-scale empirical study on Java library migrations: prevalence, trends, and rationales." *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2021.

我们的研究：深度学习供应链初探

- **数据集：**World of Code

Algorithm 1: Constructing DL SC

```
Input: R_i ;                          // url of projects on the i-th layer
Output: DL SC
initialization: i ← 1 ;               // start from the lst layer
IN_i ← {} ;   // import names of the packages released by the
  projects on the i-th layer
R_i ← {url of TensorFlow or PyTorch};
while true do
    IN_i ← {} ;
    for r ∈ R_i do
        search Libraries.io to obtain the released package p;
        if p then
            manually label the import name of p: i_n_p;
            IN_i.insert(i_n_p);
    if IN_i is null then
        return ;                      // no packages are released
    else
        i ← i + 1, R_i ← {} ;                    // next layer
        for i_n_p ∈ IN_i do
            search WoC to obtain projects that import i_n_p: r;
            R_i.insert(r);
    if R_i is null then
        return ;                  // no downstream projects exist
```



- **发现：**
  - 深度学习供应链有5~6层
  - 存在复杂的领域趋势变化
    - 领域分布
    - 领域演化趋势
  - 影响包被选择的因素：
    - 更多包作者数量 => 更多人选择
    - 更多上游依赖数量 => 更少人选择

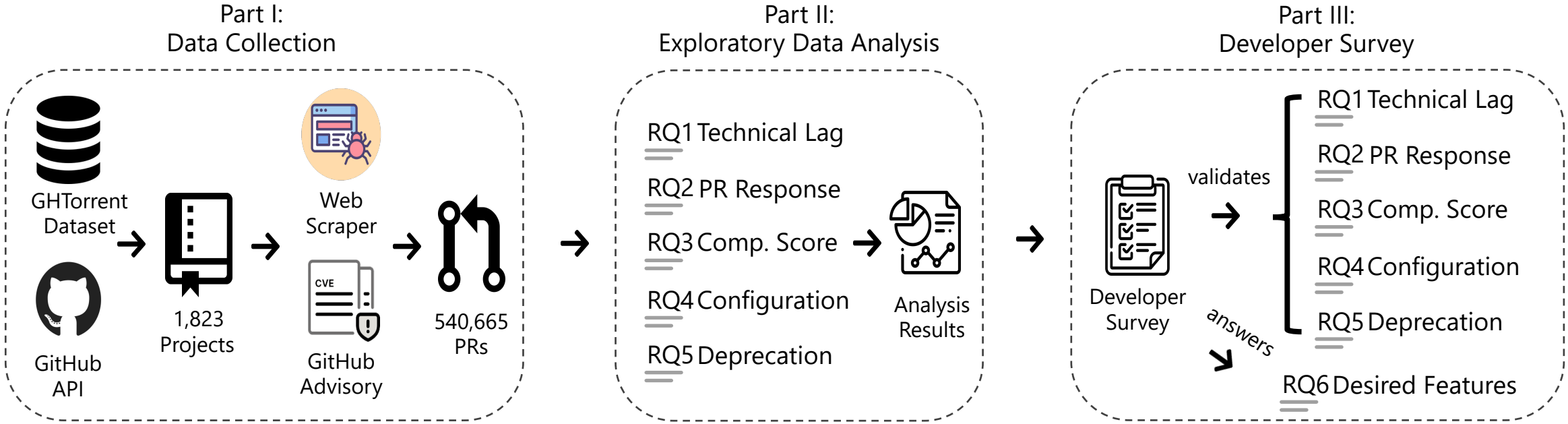
Tan, Xin, et al. "An exploratory study of deep learning supply chain." *Proceedings of the 44th International Conference on Software Engineering*. 2022.

包依赖的升级 & 版本管理

- **升级的好处：**
  - 新功能、bug修复、安全漏洞修复
  - 长期不升级带来的大量版本落后，最终会让项目在需要升级时，成本大幅上升乃至不可接受
    - Winters et al., Software Engineering at Google. O'Reilly Media
- **升级的阻碍：**
  - 额外的开发成本，额外的责任 (Kula et al., EMSE 2018)
  - 不兼容变更
  - 引入难以预测的隐藏缺陷 (Mostafa et al., ISSTA 2017)
- **解决方案：**
  - 编写完备的、高质量的测试
    - 也许并没有看上去的那么完备！(Hejderup and Gousios, JSS 2022)
  - 自动依赖升级Bot（Greenkeeper, Dependabot, Renovate Bot, Sync Bot, ...）
    - 存在交互性问题 (Mirhosseini and Parnin, ASE 2017 and Rombaut et al., TOSEM 2022)
  - 自动API适配 (Huang et al, ASE 2021; Nielsen et al., ICSE 2021, ...)

我们的研究：Dependabot的有效性

- **研究目标：**
  - 系统性地评估Dependabot用于包依赖升级的有效性
  - 提供有关依赖管理Bot应当如何设计的基础理论



He, Runzhi, et al. "Automating Dependency Updates in Practice: An Exploratory Study on GitHub Dependabot." *arXiv preprint arXiv:2206.07230* (2022).

我们的研究：Dependabot的有效性

- **结论：**
  - Dependabot有助于项目保持依赖最新
    - RDD模型显示结果相当显著
  - PR接受率高(~70%)，回复快(4 hrs.)
  - 已有机制**无法确保PR的兼容性**
    - 基于CI数据的Compatibility Score太稀疏
    - 开发者认为还是高质量的测试靠谱
  - 让Dependabot不那么**烦人**，有点难
    - 无法一个PR升级多个依赖
    - 无法自动merge
    - 包管理器兼容性
    - 配置正确的升级策略
  - 大量开发者迁移到Renovate Bot

How to Design a Dependency Management Bot?

Self-Adaptation

- Identify a sane default configuration based on language, package ecosystem, project history, etc.

Configurability

- Configurations to control noise
- Configurations to define precise update strategies

Transparency

- Breaking Changes
- Release Notes
- CI/CD Log Analysis

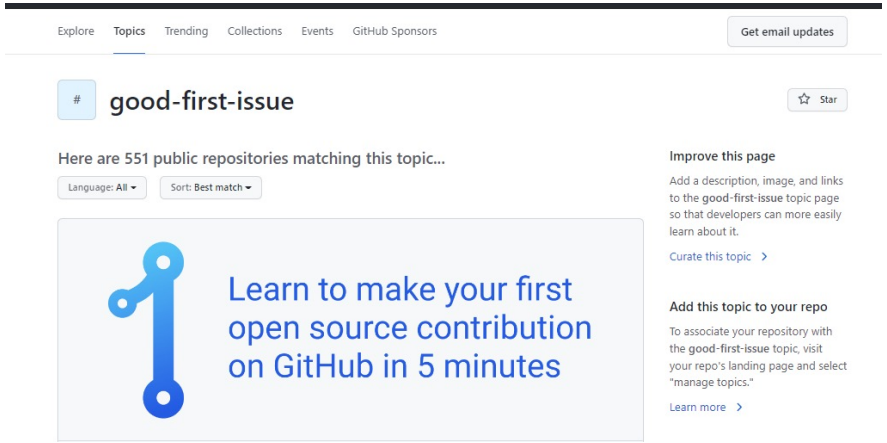
⇒ More fundamental research!

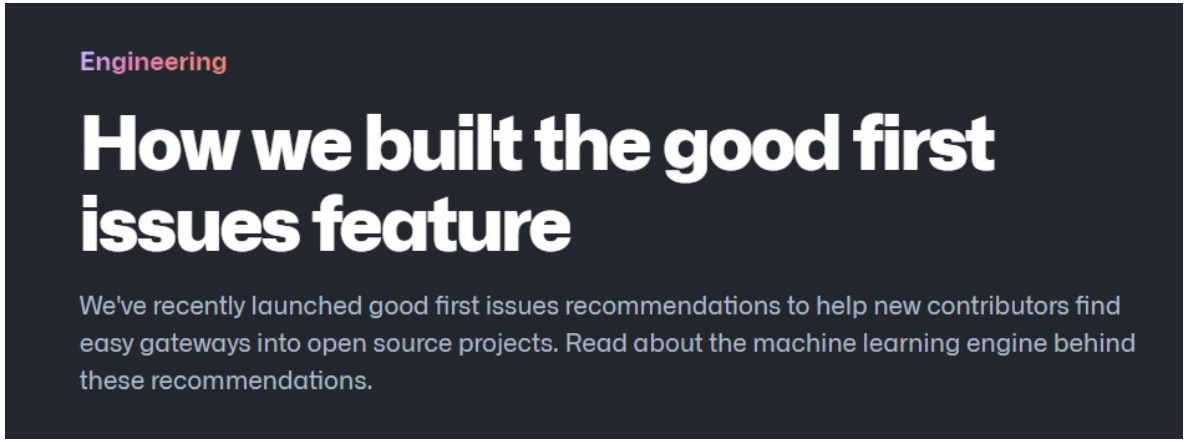
Autonomy

- Allow auto-merge in certain conditions
- The platform should do package vetting, not developers

供应链可持续性——吸引更多人加入开源

- **开源世界的法则：**如果你觉得XX不好，那你就去建设XX....
- 加入开源项目**很困难：**
  - 技术、社交、文档等多方面阻碍 (Steinmacher et al., IST 2015)
  - 熟悉软件项目需要复杂的过程 (Zhou and Mockus, ICSE 2010, FSE 2012)
  - 协调开发任务（Steinmacher et al., ICSE 2018）、mentoring (Balali et al., CSCW 2018)、...
- 如何解决？
  - 标记适合新人的开发任务（GitHub的Good First Issue机制）





我们的研究：Good First Issue机制探索

- 40.9% of GFIs that were not solved by newcomers and 31.2% of newcomers who failed to solve the GFIs after several attempts.
- A serious problem seems to be that GFIs are hard to attract long-term contributors, which makes GFIs less effective from the projects' perspectives.

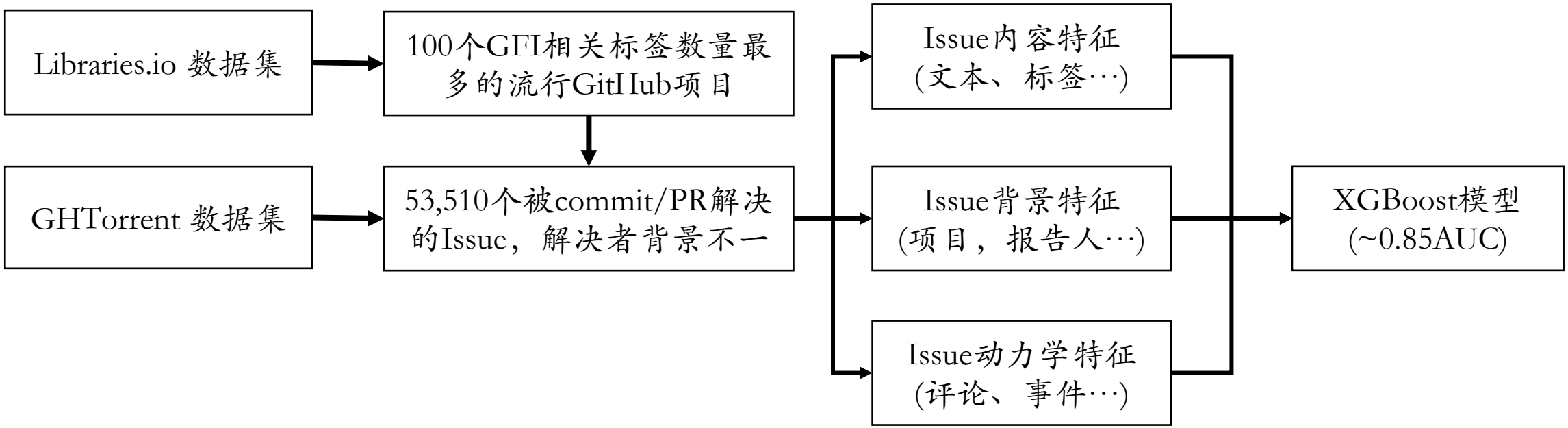
Table 3: Problems of the Mechanism of GFIs/ Why did newcomers fail to solve GFIs? (■: Newcomer ■: Project member)

Problems	Categories	Responses
Insufficient GFIs in the project	Project	11
Inappropriate GFIs	Mechanism	10
Lacking of motivations to retain	Newcomer	7
Hard to start on a new project	Newcomer	6
Few newcomers in the project	Project	5
Uneven level of newcomers' skill	Newcomer	5
Getting snapped up soon	Mechanism	3
Lacking of mentors	Mechanism	2

Tan, Xin, Minghui Zhou, and Zeyu Sun. "A first look at good first issues on GitHub." *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020.

我们的研究：Good First Issue推荐

- **问题**：从历史数据中学习Good First Issue的特征并对未解决issue进行预测
- **贡献**：数据集 + 特征工程 + 可行性验证



Xiao, Wenxin, et al. "Recommending good first issues in GitHub OSS projects." *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*. IEEE, 2022.

我们的研究：Good First Issue推荐

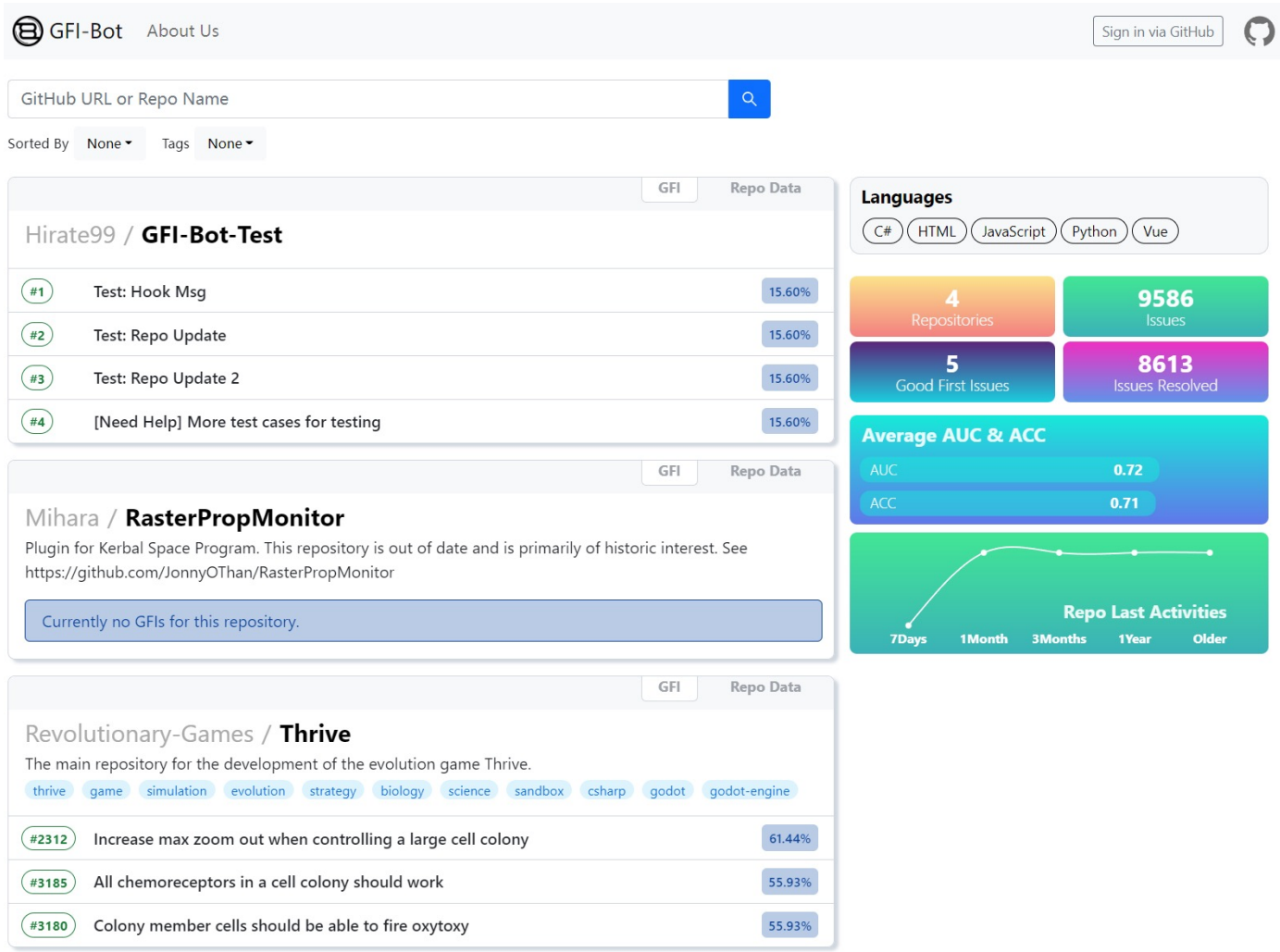
sigveio commented on 24 Aug 2021 Contributor

Marking more issues as GFI does have the potential of making projects more accessible to new contributors, so I'm intrigued!

What would you consider the base criteria for a GFI to be? Does it for example include work to triage new issues, such as determining if a bug is actually a bug, or are we talking about actionable issues in the context of producing a pull request that would be accepted / merged into the codebase?

bytrangle commented on 23 Sep 2021 • edited

I enjoy this discussion. Whenever I think of contributing to an open-source project for the first time, I spend an inordinate amount of time sifting through issues to find the most newcomer-friendly one.



https://github.com/osslab-pku/gfi-bot

未来研究方向

- Release Note生成 => 帮助供应链上下游协作、库依赖升级
  - 与已有的Semantic Release等工具结合
  - commit分类、breaking change检测、...

- 吸引更多人加入开源 => 提升供应链可持续性
  - 个性化Good First Issue推荐
  - 什么样的Good First Issue解决者更有可能成为长期贡献者？

- 相关机制理解 & 研究辅助
  - 软件包相互竞争和开发者做替换的通行规律？
  - 如何将软件包链接回GitHub仓库？

实验室宣传

- **导师：周明辉 教授**
  - zhmh@pku.edu.cn
  - https://minghuizhou.github.io/
- **方向：**开源软件、
- **实验室网站**
  - https://osslab-pku.github.io/
- **GitHub组织**
  - https://github.com/osslab-pku
- 常年招收本科实习生
  - 近两年有**六位**本科生有论文发表
  - **一作** ~ 四作均有
- 有2024年入学的博士名额

谢谢！欢迎提问