软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(8):3035−3058 [doi: 10.13328/j.cnki.jos.006374]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

开源许可证合规性研究*

王志强, 伍 胜, 肖国强, 张自力, 刘志有, 彭 景

(西南大学 计算机与信息科学学院/软件学院, 重庆 400715)

通信作者: 肖国强, E-mail: gqxiao@swu.edu.cn

摘 要: 随着开源概念的逐步深入, 开源软件成为软件发展的潮流. 同时, 开源软件的使用受各类开源许可证约束. 开源参与者在开发过程中该如何为自己的开源软件选择合适的许可证, 确保高效合理地使用社区群体智慧劳动成果, 仍是一个亟需解决的问题. 为此, 首先分析和解读了开放源代码促进会认证的常用开源许可证, 通过对许可证条款内容和结构的研究, 得到开源许可证框架及许可证兼容性推导模型, 并将该模型应用于对我国自主研发的木兰宽松许可证的分析和解读. 最后, 基于上述工作研发了开源许可证选择工具, 为开源开发者对许可证的理解和合规使用提供了参考和决策支持.

关键词: 开源软件; 开源许可证; 合规使用; 兼容性; 许可证选择

中图法分类号: TP311

中文引用格式: 王志强, 伍胜, 肖国强, 张自力, 刘志有, 彭景. 开源许可证合规性研究. 软件学报, 2022, 33(8): 3035−3058. http://www.jos.org.cn/1000-9825/6374.htm

英文引用格式: Wang ZQ, Wu S, Xiao GQ, Zhang ZL, Liu ZY, Peng J. Study of Open-source Software License Compliance. Ruan Jian Xue Bao/Journal of Software, 2022, 33(8): 3035−3058 (in Chinese). http://www.jos.org.cn/1000- 9825/6374.htm

Study of Open-source Software License Compliance

WANG Zhi-Qiang, WU Sheng, XIAO Guo-Qiang, ZHANG Zi-Li, LIU Zhi-You, PENG Jing

(College of Computer and Information Science/College of Software, Southwest University, Chongqing 400715, China)

Abstract: With the progresses of open source concept, open source software has become the trend of software development, and the use of open source software is subject to various open source licenses. How open source participants can correctly choose open source software licenses in their development to ensure the efficient and reasonable use of the collaborative results of community groups is still an urgent issue to be solved. To this end, commonly used open source licenses are firstly analyzed and interpreted for OSI certification in the paper. Furthermore, with the studies of the license terms and structure, the open source license framework, and compatibility derivation models are deduced. The model is applied to the analysis and interpretation of Mulan permissive software license independently developed in China. Finally, based on the above work, a license choosing tool for open source license is developed, which provides references and decision support for open source developers to understand and use licenses.

Key words: open source software; open source license; appropriate use; compatibility; choose license

开源软件(open-source software, OSS)是指用户可以获取源代码的计算机软件, 允许用户免费使用、复制、修改和分发以提高软件质量[1]. 云计算、大数据、移动互联这些支撑互联网发展的技术都是基于开源软件构建的, 开源软件的开发是社区成员协同合作的成果, 这种开发模式极大地促进了软件行业的发展[1,2]. 根据Red Hat 2020 年企业开源现状的调查报告《企业开源现状》显示, 95%的受访者认为开源对企业未来的发展和创新至关重要. 数据显示, 基于社区的开源也在不断地增长, 较前一年增长了 3%, 已达到 19%. 同时, 我国也在大力发展开源社区, 并不断培养开源软件人才, 以发展并完善开源生态[3].

* 基金项目: 国家重点研发计划(2018YFB1004201)

收稿时间: 2020-02-01; 修改时间: 2020-12-06, 2021-03-03; 采用时间: 2021-05-09; jos 在线出版时间: 2021-05-20

开源软件协作开发的方式打破了传统软件开发的封闭模式, 但也带来了新的问题和挑战. 为了更好地保护开源软件的知识产权不受侵害, 并规范开发者对开源软件的使用, 开源许可证应运而生. 开源许可证以法律的形式对受版权法保护的开源软件的使用、复制、修改和分发行为进行规范[4]. 不同的开源许可证在原作品及衍生作品是否必须要按照原许可证发布、是否必须公开作品源代码、是否授予专利权等方面有着不同的规定. 经 Black Duck KnowledgeBase 统计得到包含在开源软件中已存在的相关许可证已超过 2 600 份, 但经 OSI 批准的许可证仅有 121 份. 2020 年 2 月, 我国产业界自主研发的木兰系列许可证之“木兰宽松许可证, 第 2 版”(http://license.coscl.org.cn/MulanPSL2/)正式被 OSI 批准认证为“国际类别开源许可证(Int'l Licenses)”. 这是全球首个、也是目前唯一一个由中国研制并通过 OSI 认证的中英文双语国际开源许可证. 此外, 2021 年 5 月, 木兰系列许可证之“木兰公共许可证, 第 2 版”也已在中国开源云联盟官网正式上线.

开源许可证是开展开源软件开发和技术创新的基石, 不仅能够增强开源项目传播的规范性, 而且可以为软件提供保护, 防止他人无节制的滥用自己的代码[5,6]. 但很多开发人员开源意识比较淡薄, 没有意识到在自己的代码中添加特定许可(或不添加任何许可), 或错误组合不兼容许可证的开源代码可能会遇到的后果[7]. 在软件开发日益模块化、组件化和平台化的趋势下, 软件产品研发不可避免要引入第三方依赖库进行开发. 例如, Black Duck Software 公司 2019 年审查的 17 个行业 1 253 个商业代码库中使用开源组件的比例高达 99%, 其中有 9 个行业开源组件的覆盖率为 100%. 此外, 通过分析源代码中许可证使用情况, 发现开源组件之间存在开源许可证冲突的情况占 67%, 并且开源代码库中包含未经许可的软件占 33%. 通过以上统计数据可知: 越来越多的企业与开源开发者基于开源代码进行二次开发, 同时, 不同许可证开源组件的违规使用情况也不容乐观. 当开源组件所使用的许可证与整个项目所使用的许可证条款相互冲突时, 将会存在许可证兼容性问题, 从而导致开源软件的违规使用. 因此, 在鼓励开源生态建设的同时, 还应该加强开源开发者对开源许可证的正确理解和选择. 仔细考虑每份许可证的特殊性, 努力为开源软件的发展创造良好的生态环境.

为了帮助开发者能够合规使用开源许可证, 越来越多的研究者与实践者关注到开源软件开发过程中开源许可证合规使用的问题. 例如, Kapitsaki 等人[7]利用 FOSSology 检测 SPDX 软件数据包中涉及或使用的开源许可证, 并基于创建的许可证兼容性模型推断是否存在违规情况, 从而实现了许可证兼容性自动分析工具. Fendt 等人[8]从使用开源工具和相关流程的角度展开研究, 说明了规范开源许可证使用的重要性. Gamalielsson 等人[9]分析了 200 个开源软件项目的许可条款和附加贡献条件之间的关系. 另外一些研究工作[10−13]探索了许可证兼容性判断及许可证检测等辅助工具的开发对开源软件健康发展的重要性. 例如, 许洪波等人[14]在现有的许可证检测工具的基础之上, 研发了开源许可证冲突自动检测原型系统. Vendome 等人[15]利用机器学习方法对许可证异常情况进行了分析和检测, 并给出对应的解释. 此外, 还有一部分研究人员致力于许可证变更的研究[16]. 现有的研究工作主要通过检测开源软件中是否存在许可证违规情况来规范开源软件的使用, 缺乏从许可证文本内容及如何帮助开发者正确选择许可证等角度展开研究. 针对上述问题, 本文以常用开源许可证及国内自主研发的木兰系列许可证为研究对象, 分析并创建了许可证框架及兼容性推导模型, 进而研发了许可证选择工具. 最终得到的许可证框架、兼容性推导模型及许可证选择工具, 可以很好地指导开源开发者、企业及开源社区正确使用开源软件, 帮助开源软件从业者和企业正确选择合适的许可证发布自己的开源软件.

本文第 1 节对开源软件研究现状进行概述, 并以开源许可证违规使用案例为切入点, 对开源许可证使用过程中存在的问题进行归纳和分析, 明确开源许可证合规使用的重要性. 第 2 节通过对许可证文本内容进行归纳, 得到许可证框架, 以帮助开发者梳理许可证内容. 第 3 节从许可证兼容性关系出发, 得出兼容性推导模型, 并将该兼容性推导模型应用于木兰宽松许可证兼容性分析. 第 4 节主要在已有成果的基础上研发开源许可证选择工具, 以进一步指导和规范开发者合规使用开源许可证. 第 5 节分析本文的局限性. 最后总结全文并展望未来的研究工作.

1 相关工作

开源运动起源于自由软件创始人 Richard Stallman 为对抗私有软件的盛行而倡导的自由软件运动[17]. 1985 年, 由 Richard Stallman 创建的自由软件基金会(the Free Software Foundation, FSF)得到了业界的广泛响应. 1997 年, Eric Raymond 的著作“The Cathedral and the Bazaar”[18]明确区分了自由软件与商业软件, 论证了开源软件的盛行将是软件发展的必然趋势. 1998 年, “Open Source”正式用于开源软件并设立了开源计划组织[19]. 同期, 我国也积极投身于开源软件开发的行列之中. 在政府的大力支持下, 越来越多的企业参与到开源建设中, 成立了许多优秀的开源软件社区, 并研发了红旗 Linux 等来促进开源软件的发展. 到 2020 年 12 月为止, 阿里巴巴的大型开源软件总数多达 191 个, 腾讯的大型开源软件总数多达 159 个, 百度的大型开源项目总数多达 118 个, 等等. 同时, 截至 2020 年 11 月, 统计得到国内开源项目托管平台 Gitee 上已有近 17 万个开源项目率先支持木兰宽松许可证, 项目类型涵盖云计算、大数据、人工智能、开发框架等多个应用软件, 以及操作系统、数据库等基础软件和中间件.

近年来, 开源软件的创新发展已经形成全球化趋势, 在软件的开发及应用等领域发挥着重要的作用[20]. 同时, 开源许可证的种类也在不断增加, 种类繁多的开源许可证给开源软件的使用带来了巨大的挑战. 如开源软件开发过程中开源许可证的合规使用、选择、检测以及兼容性等问题. 因此, 开源开发者和企业在许可证选择以及使用开源软件二次开发的过程中要特别注意不同许可证之间的差异性, 严格遵从不同许可证要求进行组合和使用. 为了帮助开发者对常见的许可证违规使用情况有更加全面的认识, 本文根据各类违反许可证案例中违规原因的差异性, 将违规情况归纳为以下两类情况.

(1) 法律层面

- 开源许可证保护软件知识产权和合规使用. 软件知识产权的保护对软件行业的发展有着重要的影响, 而开源软件的发展增加了软件知识产权侵权的发生[21]. 例如, 谷歌与甲骨文案件是开源软件许可证中最重要的版权侵权案例之一, 该案件主要围绕 JAVA API(应用程序接口)名称的版权是否构成侵权而展开. 2010 年 8 月, 甲骨文公司控诉谷歌公司在 Android 系统使用的 37 个 JAVA API, 侵犯了其著作权, 于是向谷歌公司提起诉讼. 经过多次上诉之后, 于 2018 年 3 月, 法院宣判谷歌侵权事实成立, 要求谷歌就侵权行为赔偿甲骨文公司近 90 亿美元. 但谷歌公司反对这一判决, 并联合众多拥抱开源的公司及开发人员再次向联邦最高法院提起诉讼. 终于在 2021 年 4 月 5 日迎来最终判决, 最高法院认为, Android 系统对 Java API 代码的复制属于合理使用, 不构成侵权;
- 许可证不会因地区差异而失效. 开源软件的发展是一项国际化的运动, 各国法律具有一定的独立性和差异性, 但许可证不会因地域问题而失去其保护开源软件的执行力. 例如, 2017 年 4 月, Artifex Software 公司上诉由韩国 Hancom 公司开发的字处理软件中集成了 Artifex 公司采用双重许可模式发布的开源软件 Ghostscript. 但是 Hancom 公司既没有遵守 Ghostscript 的 AGPL 许可证进行开源, 也没有为该软件购买商业许可证, 因此, Artifex 公司向美国地区法院发起诉讼. 但是 Hancom 认为自己是美国以外的韩国公司, 而且涉案行为并非在美国境内进行, 因此不能根据美国联邦法律来索求赔偿. 但被法院驳回, 最终判定韩国公司违反 AGPL 许可证条款.

(2) 软件层面

- 不遵守许可证条款任意组合开源代码. 开源软件的复制、修改、分发受开源许可证的保护, 如果开发者或企业在商业开发中使用到开源代码, 则必须遵守对应许可证的条款, 否则将会是违法. 例如, Skype WiFi 电话使用了基于 GPLv2 许可证的代码, 但却没有按照 GPLv2 许可证的要求开源代码, 也没有基于 GPLv2 分发作品, 最终因违反 GPLv2 许可证条款而被起诉. 此外, 2019 年, 我国 GPL 第一案“柚子案”中, 柚子公司在开发中使用了数字天堂公司的 HBuilder 软件中 3 个可独立运行的插件, 柚子公司认为这 3 个插件作为基于 GPL 许可证分发程序的一部分, 也应该受 GPL 许可证限制, 接收者可以自由使用并创造衍生作品而无需软件所有者单独授权. 然而法院判定: 插件存放在独立的文件夹中并且不含 GPL 许可证文件, 因此不属于 GPL 衍生作品. 最终判决柚子公司侵犯行为成立, 要求

其停止侵权并赔偿数字天堂公司;

- 未按照开源组件中双重许可情况执行. 双重许可证的前提是代码的所有版权归软件开发的公司所有, 可以采用开源许可证免费发布源代码, 也可以采用商业授权发布[22]. 双重许可模式是指接收者可以通过向版权所有者支付一定的费用来获得该软件的版权以供自己私有化使用, 同时, 该作品作为开源软件依然对接收者免费但必须遵循相应的开源许可证条款. 双重许可证可以解决许可证兼容性问题, 并为开源软件开发带来一定的收入, 但是也会存在很多潜在的风险[23]. 比如, MySQL 免费版是基于 GPL 许可证发布, 但如果接收者不想受 GPL 许可证的限制, 可以购买其商业授权许可, 这样将不必发布修改后的源代码. 但是接收者既不按照 GPL 许可证分发也不购买商业授权, 而将作品基于其他许可证分发的行为是禁止的.

通过许可证违规案例的分析可以发现: 开源许可证的合规使用不仅要注重对许可证条款的充分理解, 还应该关注许可证兼容性、许可证选择等问题. 目前, 有越来越多的研究者与实践者关注到这些问题并展开了充分研究, 其成果涵盖了法律领域、开源领域和软件工程领域等. 特别地, 法律层面主要关注开源许可证中与软件相关的知识产权遵守情况, 及从法律的角度分析开源许可证中知识产权的具体法律效力和法规维权等问题[24]. 例如, Hammouda 等人[25]以已有的开源组合模式及许可证本身相关的法律效力为基础, 引入了开源合法性模式, 以确保组合代码能够遵守所有许可条款. 张平等人[26]研究了开源软件与商业软件结合时可能会遇到哪些潜在的风险. 在开源和软件工程领域, 开源软件与私有软件的组合、不同许可证开源组件的组合变的越来越普遍, 同时也常常伴随着开源许可证合规使用问题. 研究者主要从 3 个方面解决这类问题.

(1) 一部分研究者致力于软件项目中包含的开源许可证的检测[27,28]. 例如, Kechagia 等人[29]通过研究开源软件包之间的依赖关系, 验证了许可证的受欢迎程度与包的组合关系密切相关, 该结论可以指导兼容性关系的研究. German 等人[30]利用一种句子匹配算法设计实现了自动识别开源代码中所涵盖的许可证检测工具 Ninka, 长久以来得到了广泛的使用. Jaeger 等人[31]开发的 FOSSology 工具利用 bSAM 算法实现开源软件中许可证的检测与匹配. 此外, Kapitsaki 等人[32]还提出, 利用 LDA 主题提取模型自动识别开源软件许可证中包含的条款, 以自动化地表示许可证. Feng 等人[33]研究设计了一种可扩展的、全自动的二进制软件开源许可违规检查系统. Zhang 等人[34]研究了主流的 4 种开源软件扫描工具, 通过对比这几种开源软件扫描工具的优缺点, 从而帮助用户在开源项目时选择合适的检测工具来分析开源许可证合规使用情况;

(2) 一部分研究者关注许可证兼容性问题. 例如, Monden 等人[35]通过探索代码重用是否存在的指标, 判断开源软件中包含的克隆或复制代码中是否存在违反许可证的情况, 从而指导和规范开发者代码整合. Kapitsaki 等人[36]以开源软件库中最常用的几种许可证为研究对象, 在 Wheeler[37]研究成果的基础上, 以图形方法可视化许可证之间的兼容性关系, 并研发了许可证兼容性推理模型. 此外, 通过双重许可的方式也能有效地缓解许可证兼容性问题. 例如, Margaret[38]分析了双重许可对于开源软件与商业软件的合理组合提供了机遇;

(3) 另外一部分研究者致力于许可证建模. 例如, Alspaugh 等人[39]研究了 10 份开源许可证, 并引入了一个包括权利和义务的许可模型. 在 Kapitsaki 等人的最新研究[13,40]中, 首先通过分析 33 份许可证创建了包含 38 个通用许可证条款的许可证框架, 然后研发了基于推荐算法的许可证推荐工具. Zahoor 等人[41]利用自然语言方法自动提取开源许可证中的标语, 以方便开发者快速了解许可证. German[42]等人提供了一个通用的文档式的许可证信息及许可证冲突表示模型, 但没有将其进行概念化的表示. Gordon 等人[43]基于语义 Web 技术和前期的理论和实践工作开发了一个开源许可模型, 并应用于 8 份流行的开源许可证.

本文在现有工作的基础之上, 按照许可证的解读、兼容性分析、许可证选择的顺序展开研究, 以帮助开发者理解和合规使用开源许可证. 本文结合相关文献及许可证文本等基础知识, 对软件托管平台 Github (github.com)的开源项目、企业(阿里、华为等)开源项目中最常用的十几种许可证及国内自主研发的木兰宽松

许可证展开研究. 在深入解读许可证的基础上构建了许可证框架, 并进一步分析了不同许可证之间及同一许可证不同版本之间的兼容性关系, 进而提炼出更加完善、通用的兼容性规则及兼容性推导模型. 最后, 基于以上研究进一步开发了许可证选择工具, 方便开发者根据自身需求选择合适的许可证. 本文的研究成果为开源开发者和企业在开源软件的使用、发布过程中开源许可证的理解及选择提供参考.

2 开源许可证解读和框架构建

2.1 开源许可证的解读

开源许可证在版权法保护下对开源软件的使用、修改或分发行为进行规范, 其基本目的是拒绝任何人独享作品[44]. 开发者需要将作品中所包含的所有版权开放给任意用户, 这样将有助于吸引更多的开发者投入到作品的开发之中, 极大地提高作品质量[45]. 本文通过从 OSChina 国产开源板块及开源公司板块收集和汇总了国内外 27 家开源企业的 638 个开源项目, 并将统计数据在 Github 上进行了开源(https://github.com/TheBigCock/license_choose). 通过对这些项目中开源许可证使用情况的分析以及 OSChina 开源社区中主流许可证使用情况的分析, 我们得出了在众多开源许可证中使用最广泛的开源许可证. 最终结果如图 1 所示, 包括 Apache v2.0, MIT, BSD, MPL, AGPL, GPL, LGPL, EPL, ZPL, zlib/libpng 许可证等. 因此, 本文将这些常用开源许可证作为主要的研究对象.

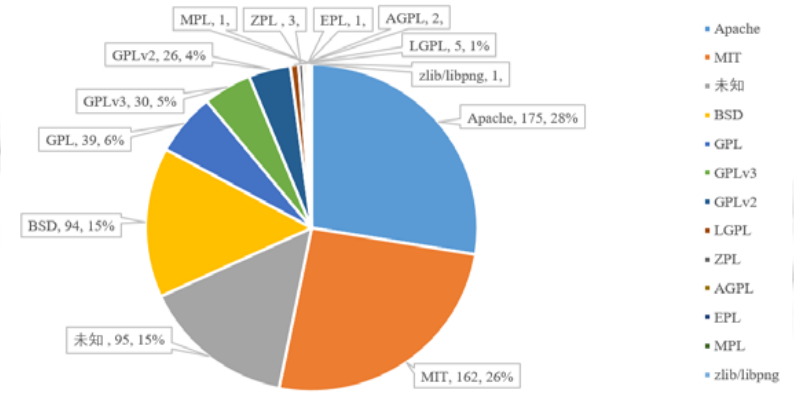


图 1 常用开源许可证使用比例图

开源许可证根据限制条件的强弱, 可以分为强 Copyleft 许可证、弱 Copyleft 许可证和宽松型许可证[32]. 强 Copyleft 许可证要求任何衍生作品都按照原始作品所使用的同一许可证分发并保持开源, 例如 GPLv3, OSLv3, Mulan PubL v2; 弱 Copyleft 许可证允许衍生作品基于其他许可证分发, 但要保证衍生作品中所使用的弱 Copyleft 许可证授权代码的许可证不做更改并保持开源, 例如 EPLv2, MPLv2; 宽松许可证允许作品中的任意代码基于其他许可证分发, 例如 BSD 许可证、Mulan PSL v2、Apachev2.0 许可证. 其中: Copyleft 是由自由软件运动发展而来的概念, 是一种开源许可方式. 它授予用户复制、修改或分发软件的权利, 但要求重新分发的作品都将基于相同的开源许可证发布并保持开源, 无论修改与否. 因此, 本文首先按照许可证类型对上述 11 种许可证从许可证创建的目的以及各个许可证所包含的主要内容做详细的介绍, 以便更好地了解许可证之间存在的共性与差异性, 进而帮助开发者梳理和掌握常用许可证中主要条款所包含的权利、义务及其他限制条件.

2.1.1 强 copyleft 许可证

(1) GPL 许可证

GNU 通用公共许可证(GNU general public license, GPL)由 Richard Stallman 于 1989 年编写[46]. GPL 许可证是最具影响力与使用最广泛的开源许可证, 同时也是开源许可证中限制条件最严格的许可证之一. GPL 的出

发点是保证用户可以自由地使用、修改和分发开源代码, 但是不允许用户将修改后的代码或创建的衍生作品在完全闭源之后, 以商业软件的方式进行分发或借此方式来实现盈利. 例如, 如果一个作品是在使用 GPL 许可证分发的开源代码基础上创建而成的, 那么该作品也必须采用 GPL 许可证分发, 并且必须公开其软件源代码或给出获取软件源代码的有效方式, 即 GPL 许可证具有“传染性”[47]. GPL 许可证主要包括 4 个主流版本(GPLv2, GPLv2+, GPLv3, GPLv3+), 但各个版本之间并非都兼容, 其中, GPLv2 许可证与 GPLv3 及 GPLv3+不兼容. 相对于 GPLv2 而言, GPLv3 进一步解决了软件专利的问题、许可证违规终止条款和开源许可证与其他商业许可证之间兼容性问题. 如果用户想要整合不同版本的 GPL 许可证的代码, 只有在低版本 GPL 许可证中包含“本许可证或任何更新版本”语句时才能够进行代码整合. 即: 只要 GPL 许可证中明确指出如果存在新版本, 则可以选择按照许可证当前版本或最新版本执行, 此时可以进行代码整合, 否则将会违反许可证条款.

(2) AGPL v3.0 许可证

AGPL v3.0(GNU affero general public license version 3, AGPLv3.0)是由 Affero 公司发布的. AGPL v3.0 是比 GPLv3 更为严格的许可证, 其继承了 GPLv3 的条款并弥补了 GPLv3 的许可漏洞[46], 即专门针对作品的网络分发进行限制. 通过网络服务方式在互联网上直接使用或修改后, 使用 AGPL v3.0 分发的网络服务系统, 其软件作品也必须开源.

(3) 木兰公共许可证, 第 2 版(Mulan PubL v2)

木兰公共许可证, 第 2 版(Mulan public license, version 2, Mulan PubL v2)是我国自主研发的强 copyleft 类许可证. 木兰公共许可证致力于让任何创造性作品都有机会被更多人分享和再创造, 共同促进开源作品的传播以发挥其最大价值. 木兰公共许可证填补了现有许可证关于云服务场景适用性的空缺, 并明确了与 AGPL 许可证及更新版本单向兼容. 具体内容如下.

0) 定义. 通过添加“衍生作品”、“分发”等定义, 明确了分发行为的范围包括了网络分发的行为, 并对云服务模式进行了补充. 即: 如果云服务平台使用了基于本许可证分发的代码, 则基于云服务平台的应用也会受本许可证约束; 如果只是基于云服务平台开发的应用使用了基于本许可证的代码, 则只有该应用会受本许可约束, 该约束将不会扩散到云服务平台;

1) 明确授予版权. 在满足木兰公共许可证条款的前提下, 版权所有者免费授予用户使用、复制、修改、分发贡献或者衍生作品的权限;

2) 明确授予专利许可. 在专利侵权保护方面, 解决现有许可证存在的联盟互诉漏洞;

3) 明确不授予商标许可. 木兰公共许可证禁止借商标权人的名义进行广告或宣传;

4) 明确分发限制条件. 可以在任何媒介中自由分发, 但在分发时必须满足以下条件.

   1. 需要附带本可证副本; 并且
   2. 如果直接分发原始作品, 则需要确保原始作品的源代码可以获得并基于本许可证分发; 如果分发其衍生作品, 需要提供对应的源代码或对应源代码的有效的访问链接或书面请求, 并基于本许可证分发, 且有效期不得少于 3 年;

5) 违约与终止. 如果用户违反了本许可证的条款, 任何贡献者有权通过书面通知的方式要求用户停止侵权行为, 并给予了更多的补救机会. 只要下游用户遵守本许可的条款, 则不会受其影响;

6) 例外条款. 明确了与 AGPL 许可证或其更新版本兼容;

7) 明确免责声明与责任限制. “贡献者”或版权所有者不对用户使用过程中引发的任何直接或间接损失承担责任;

8) 语言条款. 木兰公共许可证采用中英文双语表示, 中英版具有相同的法律效力. 当发生冲突时, 以中文版为主.

2.1.2 弱 copyleft 许可证

(1) LGPL 许可证

GNU 宽通用公共许可证(GNU lesser general public license, LGPL)是由自由软件基金会公布的接口级别的

开源许可证, 其主体内容仍沿用 GPL 许可证的主要内容, 但 LGPL 与 GPL 全部软件被“感染”的特性不同. 相比 GPL, LGPL 的限制条款是较为宽松, 主要针对类库的使用进行了规范, 明确了以动态链接方式使用 LGPL 代码时不属于 LGPL 衍生作品. 当商业软件是通过库引用的方式来使用 LGPL 代码时, 只需要说明 LGPL 代码的来源即可, 整个商业软件的源代码不需要完全按照 LGPL 许可证进行开源. 此外, LGPL 不允许将基于 LGPL 分发的库函数代码修改后应用到非库的作品中或者基于 LGPL 库创作的作品中, 基于 LGPL 库创建的作品必须仍然以库的形式提供, 并附加显著的说明告知用户作品中包含 LGPL 库, 该作品将仍然受 LGPL 约束; 并且规定了对原作品的重写仍然属于修改作品, 同样受 LGPL 约束. 其次, LGPL 允许与非 copyleft 许可证下的代码组合, 但组合作品中包含的 LGPL 的代码仍需要满足 LGPL 条款.

(2) MPL 许可证

MPL (the Mozilla public license)是由 Mozilla 基金会开发并持有的模块级别的开源许可证. MPL 的出现, 主要是为了平衡专业软件与开源软件之间的关系. MPL 主要包括 3 个版本: MPLv1.0, MPLv1.1 和 MPLv2.0, 其中: MPLv1.0 与 MPLv1.1 与 GPL 不兼容, 但 MPLv2.0 与 GPLv2 或更新版本、Apache v2 许可证、LGPLv2.1 或更新版本、AGPLv3.0 或更新版本兼容. MPLv2.0 明确不授予用户对贡献者商标使用的权利. MPLv2.0 对专利有明确说明, 用户可以自由地使用和分发, 甚至出售专利保护的代码, 同时禁止对已受专利保护的源代码再分发时再次申请专利. 此外, MPLv2.0 许可证允许将 MPLv2.0 代码与其他许可证授权的代码进行整合, 但要求使用或修改了 MPLv2.0 授权的源代码部分继续采用 MPLv2.0 许可证, 而整个作品可以采用其他的许可证对外许可. 当使用 MPLv2.0 许可证或更新版本的代码与 GPLv2 或更新版本、LGPLv2.1 或更新版本、AGPLv3.0 或更新版本下的代码文件组合时, 整体作品将按照所选 GNU 许可进行许可, 但 MPLv2.0 所覆盖部分将是双许可, 后续接收者可以选择按照 GNU 许可或 MPLv2.0 进行许可.

(3) EPL 许可证

EPL (the Eclipse public license)是由 Eclipse 基金会创建的、针对文件级别的许可证, 旨在成为商业友好的开源许可证. 其中, EPLv1.0 取代了 CPL v1.0 (common public license), 并删除了与专利诉讼相关的条款. 2017 年 8 月发布的 EPLv2.0 通过添加二级许可证列表的方式增加了与 GPLv2 或更新版本的兼容性. 但如果没有将 GPLv2 或更高版本指定为二级许可证, 则 EPL 仍与 GPL(任何版本)不兼容. EPL v2.0 同样允许开发者免费地复制、分发和修改, 并授予贡献者版权及专利许可. EPL v2.0 规定: EPL v2.0 代码可以与其他许可证的代码组合成为新作品, 并采用其他许可证进行分发, 但必须声明哪部分使用了 EPL v2.0 代码, 并且此部分代码将继续遵循 EPL v2.0 条款. 同时, 该许可证规定: 当你在发布修改后的代码时, 必须声明原作品的源代码是可获得的, 并告知接收者合理获取的方式.

2.1.3 宽松型许可证

(1) Apache 许可证

Apache 许可证(Apache license)是由著名的非盈利组织 Apache 软件基金会发布的. Apache v2.0 许可证同样允许源代码的复制、修改和分发, 也鼓励保护原作者版权, 并为贡献者提供专利保护和禁止商标, 非常适合于注重专利内容的开发者. Apache v2.0 许可证规定: 如果修改了 Apache v2.0 的源代码, 则需要对修改内容添加明显的修改说明文件. Apache v2.0 许可证还规定: 在分发衍生作品时, 可以使用其他许可证分发, 但需要附加 Apache v2.0 许可证副本, 并保留原作品中的版权、商标、专利声明以及作品来源的归属通知等相关说明. Apache v2.0 许可证规定接收者可以根据自己的意愿附加单独的免责声明和责任担保.

(2) BSD 许可证

BSD 许可证(Berkeley software distribution license)是由加州大学伯克利分校发表的, BSD 许可证是对软件的分发使用的限制条件最为宽松的许可证. BSD 许可证包含多个不同版本的许可证, 其中, 3-Clause BSD 许可证不包含专利许可证条款, 允许接收者修改原始作品的代码, 并可以将修改后的代码再次开源或者作为商业软件闭源使用. 但不论是以源代码形式还是以二进制形式分发新作品, 都必须保留源代码中的版权声明与免责声明等. 此外, 3-Clause BSD 许可证规定, 不能利用贡献者的名字进行商业宣传和市场推广.

(3) MIT/X11 许可证

MIT/X11 许可证(MIT/X11 license)是由麻省理工学院创建的许可证. MIT 许可证是和 BSD 许可证一样宽松的许可证, 同样具有很好的兼容性. MIT 许可证不包含专利许可证条款, 任何人可以无限制地使用、修改、分发甚至出售作品副本, 但要求保留原作品中许可证的版权声明和本许可证声明.

(4) Zlib/libpng 许可证

Zlib/libpng 许可证(zlib/libpng license)是由 Jean-loup Gailly 和 Mark Adler 共同完成的, Zlib/libpng 许可证主要用于限制 Zlib 软件库的分发. 该许可证规定: 在分发作品时需要保留原作品的来源, 不得将修改后的作品歪曲为原创作品, 并不得删除原作品中相关的许可声明.

(5) ZPL 许可证

ZPL 许可证(the zope public license)是由 ZPL 社区发布的, ZPL v2.0 许可证类似于 BSD 许可证, 并且自由软件基金会(FSF)指定其与 GPL 许可证兼容. ZPL v2.0 许可证明确了无商标许可, 禁止利用贡献者的名声进行推广. ZPL v2.0 规定对原作品任何文件的修改都必须添加修改声明, 包括修改人和修改日期. 此外, ZPL v2.0 要求无论以何种形式分发衍生作品, 都必须保留许可证中的版权声明、许可证条款列表及免责声明.

(6) 木兰宽松许可证

木兰宽松许可证, 第 2 版(Mulan permissive software license, version 2, Mulan PSL v2)是我国自主研发的许可证, 并通过了 OSI 认证. 为进一步促进国内开源生态的构建和为开源许可证的发展奉献力量, 国内各优势团队联合发布了 Mulan PSL v2 许可证. Mulan PSL v2 许可证以中英文双语表述, 具备被全球任意开源组织、企业、软件广泛采用的无差别性和国际通用性. Mulan PSL v2 的主要内容包括一下几点.

1) 明确授予版权. 在满足木兰宽松许可证条款的前提下, 版权所有者免费授予用户使用、复制、修改、分发原作品或者其衍生作品的权限;
2) 明确授予专利许可. 在专利侵权保护方面, 解决联盟存在的互诉漏洞问题. 木兰宽松许可证明确规定: 禁止贡献者或“关联实体”直接或间接地(通过代理、专利被许可人或受让人)进行专利诉讼或其他维权行动, 否则专利许可终止;
3) 明确不授予商标许可. 木兰宽松许可证禁止借商标权人的名义进行广告或宣传;
4) 明确分发限制条件. 可以在任何媒介中自由分发, 但在分发时需要附带木兰宽松许可证副本;
5) 明确免责声明与责任限制. “贡献者”或版权所有者不对用户使用过程中引发的任何直接或间接损失承担责任;
6) 语言条款. 木兰宽松许可证采用中英文双语表示, 中英版具有相同的法律效力. 当发生冲突时, 以中文版为主.

2.2 开源许可证框架的构建

通过对开源许可证内容的深入解读, 能够帮助开源开发者与企业更好地理解许可证条款内容. 同时, 在对常用许可证研究过程中, 我们发现许可证之间存在明显的演化关系, 即许多新许可证是在常用许可证基础上修改完善而创建的. 并且不同开源许可证之间存在很多共性, 简要概括为以下几点.

(1) 用户对获得源代码有使用和修改的权利;
(2) 用户对使用或修改后的代码有再分发的义务;
(3) 用户对获取的源代码进行分发时必须保持源代码的完整性;
(4) 开源许可证通常都附加免责声明与责任限制.

同时, 许可证之间存在明显的演化关系, 即许多新许可证是在常用许可证修改完善的基础上创建的. 通常表现为两种演化形式: 一种最为常见的方式是同一许可证派生出不同的版本, 例如, GPLv3.0 与 GPL3.0+基本一致; 另一种方式是不同许可证之间的派生, 最为典型的例子是早期的 Apache 许可证与 BSD 许可证基本相同, 之后的 Apachev1.1 许可证只是在 BSD3 许可证的基础上附加部分条款之后组成的. 同样, Zlib/libPng 许可证和 Sleepycat 许可证也是从 BSD 许可证派生而来. ECL-2.0 由 Apache 2.0 许可组成, 该许可证通过修改

Apache 2.0 第 3 节中专利授权的范围, 以满足特定于教育社区的使用需要. OpenSSL 许可证是在 Apache v1.1 许可证基础上衍生而来的, CDDL 是在 MPL1.1 基础上创建的等.

因此, 为了进一步帮助开发者完成对现有开源许可证快速梳理与理解, 本文从许可证条款的结构出发, 整理并提取了开源许可证统一框架, 如图 2 所示. 该框架适用于指导开源许可证的理解及新开源许可证的编制, 其中, 根植于本土的木兰系列许可证就是在此基础上创建的. 开源许可证的统一框架主要由 3 类 10 项指标构成, 主要包括: (a) 许可证的基本信息; (b) 序言; (c) 定义条款; (d) 知识产权条款; (e) 使用、修改与分发行为限制条款; (f) 许可证版本与兼容性; (g) 准据法条款; (h) 违约与授权终止条款; (i) 担保与责任限制条款; 以及(j) 许可证使用说明. 我们在之前的文献[48,49]中对这 10 项指标做了详细的说明.

其中, (a) 许可证的基本信息、(d) 知识产权条款中的版权许可、(e) 限制条款和(g) 担保与责任限制条款等为常见的开源许可证必选条款. 根据开源许可证不同类型限制程度的差异性, 框架中, (e) 限制条款对分发开源软件的义务和权利存在一定的差异性, 不同许可证根据其发布目的及所维护的实际需求进行特别说明. 具体表现在作品的分发形式(源代码形式或目标代码形式)、分发媒介(物理介质、网络分发等)、传染性(对再许可方式的限制)、附件文件(是否需要附加修改声明或添加 Notice 等文件)、是否收费(付费或免费分发)以及其他一些特有的限制条件(如不得歪曲原始软件来源等).

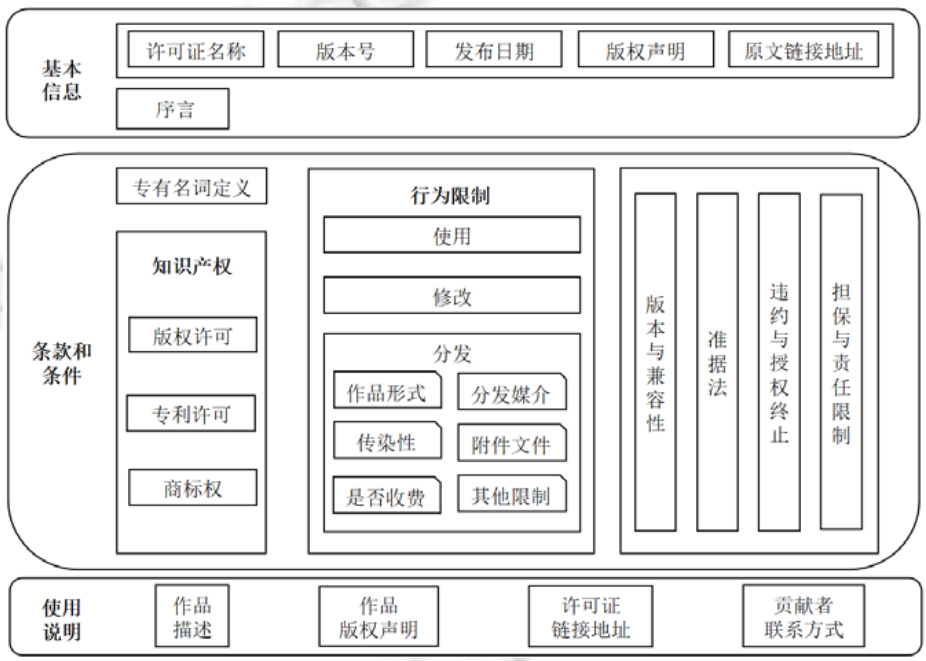


图 2 开源软件许可证模型框架

2.3 开源许可证框架的应用

下面, 本文以常用的两种许可证(Apache v2.0、GPLv3)为例进行结构化的精简解读, 并进一步验证该框架的有效性, 如图 3 和图 4 所示. 图中的数字编号表示的是许可证条款顺序, 括号内的内容表示的是与框架所对应的维度. 我们按照许可证文本内容的描述顺序对许可证进行精简概括, 阐述了许可证对应的条款, 以方便开发者快速了解该许可证包含哪些具体的条款. 开发者能够在分析许可证过程中, 结合许可证框架结构化地梳理许可证内容, 明确许可证中包含的权利与义务, 在理解许可证内容的基础上, 明确该许可证是否满足自己的开源需求.

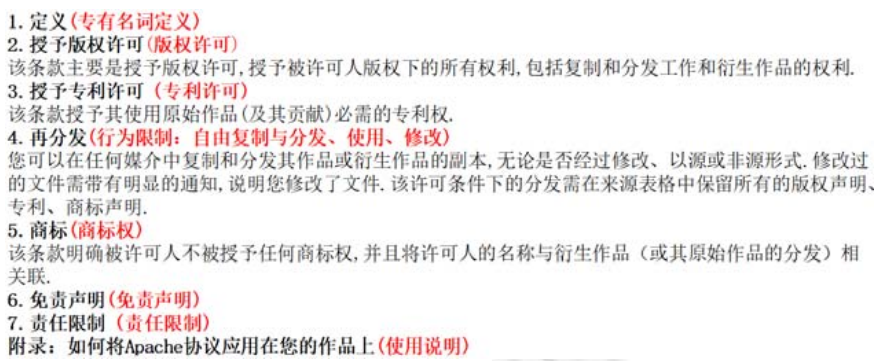


图 3 Apache v2.0 主要内容解读

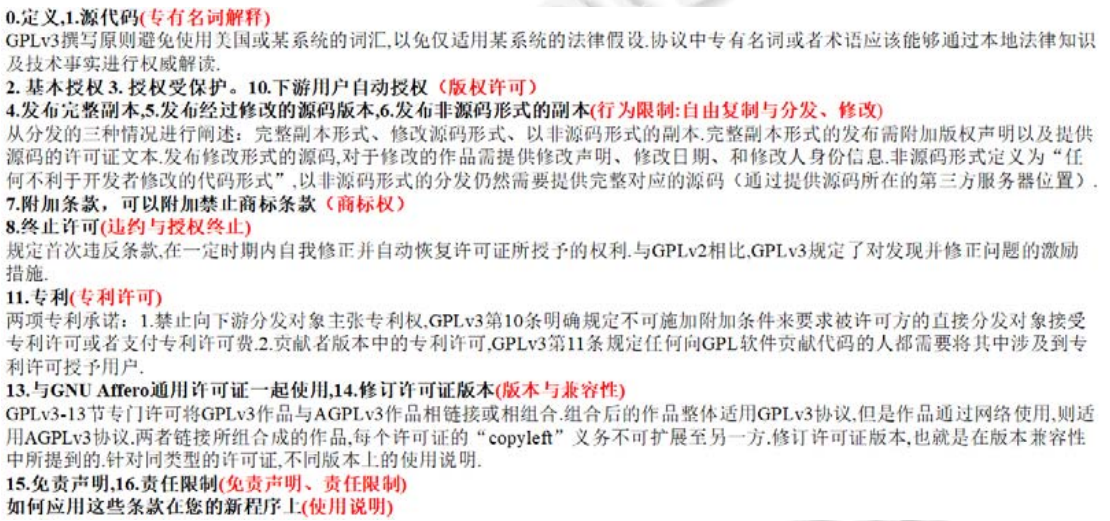


图 4 GPLv3 主要内容概括

3 开源许可证兼容性分析

许可证兼容性是指能够将不同许可证授权的源代码组合到同一软件中[36]. 如果开源软件开发过程中组合了多个基于不同许可证的源代码, 并且这些许可证条款中包含相互冲突的必要条件, 那么开发者无法将这些源代码组合到同一软件. 例如, 开发者将 Apache v2.0 代码和 GPLv2 代码组合将会是违规的, 这是因为在 Apache v2.0 中明确授予了用户专利许可及某些专利终止和侵权保护条款, 而 GPLv2 禁止专利许可, 因此 Apache v2.0 与 GPLv2 是相互不兼容的. 而对于 GPLv3 而言, 将 Apache v2.0 代码和 GPLv3 代码的组合将是允许的, 并且我们可以将组合后的代码基于 GPLv3 分发, 但基于 Apache v2.0 分发将是禁止的. 因为 GPLv3 明确授予用户专利许可证, 并且其他条款也与 Apache v2.0 保持一致, 同时, Apache v2.0 允许基于其他许可证进行二次分发, 因此 Apache v2.0 兼容 GPLv3. 但 GPLv3 是强 copyleft 许可证, 要求任何衍生作品都要基于 GPLv3 分发并提供相应的源代码, 禁止使用其他许可证进行分发, 因此, GPLv3 不兼容 Apache v2.0.

开源许可证兼容性的研究对指导开发者合规使用开源许可证有着重要作用, 它可以帮助开发者判断是否可以将两个或多个不同许可证下的作品组合到另一个许可证的作品中. 下面, 本文将根据开源许可证规定的权利与义务及许可证官方网站汇总的常见问题列表等, 对常用的开源许可证以及木兰宽松许可证之间的兼容性关系进行梳理和详细分析.

3.1 常用开源许可证兼容性分析

根据第2节中对常用开源许可证主要内容的梳理可知, 同一许可证不同版本之间不一定兼容. 例如, 不同版本的 GPL 许可证之间并不都相互兼容. 同时, 我们发现许多常用开源许可证在版本更新过程中主要着力解决知识产权方面的问题以及与 GPL 许可证之间的兼容性问题. 据此, 本文首先根据对许可证文本内容的理解、相关文献及兼容 GPL 的许可证列表(https://www.gnu.org/licenses/license-list.html)等先验知识分析不同版本的常用许可证与不同版本的 GPL 之间的兼容性, 并分析了不兼容原因, 具体分析结果见表 1. 通常, 宽松版许可证能够很好地兼容 GPL, 但当宽松版许可证包含广告条款(即所有使用到该软件的产品都需要附加该软件所属的组织)时将会打破这种兼容关系. 例如, 4-BSD 许可证中包含了广告条款, 使其与 GPL 不兼容, 但删除了广告条款的 3-BSD 许可证能很好地兼容 GPL. 表中的"–"表示由于该许可证与 GPL 单向兼容, 因此省略与 GPL 不兼容原因. 此外, 如果表中没有特别指出 GPL 的版本号, 则默认表示 GPLv2, GPLv3 及这两份许可证的更新版本.

表 1 常用许可证与 GPL 之间的兼容性

分类	许可证名称	与 GPL 兼容性	与 GPL 不兼容原因
强 copyleft 许可证	GPL	GPLv2 与 GPLv3 互不兼容	专利问题
	AGPL v3	AGPL v3 不兼容 GPLv3 GPL v3 兼容 AGPLv3	AGPL v3 覆盖了通过网络服务或计算机网络分发只适用于代码组合或链接情况
弱 copyleft 许可证	LGPL v3	LGPL v3 兼容 GPLv3	–
	MPL	MPL v1.0/v1.1 与 GPL 互不兼容 MPLv2.0 兼容 GPLv2 或更新版本	附加条款 –
	EPL	EPL v1.0 与 GPL 互不兼容 EPL v2.0 与 GPL 互不兼容	EPL 弱 copyleft 属性和选择适用法律条款 (但如果初始贡献者在分发时指明以 GPLv2 或更新版本作为次级许可证时兼容 GPL)
宽松型许可证	Apache v2.0	Apache v2.0 兼容 GPLv3	–
	MIT	MIT 兼容 GPL	–
	Zlib	Zlib 兼容 GPL	–
	ZPL	ZPLv1.0/v1.1 与 GPL 互不兼容 ZPLv2.0/v2.1 兼容 GPL	附加广告说明 –
	BSD	原始 BSD 与 GPL 互不兼容 2/3-BSD 兼容 GPL	附加广告说明 –

根据表 1 的结果显示, 本文已经得到常用许可证与 GPL 之间的兼容性. 接着, 本文进一步推理出了许可证兼容性关系的一般规则, 可以概括为以下几点.

(1) 许可证兼容是有方向的, 并不是双向可逆的. 本文所讨论的兼容性关系指的是单向兼容, 即: 如果一个软件中包含两个不同许可证 L_1, L_2 授权的开源组件, 且可以根据 L_1 分发, 或者 L_2 中的许可条款完全遵循 L_1 的许可条款, 那么我们认为 L_2 单向兼容 L_1. 例如, GPLv3 兼容 AGPLv3.0, 即: 您可以将基于 GPLv3 许可的作品与基于 AGPLv3.0 许可的作品组合成一个新的作品, 组合后的新作品可以基于 AGPLv3.0 再分发. 需要注意的是: 如果您将基于 GPLv3 许可的作品直接替换为 AGPLv3.0 或简单修改后基于 AGPLv3.0 分发都将是禁止的, 反之亦然. 即, 该兼容性关系只适用于组合或链接作品的再分发情况;

(2) 向上兼容规则. 对于同类型或不同类型许可证而言, 限制条件较少的许可证兼容限制条件较多的许可证, 即便是限制条件类似的两份许可证. 例如 MIT 与 BSD, 最终软件也应该在相对更严格的许可证下许可, 即基于 BSD 许可证许可;

(3) 对于同一许可证而言, 一般旧版本的许可证兼容新版本, 反之不成立(除 GPL 许可证).

在此基础上, 本文继续研究了不同类型的常用许可证之间的兼容性关系.

我们将常用许可证表示为 $L=\{l_1,l_2,l_3,\ldots\}$, 许可证分类类型表示为 $LT=\{S,W,P\}$, 其中, S 表示强 copyleft 许可证, W 表示弱 copyleft 许可证, P 表示宽松型许可证. 许可证兼容性关系取值的集合为

$$CT=\{C_{l_x\_\text{to}\_l_y},I_{l_x\_\text{to}\_l_y},U_{l_x\_\text{to}\_l_y}\},$$

其中, $C_{l_x\_\text{to}\_l_y}$ 表示任意两个许可证 l_x 单向兼容 l_y, I 表示任意两个许可证 l_x 不兼容 l_y, U 表示任意两个许可证 l_x 与 l_y 的兼容性关系不确定. 在许可证分析过程中, 我们将初始的许可证定义为 l_o, 将作品再分发所使用的许可证定义为 l_t. 因此, 许可证之间的兼容性关系可以表示为$\{(l_o,l_t),CT_*\}$, 其中, CT_*取 CT 集合中的任意值. 如果是作品中集成了多个不同许可证的代码, 那么原始许可证可以表示为 $l_{o1},l_{o2},l_{o3},\ldots$. 多许可证兼容性关系可以表示为$\{(l_{o1},l_{o2},l_{o3},\ldots,l_t),\{CT_*\}\}$, 最终的许可证兼容性为多对许可证兼容性关系的综合结果, 如果存在任意的不兼容许可证对, 则最终的代码组合判定为不兼容. 同时, 结合第 2 节中给出的许可证框架, 我们将许可证框架维度特征集合表示为 $F=\{f_a,f_b,f_c,\ldots\}$, 其中, 每个维度内包含的条款表示为 $f_a=\{f_{a1},f_{a2},f_{a3},\ldots\}$, 见表 2. 然后, 我们将每个条款所对应的属性标签表示为 $T=\{NM,Y,N\}$, 其中, NM 表示许可证文本中没有提及该条款, Y 表示对该条款的正面描述, N 表示反面描述.

表 2 许可证特征短语集合

类别	特征短语	特征符号
知识产权 f_a	版权许可	f_{a1}
	专利许可	f_{a2}
	商标	f_{a3}
行为限制 f_b	分发代码	f_{b1}
	合并/修改代码	f_{b2}
	传染性	f_{b3}
	声明变更	f_{b4}
	使用库	f_{b5}
	网络分发	f_{b6}
其他条款 f_c	专利维权	f_{c1}
	免责声明	f_{c2}
	责任担保	f_{c3}
	准据法	f_{c4}
	版本与兼容性	f_{c5}

根据以上定义, 我们可以使用许可证维度特征集合中内包含的条款及其对应的标签来表示一个许可证. 例如, 我们将某个特征在许可证 l_i 中的关系表示为 $(f_{a1},T_{*(a1,l_i)})$, 其中, $l_i\in L$, $f_{a1}\in f_a\in F$, $T_{*(a1,l_i)}$ 根据具体的许可证文本内容取 T 集合中的对应值. 可以将许可证 l_i 表示为 $\{(f_{a1},T_{*(a1,l_i)}),(f_{a2},T_{*(a2,l_i)},(f_{a3},T_{*(a3,l_i)}),\ldots)\}$. 最后, 将许可证的分类类型加入到许可证的表示式中, 则 $l_i=\{(f_{a1},T_{*(a1,l_i)}),(f_{a2},T_{*(a2,l_i)},(f_{a3},T_{*(a3,l_i)}),\ldots,LT_*)\}$. 这样就可以通过表达式 l_i 完整地表示一个许可证. 下面, 本文将在许可证表示的基础上加入兼容性关系, 主要以判断两个许可证之间的兼容性关系为例进行说明.

在判断兼容性时, 首先按照特征集合将原始许可证文本与框架进行映射得到每个特征集合内许可证条款 $f_{a^*(l_o)}$ 及对应的属性值 $T_{*(a^*,l_o)}$, 接着对目标许可证 l_t 执行相同步骤. 接着判断两个许可证的兼容性关系, 考虑到在许可证框架分析过程中, 许可证的主要差异性体现在知识产权及分发限制条款方面, 因此, 我们主要关注两个许可证中知识产权条款特征 f_a 及使用、修改与分发行为限制条款特征 f_b 属性标签关系. 我们通过分析对应条款之间的缺失、包含、对立关系, 并结合上文中的许可证兼容性规则, 初步完成许可证兼容性关系的分析. 以 Apachev2.0 与 GPL v3.0 之间兼容性关系为例, Apachev2.0 许可证中的主要特征可以表示为

$$\{(f_{a1},T_{Y(a1,l_o)}),(f_{a2},T_{Y(a2,l_o)}),(f_{a3},T_{N(a3,l_o)}),(f_{b1},T_{Y(b1,l_o)}),(f_{b2},T_{N(b2,l_o)}),(f_{b3},T_{N(b3,l_o)}),$$
$$(f_{b4},T_{Y(b4,l_o)}),(f_{b5},T_{N(b5,l_o)}),(f_{b6},T_{N(b6,l_o)}),(f_{c1},T_{Y(c1,l_o)}),LT_P\}.$$

GPL v3.0 许可证中的主要特征可以表示为

$$\{(f_{a1},T_{Y(a1,l_t)}),(f_{a2},T_{Y(a2,l_t)}),(f_{a3},T_{NM(a3,l_t)}),(f_{b1},T_{Y(b1,l_t)}),(f_{b2},T_{Y(b2,l_t)}),(f_{b3},T_{Y(b3,l_t)}),$$
$$(f_{b4},T_{Y(b4,l_t)}),(f_{b5},T_{N(b5,l_t)}),(f_{b6},T_{NM(b6,l_t)}),(f_{c1},T_{Y(c1,l_t)}),LT_S\},$$

其中, 特征符号的含义如表 2 所示. Apachev2.0 许可证不要求合并或修改后的代码必须开源, 并且不要求使用相同许可证分发(即传染性); 而 GPLv3.0 许可证正好相反, 除此之外不存在其他对立的条款. 结合许可证向上

兼容规则以及 GUN 社区的兼容 GPL 许可证列表, 进一步确认 Apachev2.0 许可证兼容 GPL v3.0. 最终, Apachev2.0 许可证兼容 GPL v3.0 的关系可以表示为 $\{(l_o,l_t),CT_{\text{CApachev2.0\_to\_GPL v3.0}}\}$, 但 GPLv3.0 不兼容 Apachev2.0 许可证, 表示为 $\{(l_o,l_t),CT_{\text{IGPL v3.0\_to\_Apachev2.0}}\}$.

通过上述方法完成了许可证文本内容的分析, 然后识别影响许可证兼容性的维度并进行许可证条款差异性对比; 接着, 进一步结合其他先验知识分析对应的兼容性关系. 最终, 我们得到了表 3 的兼容性分析表格, 其中附加了许可证之间单向兼容或不兼容关系、原因及相关来源.

最终得到的许可证兼容性结果可以指导开发者完成对常用许可证兼容关系的判断. 表 3 中影响许可证兼容性部分维度的具体含义如下.

- 分发代码: 将该许可证下的原件或者复制件提供给第三方的行为;
- 合并/修改代码: 将开源软件中的整体/部分代码(无论修改与否)添加到自己的代码中, 从而构成一个新作品, 该作品在分发时必须提供源代码或源代码获取方式;
- 使用库: 要求在编译或运行时通过链接(静态链接或动态链接)、导入等方式把开源代码与自己的代码绑定在一起后, 自己的代码分发时必须提供源代码或源代码获取方式;
- “传染性”: 要求通过“合并/修改、使用库”等方式将开源代码整体或部分与自己的代码组合后, 必须使用与开源软件相同的许可证进行分发并禁止闭源分发;
- 网络分发: 通过网络服务或计算机网络分发开源软件的行为.

为了更加清晰地描绘许可证之间的兼容性关系, 本文通过可视化的方式表示得到的兼容性结果. 最终得到常用开源许可证兼容性推导模型, 如图 5 所示. 此前, Wheeler[37]通过收集汇总许可证的各类基础信息, 第 1 次利用图方法表示了许可证之间的兼容性关系. Kapitsaki 等人[36]在其基础上涵盖了更多的许可证, 并进一步提供了对应的信息来源, 然后通过同样的方法对兼容性结果进行更加精细的可视化表示.

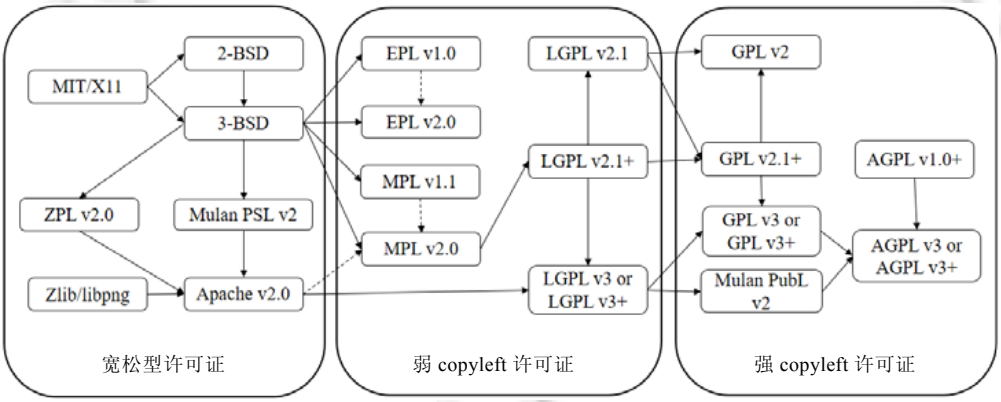


图 5 常用许可证兼容性推导模型

相较于 Kapitsaki 等人的分析方法及兼容性图, 本文同样沿用了 Wheeler 的基础研究方法. 同时, 在许可证兼容性判断过程中, 本文除了利用已有的先验知识之外, 还增加了许可证条款对比来进一步说明许可证之间的兼容性关系(见表 3). 然后根据兼容性规则第 2 条, 本文在兼容性推导模型中, 将限制条件较少的许可证作为初始端, 将限制条件较多的许可证作为尾端. 图 5 中有两种类型的边(兼容性可传递边和兼容性不可传递边), 实线箭头为兼容性可传递边, 虚线兼容性为不可传递边; 图中的端点表示开源许可证.

表 3 常用许可证兼容性分析

类别	开源许可证	版权许可	专利许可	专利维权	商标	声明变更	分发代码	合并/修改代码	传染性	使用库	网络分发
强 copyleft 许可证	GPLv2	√				√	√	√	√	√	
	GPLv3	√	√	√		√	√	√	√	√	
	AGPLv3	√	√	√		√	√	√	√	√	√
弱 copyleft 许可证	LGPLv2.1	√	√	√		√	√	√			
	LGPL v3	√	√	√		√	√	√			
	EPLv1.0	√	√				√	√			
	EPLv2.0	√	√				√	√			
	MPLv1.1	√	√		√	√	√	√			
	MPLv2.0	√	√		√		√	√			
宽松型许可证	MIT License	√					√				
	BSD 2-Clause	√					√				
	BSD 3-Clause	√			√		√				
	Apache 2.0	√	√	√	√	√	√				
	ZPLv2.0	√			√	√	√				
	Zlib/libpng	√					√				

表 3 常用许可证兼容性分析(续)

类别	兼容性		
	关系	许可证	原因及来源
强 copyleft 许可证	不兼容	GPLv3	GPLv2 本身不与 GPLv3 兼容
	不兼容	AGPLv3	AGPL 不兼容 GPLv2
	不兼容	GPLv2	GPLv3 本身不与 GPLv2 兼容
	兼容	AGPLv3	GPLv3 第 13 节指出, 可与 AGPLv3 授权的程序链接或组合
	不兼容	GPL	同上. 此外, 您不能按照 GPLv3 的条款获取在 AGPL 下发布的代码并按照您喜欢的方式传递或修改它; 反之亦然
弱 copyleft 许可证	兼容	GPLv 和 GPLv3	LGPLv2.1 不是一个强 copyleft 许可证, 因为它允许与非自由模块链接. 它与 GPLv2 和 GPLv3 兼容
	兼容	GPLv3	同理, LGPLv3 不是一个强 copyleft 许可证, 因为它允许与非自由模块链接. 它与 GPLv3 兼容
	不兼容	GPL/AGPL/LGPL	弱 copyleft 属性和选择使用法律条款
	不兼容	GPL/AGPL/LGPL	弱 copyleft 属性和选择使用法律条款(如果初始贡献者在分发时指明以 GPLv2 或更新版本作为次级许可证时兼容 GPL)
	不兼容	GPL/AGPL/LGPL	任何一个 GPL 模块和一个 MPLv1.1 模块不能合法连接到一起. 这意味着只能在 MPL 以前版本下可用的软件仍然与 GPL 和 AGPL 不兼容
	不兼容	MPLv1.1	MPL 1.1 中缺少 MPL 2.0 的第 3.3 条款
	兼容	LGPLv2.1+	MPLv2.0 第 3.3 节提供了本授权与 GPLv2.0, LGPLv2.1, AGPLv3.0 以及这些许可的所有后续版本之间的间接兼容性
宽松型许可证	兼容	BSD 2-Clause 和 BSD 3-Clause	BSD 2-clause 和 3-clause 许可都声明“允许以源代码和二进制形式(无论是否经过修改)重新发布和使用”
	不兼容	MIT	MIT 许可证允许在没有贡献积分的情况下发布, 而 BSD 许可证则不允许.
	兼容	BSD 3-Clause	BSD 2-Clause 删除了 BSD 3-Clause 中有关商标的条款.
	不兼容	MIT	MIT 许可证允许在没有贡献积分的情况下发布, 而 BSD 许可证则不允许
	兼容	ZPLv2.0	ZPLv2.0 是在 BSD 许可证基础上修改的, 并拥有比 BSD 许可证更多的权限
	不兼容	MPLv1.1	MPLv1.1 的第 13 条在 Apache v2.0 中不存在
	兼容	MPLv2.0	在 MPL 许可证常见问题列表(https://www.mozilla.org/MPL/2.0/Revision-FAQ.html#what-has-changed)的 No8“中指出该许可证与 Apache 许可证兼容: 任何遵守 MPLv2.0 条款的人也应该遵守 Apache 许可证的条款
	不兼容	GPLv2.0	Apache v2.0 与 GPLv2.0 不兼容, 因为它有一些在该 GPL 版本中没有的要求. 这些条款包括某些专利终止和赔偿条款
	兼容	GPLv3.0	根据 GNU 社区(https://www.gnu.org/licenses/license-list.htm)“Apache v2.0 兼容 GPLv3.0”
	兼容	Apache v2.0	根据 Apache 社区(https://www.apache.org/legal/resolved.html#category-a)“可以在 ASF 项目中包括 ZPLv2.0 授权的代码”表明兼容 Apache v2.0. https://www.apache.org/legal/resolved.html#category-a
	兼容	Apache v2.0	根据 Apache 社区“可以在 ASF 项目中包括 zlib/libpng 授权的代码”表明兼容 Apache v2.0

首先, 我们需要明确两类箭头直接相连的两个许可证按照边的箭头指向单向兼容, 但在兼容性是否可传递上存在差异. 最终, 许可兼容性推导模型中存在以下几种推导关系.

(1) 通过兼容性推导模型, 当从一个许可证出发到另外一个许可证终止, 至少有一条通路并且全部都是实线时, 不同许可证之间的组合是合规的, 组合后的作品可以基于通路终点的许可证进行分发; 例如图 5 中 MPLv2.0→LGPLv2.1+→GPLv2.1+→GPLv2 是一条全为实线的通路, 则 MPLv2.0 与 GPLv2 兼容, 可以进行代码整合, 并基于 GPLv2 分发;

(2) 当从一个许可证出发经过其他许可证到达另外一个许可证终止的唯一通路中存在虚线时, 那么初始许可证兼容许可证列表至第 1 次出现虚线时所指的许可证为止, 与之后的许可证是不兼容的. 例如图 5 中存在通路 Apachev2.0 通过虚线指向 MPLv2.0, 然后紧接着 MPLv2.0→LGPLv2.1+→GPLv2.1+→GPLv2, 从 Apachev2.0 出发到 GPLv2 的通路中存在虚线, 则该通路中 Apachev2.0 兼容 MPLv2.0, 但与 MPLv2.0 之后的所有许可证都不兼容, 不能进行代码整合;

(3) 当我们从不同的许可证出发, 终点指向同一许可证. 这种情况下, 将按照最终共同所指的许可证分发开源软件. 例如图 5 中 ZPLv2.0 通过实线指向 Apachev2.0, Zlib/libpng 也同样通过实线指向 Apachev2.0, 因此可以将 ZPLv2.0 代码与 Apachev2.0 代码组合后形成的作品 W_1, 以及 Zlib/libpng 代码与 Apachev2.0 代码组合后形成的作品 W_2, 最终将 W_1 与 W_2 组合后, 再基于 Apachev2.0 分发将是合规的;

(4) 当组合代码中包含双重许可证时, 双重许可证是一种或的关系, 您可以选择其中任意一份或几份许可证进行分发. 双重许可证分为两种类型.

- 第 1 种类型为双重许可证中一份为开源许可证, 一份为商业许可证. 如果您使用开源许可证, 则按照上述第 1 种情况进行处理; 如果您购买商业许可证, 则您将作为例外按照商业许可证条款使用作品;
- 另一种类型为包含了两份或多份不同的开源许可证. 您可以选择对双重许可证中的一份许可证与其他许可证的兼容性关系进行判断: 如果该许可证与其他许可证兼容, 则认为该双重许可证与其他许可证兼容(https://www.gnu.org/licenses/license-compatibility.en.html).

通过跟踪这些许可证连接的边来推导许可证兼容性, 以查看它们是否在某个端点上到达相同的许可证, 或者是否其中一个可以从不同集合的其他许可到达. 如果存在连接两个许可证的有效通路, 那么尾端许可证就可以作为项目最终的许可证. 对于多许可证代码组合的情况, 开发者可以通过在兼容性图中找到当前项目所采用的开源许可证, 并从左侧出发找到需要组合的其他代码所使用的开源许可证, 然后可以从许可证兼容性推导模型中找到许可证之间连接这些许可证的最短的通路, 如果存在, 则在实际开发中可以进行多个兼容许可证代码的组合, 并将组合后的代码基于通路终点的许可证进行许可分发. 因此, 开源开发者根据本文的兼容性推导图不仅能够明确两个许可证之间的兼容性关系, 对于多个不同许可证代码的组合也同样适用.

根据许可证兼容性推导模型, 我们能够准确地得出不同许可证之间的兼容性关系. 同时将伴随着另外一个问题: 如果已经明确组合代码中包含的所有许可证兼容, 那么组合后的代码该如何分发?

当不同许可证的代码组合后再分发时, 首先需要明确这些组合代码中具体包含哪些许可证, 并汇总得到许可证列表. 然后, 通过兼容性推导模型进行排查, 如果许可证 A 可以被许可证 B 包含, 即许可证 A 兼容许可证 B, 则将许可证 A 从列表中删除. 重复操作之后, 最终将会剩余一份许可证. 此时, 您可以使用该许可证分发您的作品, 或选择与之兼容的其他新许可证分发作品. 最终分发的组合作品中, 各个组件继续遵循原许可证条款, 作品整体将遵循分发时所使用的许可证.

3.2 许可证兼容性推导模型的应用

本文通过对比木兰宽松许可证(Mulan PSL v2)与常用宽松许可证(Apache v2.0, 3-BSD, MIT, Zlib/libpng, ZPL v2.0)主要内容得到木兰宽松许可证与常用宽松许可证之间的异同点(表 4), 从而推断出木兰宽松许可证的兼容性, 并验证许可证兼容性推导模型的有效性及可扩充性. 木兰宽松许可证与其他宽松许可证一样赋予

开发者更多的自由, 能够更好地维护开发者合法权益. 同时, Mulan PSL v2 对版权许可以及专利许可的目标人群进行了更加明确的划分, 相对于 Apachev2.0 许可证对专利终止情况进行了更加详细的说明, 防止联盟专利互斥漏洞.

依据表 4 中开源许可证主要内容的对比结果及第 3.1 节的许可证兼容推导模型, 可以得到木兰宽松许可证与其他许可证之间的兼容性关系, 最终验证并扩充了兼容性推导模型. 根据对比结果, 我们可以将木兰宽松许可证(Mulan PSL v2)在 BSD 3-Clause 与 Apache V2.0 许可证之间添加一条与 ZPLv2.0 并列的通路. 最终, 通过兼容性推导模型我们可以得知 MIT, BSD 2-Clause, BSD 3-Clause 许可证兼容 Mulan PSL v2, 同时, Mulan PSL v2 兼容 Apache v2.0, L/GPLv3 等开源许可证. 即许可在 MIT, BSD 2-Clause, BSD 3-Clause 许可证的代码可组合到基于 Mulan PSL v2 分发的项目中, 许可在 Mulan PSL v2 下的代码同样也可以组合到基于 Apache v2.0, L/GPLv3 等许可证分发的项目中.

表 4 木兰宽松许可证与常用宽松型许可证对比表

许可证	版权	专利及 专利维权限制	商标	分发限制	免责声明与 责任限制	语言
Mulan PSL v2	所有 贡献者 及其 关联公司 提供 版权许可	1. 所有贡献者及其关联 实体提供专利许可; 2. 用户及其关联实体 直接或间接的发起 专利侵权诉讼或其他 专利维权行动, 专利许可终止	明确禁止借 商标权人的 名义进行 广告或宣传	分发时附带 许可证副本, 并保留版权、 专利、商标、 免责声明	1. 不提供任何明示 或默示的担保; 2. 不对任何直接 或间接损失 承担责任	中英 双语 具有 同等 法律 效力
Apache v2.0	原始 版权人, 后续 贡献者 及其 关联实体 提供 版权许可	1. 原始许可人, 后续 贡献者及其关联 实体提供专利许可; 2. 用户及其关联实体 直接发起专利诉讼, 专利许可终止	明确禁止借 商标权人的 名义进行 广告或宣传	1. 分发时附带许可证 副本,并保留 各种声明; 2. 分发修改版时应声明 已修改的 文件及日期; 3. 如许可软件含 Notice 文件, 则分发修改版 时应附归属声明	1. 不提供任何明示 或默示的担保; 2. 不对任何直接 或间接损失 承担责任; 3. 分发时可附加 额外的保证 或责任担保	只有 英文
3-BSD	所有 贡献者 提供 版权许可	无明确专利许可	明确禁止借 商标权人的 名义进行 广告或宣传	保留版权声明, 分发 条款列表和免责声明	1. 不提供任何明示 或默示的担保; 2. 不对任何直接 或间接损失 承担责任	只有 英文
MIT	所有 贡献者 提供 版权许可	无明确专利许可	无明确 商标许可	保留版权声明 和许可声明	1. 不提供任何明示 或默示的担保; 2. 不对任何直接 或间接损失 承担责任	只有 英文
Zlib/ libpng	所有 贡献者 提供 版权许可	无明确专利许可	无明确 商标许可	保留许可声明, 不得 歪曲软件来源, 保护 原作品版权	1. 不提供任何明示 或默示的担保; 2. 不对任何直接 或间接损失 承担责任	只有 英文
ZPL v2.0	所有 贡献者 提供 版权许可	无明确专利许可	明确禁止借 商标权人的 名义进行 广告或宣传	1. 保留版权声明, 分发条款列表 和免责声明; 2. 分发修改版时应明显 声明已修改的文件	1. 不提供任何明示 或默示的担保; 2. 不对任何直接 或间接损失 承担责任	只有 英文

4 开源许可证选择工具

4.1 许可证选择工具的构建

我们已经得到了许可证框架以及许可证兼容性推导模型, 开发者对许可证内容及对组合代码开源许可证的合规使用有了更好的认识. 在开源软件开发过程中, 一个优秀的开源项目除了项目本身的独特性与创新性能够吸引社区开发者参与到项目的开发中之外, 为该开源项目选择合适的许可证, 也能够吸引更多的开发者投入到自己的项目开发中, 让自己的项目产生真正的效益[50]. 因此, 在正确理解许可证及代码合规组合的基础之上, 为新作品或独创作品选择合适的开源许可证, 也是影响开源软件广泛传播的关键因素.

根据 Cotton 等人[51]的研究可知: 许可证的选择对软件质量的提高有着非常重要的作用, 正确地选择许可证能够帮助开发者从源头上规范自己的开源项目. 目前社区已有的开源许可证选择方案都太过于专业化, 例如由 OSSWATCH 研发的 Licence differentiator (http://oss-watch.ac.uk/apps/licdiff/)主要是基于开源许可证部分条款之间的差异帮助开发者选择许可证, 但其在问题设置以及选择工具设计方面都只适用于对许可证有充分理解的专业人士才能够有效使用, 对于开源爱好者都不够友好. 因此, 本文在结合许可证文本内容的基础上加入了许可证兼容性分析, 同时参考 choosealicense 网站(choosealicense.com)和 tldrlegal 网站(https://tldrlegal.com/licenses/browse)的许可证维度, 进一步细化了许可证框架中确定的许可证维度在限制条件方面的差异性, 进而为开发者提供了更加完整的许可证选择决策方案.

首先, 我们以常用许可证为例绘制了如图 6 的许可证选择决策树, 以可视化许可证选择工具的实现机制. 许可证选择决策树包含两个部分: 一部分为开源许可证之间的共同点, 另一部分为不同点. 图 6 中矩形框表示许可证之间的共同点, 通过菱形来判断许可证之间的差异性, 圆角矩形表示开源许可证. 并且, 图中各个节点及层级的设置与许可证选择工具相对应. 例如: 如果分发时对接收者修改源代码后是否开放源代码的行为不做要求, 但希望该许可证能够充分地维护贡献者的版权与专利权, 并且禁止借商标权人的名义进行广告宣传. 我们按照许可证选择决策树进行分析, 最终将会推荐 Mulan PSL v2 与 Apache v2.0 许可证. 如果开发者要求接收者对修改文件进行声明, 或允许后续贡献者自己附加担保或者责任声明, 则最终将会推荐 Apache v2.0 许可证, 否则将会推荐 Mulan PSL v2. 当然, 在开发过程中, 开发者也应该注意避免引入其他开源组件, 或确保引入组件所使用的许可证兼容最终分发时使用的许可证, 在确保开源软件合规使用的前提下, 按照许可证的要求分发项目.

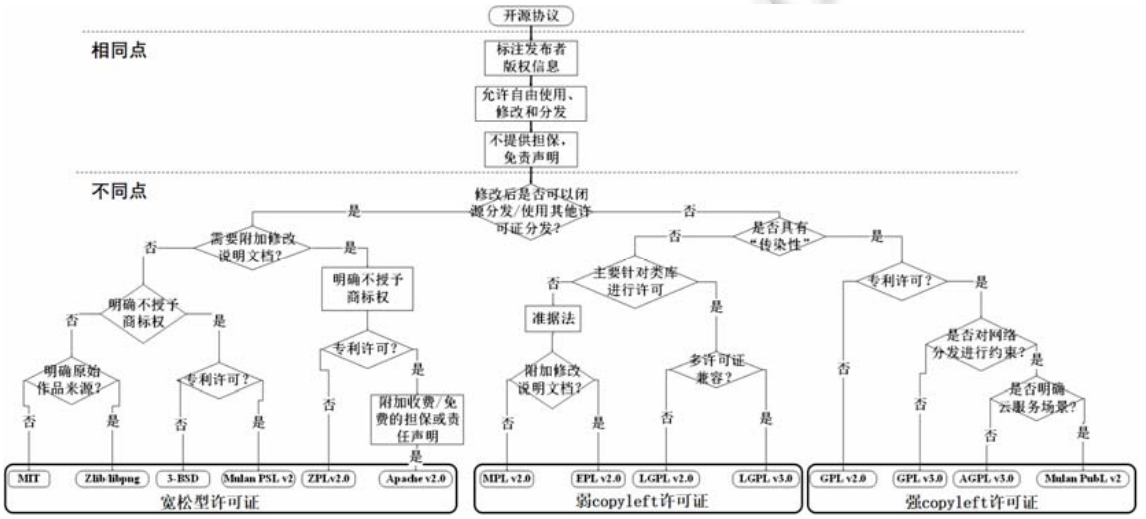


图 6 常用许可证选择决策树

在许可证选择工具的设计方面, 我们已经根据第 2 节中的许可证框架完成对许可证的结构化表示以及图

6 的可视化表示, 提取归纳了许可证中的共性以及差异性. 其中, 许可证的共性条款主要帮助开发者了解许可证包含的主要条款, 但对许可证的选择与区分的帮助较小. 因此, 在选择工具构建过程中主要关注许可证条款的差异性. 本文按照图 6 中区分许可证特性的各个节点作为参考进行许可证工具问题的设置: 首先, 将许可证选择的决策分为 4 类; 然后, 按照类别和框架中的维度确定了更细节的决策节点. 最终按照 4 个类别共确定了 15 个问题, 这些问题之间相互关联并在某些问题之间设置了制约条件, 表 5 中按照类别陈列了所有相关问题. 下面将对问题的设置及排序进行详细的说明.

表 5 许可证选择工具问题列表

类别	许可证问题列表
开源许可证列表的确定	1. 您是否对许可证推荐列表进行限制? 2. 您对许可证类型的要求?
行为限制: 分发、修改及复制相关问题	2(a) 您是否要求原始作品的源代码依旧保持开源或公开源代码的获取方式? 2(b) 您是否希望使用了您作品的任何衍生作品都在与您原始作品相同的许可证下分发? 3. 您是否希望用户对作品(或原许可证)的修改附加说明或通知? 4. 您是否永续第三方从具有不同许可证的代码动态链接到您的代码? 5. 您是否要求对网络使用或部署进行限制, 并希望将修改开源? 6. 您的作品是否作为库进行发布? 7. 您是否希望许可证与 GPL 兼容?
知识产权相关问题	8. 对于所选择的许可证, 您对版权的态度是什么? 9. 对于所选择的许可证, 您对专利的态度是什么? 9(a) 您是否需要包含相应的专利报复条款? 10. 对于所选择的许可证, 您对商标的态度是什么?
其他相关问题: 免责与责任担保	11. 您是否允许第三方根据自己的需求附加担保? 12. 您是否对许可证准据法及违规终止有要求?

许可证选择工具中涵盖的开源许可证, 除第 2 节中企业广泛使用的许可证列表之外, 进一步分析了更多的许可证. 这些许可证包含了 OSI 认证的许可证以及部分未认证的许可证, 共 41 份. 我们采用与上文中同样的方法对这 41 份许可证从更细的维度进行分析[52], 完整的许可证列表名称以及最终的许可证选择工具已正式开放使用(http://139.159.212.56:8000/). 关于许可证选择工具中问题的设计, 主要分为 4 个类别: 1) 开源许可证列表的确定; 2) 行为限制分发、修改及复制相关问题; 3) 知识产权相关问题; 4) 免责与责任担保. 其中, 开源许可证列表主要是基于先验知识(是否通过 OSI 认证、受欢迎程度、许可证类别)确定, 其他 3 类主要通过分析许可证文本内容确定. 在类别顺序确定方面, 我们主要从开发者对许可证需求的重要程度考虑, 下文中对应问题的陈列顺序也基于此设置. 开发者在选择许可证时关心较多的是许可证中行为限制条款, 其次是知识产权相关的问题, 有时开发者也会关注到免责与责任担保方面的条款, 因此我们也添加了免责与责任担保方面的问题设置. 每个类别中问题设置具体如下.

1) 开源许可证列表的确定

许可证列表的确定主要关注两个问题.

- 一是确定可供选择的许可证列表, 开发者可以选择经 OSI 认证的、深受开发者喜爱且被广泛使用或拥有强大社区的许可证作为选择列表, 这样可以保证开发者现在的许可证是主流许可证; 或者开发者可以只将 OSI 认证的许可证作为选择列表, 这样保证开发者选择的许可证权威可靠; 或者开发者可以将选择工具中包含的全部许可证作为许可证列表, 这样将提供更加全面丰富的特性;
- 另一个问题是所选许可证列表中许可证类型的确定: 如果开发者对许可证类型有基础的了解, 可以直接确定自己比较感兴趣的类型, 然后根据类型有目的的确定其他限制条件; 如果开发者缺乏对许可证类型的认识, 可以忽略此问题, 并根据后续分发限制中的问题进一步确定. 如图 7 所示, 我们将问题 1 选择“OSI 认证的许可证”时得到对应的许可证列表.

2) 行为限制: 分发、修改及复制相关问题

在行为限制方面, 我们设置了 7 个相关问题, 行为限制是开发者根据个人需求或兴趣确定许可证至关重要的一部分. 首先, 表 5 中的问题 2(a)与问题 2(b)与问题 2 息息相关, 问题 2(a)与问题 2(b)是针对不同类型的

许可证限制条款的特性设置的. 如果开发者在问题 2 中选择“强 copyleft 许可证”,则问题 2(a)与问题 2(b)将会被默认设置为选择“是”选项; 如果开发者选择“弱 copyleft 许可证”, 则问题 2(b)将会被隐藏, 只有问题 2(a)会被默认设置为选择“是”选项; 如果选择“宽松型许可证”, 则问题 2(a)与问题 2(b)都将会被隐藏. 此外, 根据许可证的其他特性及开发者在许可证选择中关注的进一步需求, 我们设置了问题 3–问题 6: 问题 3 关注于是否需要接收者对所有的修改附加说明, 这样将有助于代码溯源; 问题 4 关注于是否允许将不同许可证代码以动态链接的方式链接到原作品代码, 对于强 copyleft 许可证动态链接的方式通常是禁止的; 问题 5、问题 6 分别关注于使用网络分发以及使用库方式. 这主要是考虑到部分开源作品通过在线方式提供服务, 这种情况下, 开发者往往对于许可证是否会限制网络使用、分发, 以及通过动态链接或静态链接引用的代码是否受许可证限制等存在疑惑. 此外, 我们也在选择工具中添加了许可证兼容性问题的关注, 问题 7 关注于其他许可证与 GPL 许可证之间兼容性关系问题. 如图 8 展示了兼容 GPLv3 或 GPLv3 兼容的所有经 OSI 认证的推荐许可证.

图 7 许可证选择工具问题 1 示例

图 8 许可证选择工具问题 7 示例

3) 知识产权相关问题

这一部分主要关注许可证在版权、专利、商标等知识产权方面的不同限制和特定要求. 其中, 问题 8 关注于版权问题. 任意的许可证都会授予用户版权, 但会附加不同的义务声明, 比如保留版权说明, 或者用户根据自己的需求为修改作品添加相应的版权等. 问题 9 与问题 9(a)关注专利及专利维权问题: 如果开发者在问题 9 中选择了“明确包含专利授权条款”, 则会显示问题 9(a); 同时, 考虑到存在一部分许可证只包含了专利维权条款而没有专利授权条款(例如 GPLv2 禁止专利维权), 所以当开发者选择“我不关心”选项时, 也将显示问

题 9(a). 问题 10 是关于商标条款的问题, 通常包含商标条款的许可证都禁止商标授权. 图 9 显示了选择工具覆盖的所有经 OSI 认证的许可证中包含专利的推荐许可证.

4) 免责与责任担保.

通常, 许可证都会包含免责与责任担保条款, 但有些许可证允许用户根据自己的需求附加额外的担保, 因此问题 11 主要是针对这一问题设置的. 此外, 许可证会对违反许可的用户终止许可证中的授权, 同时, 考虑到不同国家对许可证的执行力度或解释上存在差异, 部分许可证中会添加准据法条款. 准据法是指当发生冲突时指定遵从某个国家的法律, 基于此设置了问题 12. 图 10 展示了许可证列表中包含违规终止条款的推荐许可证.

图 9 许可证选择工具问题 9 示例

图 10 许可证选择工具问题 12 示例

许可证选择工具设计过程中, 尽量地避免过于专业的问题设置, 在保证正确表达不同许可证内容差异性的基础上, 以精简易懂的方式表述许可证之间的差异性, 能够更加有效地帮助开源爱好者更好地理解许可证内容, 并帮助开源爱好者为自己的开源项目选择合适的许可证.

4.2 许可证选择工具的应用

许可证框架能够帮助开发者理解开源许可证, 许可证兼容性推导模型可以指导开发者正确组合不同开源许可证代码, 许可证选择工具能够指导开发者根据自己的需求选择合适的许可证. 在开发过程中, 贡献者可以利用许可证框架完成开源许可证内容的梳理和归纳, 在对许可证有基本的了解之后, 根据自身需求, 通过开源许可证选择工具确定可选的许可证列表. 如果开发过程中使用了其他不同许可证的代码, 可以通过兼容性推导模型确定这些许可证之间的兼容性, 如果不兼容, 需要及时规避风险.

下面以一个实际的应用为例进行说明. 我们在开发时组合了开源项目 yui-lint (github.com/yui/yui-lint)的

代码用于交互式 Web 应用程序的开发, 首先使用开源许可证检测工具 license-checker (github.com/davglass/license-checker)对该项目中所使用的开源组件及包含的许可证进行检测, 最终检测得到的许可证列表如图 11 所示. 该开源项目基于 BSD 许可证分发, 并且根据图 11 可以得知, 该开源项目中使用到的开源许可证包括 MIT, ISC 和 BSD 这 3 种许可证. 通过对这些许可证分类, 得知这 3 份许可证都是宽松型许可证, 并根据兼容性推导图发现能很好地兼容其他任意许可证. 因此, 可以将组合后的项目基于其他许可证分发.

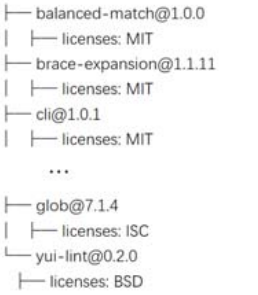


图 11 yui-lint 项目中包含的部分开源组件及其许可证

通过检测工具已经知道项目中所包含的其他许可证, 然后利用许可证选择工具, 按照自己的需求选择最终分发的许可证. 假如您是一个开源新人或对许可证了解甚少, 首先, 在许可证列表确定时, 您更倾向于被开源贡献者广泛使用的许可证, 这样能够确保最终选择的许可证更加权威可靠, 因此选择“OSI 认证的许可证”作为许可证列表. 其次, 作为新人, 您对许可证分类不是很了解, 所以忽略该问题. 您对衍生作品是否需要公开源代码以及是否基于同一许可证分发并不关心, 只希望接收者能够保留项目中已有的版权说明并附带许可证副本即可, 并要求所选择的许可证能够兼容 GPLv3 许可证, 以便更好地被其他系统软件组合. 同时, 您比较关注专利许可并有意阻止第三方提起专利相关的法律诉讼. 此外, 为了消除作品迭代过程中潜在的问题, 您禁止借您的名义进行宣传, 并且不希望接收者附加额外担保. 最终, 我们的许可证选择工具根据用户需求满足的条件给出了对应的推荐列表和相应的得分, 如图 12 所示. 您可以根据推荐列表去仔细研读对应的许可证文本内容, 在准确理解许可证内容的基础上确定最终的分发许可证.

图 12 许可证选择工具推荐结果

5 局限性

本文的局限性主要体现在以下几个方面.

- 在许可证框架分析方面, 本文通过分析常用许可证, 确定了通用许可证框架, 该框架能够很好地帮助开发者完成许可证的结构化表示, 但需要通过手工的方式完成许可证条款与框架之间的映射. 今后, 我们将进一步致力于自动提取许可证条款的研究, 以便帮助开发者快速了解许可证内容;
- 在许可证兼容性分析方面, 本文展示的许可证兼容性推导图能够很好地指导开发者正确组合代码.

但其涵盖的许可证种类依旧较少, 面对未知许可证将会失效. 此时需要开发者结合许可证框架完成许可证条款的提取, 然后根据得到的许可证条款之间的包容关系完成许可证兼容性判断;

- 在许可证选择工具方面, 本文开发的许可证选择工具能够有效地帮助开发者初步选择许可证, 并验证了许可证选择工具的有效性和普适性. 目前, 许可证工具只涵盖了其他许可证与 GPL 许可证之间的兼容性判断, 对于其他许可证兼容性关系的判断还需要进一步完善. 我们将进一步完善许可证选择工具的功能, 以更好地服务开发者. 需要说明的是: 许可证选择工具所推荐的许可证列表只能作为参考, 开发者仍然需要结合自身需求认真研读许可证文本.

6 总结与展望

开源许可证是开源运动的基石, 开源开发者对开源软件的合规使用是促进开源生态健康发展的关键[53]. 本文主要选用实际项目开发中使用率最高的十几种开源许可证及木兰系列许可证为研究对象, 从许可证解读出发, 构建了许可证框架, 并进一步分析各个许可证之间的兼容性关系. 最后, 在此基础上搭建了许可证选择工具, 以帮助开发者根据自身需求选择合适许可证. 总之, 本文的主要贡献可以概括为以下几点.

(1) 通过对许可证文本的解读, 归纳出了许可证框架, 并完成了常用许可证的主要内容的分析;

(2) 通过对许可证限制条款冲突性分析, 得到许可证兼容性推导模型, 并利用木兰宽松许可证验证了许可证兼容性推导模型的有效性;

(3) 结合许可证框架和许可证兼容性推导模型搭建了许可证选择工具, 并验证了其可行性.

本文的研究成果能够有效地指导开发者正确理解和选择开源许可证, 规范开源许可证的合规使用. 在今后的工作中, 我们将继续完善许可证选择工具的研发, 扩充兼容性推导模型, 并将其集成到许可证选择工具中. 同时, 我们将根据确定的许可证框架着手开源许可证条款自动识别工作, 以更加高效地提取许可证条款. 并持续推广木兰系列许可证, 更好地发展开源软件.

References:

[1] Jin Z, Zhou MH, Zhang YX. Open source software and its eco-systems: Today and tommorow. Science & Technology Review, 2016, 34(14): 42−48 (in Chinese with English abstract). [doi: 10.3981/j.issn.1000-7857.2016.14.005]

[2] Sauer RM . Why develop open source software? The role of nonpecuniary benefits, monetary rewards and open source licence type. Oxford Review of Economic Policy, 2007, 23(4): 605−619.

[3] Mei H, Zhou MH. Challenges brought by open source to the talents of the software. Computer Education, 2017(1): 2−5 (in Chinese).

[4] Ballhausen M. Free and open source software licenses explained. Computer, 2019, 52(6): 82−86.

[5] St Laurent AM. Understanding open source and free software licensing. O'Reilly Media, Inc., 2004.

[6] Vendome C, German D, Penta MD, *et al*. To distribute or not to distribute? Why licensing bugs matter. In: Proc. of the 40th Int'l Conf. on Software Engineering. ACM, 2018. 268−279.

[7] Kapitsaki GM, Kramer F. Open source license violation check for SPDX files. In: Proc. of the Int'l Conf. on Software Reuse. Cham: Springer-Verlag, 2015.

[8] Fendt O, Jaeger MC. Open source for open source license compliance. In: Proc. of the IFIP Int'l Conf. on Open Source Systems. Cham: Springer-Verlag, 2019.

[9] Gamalielsson J, Lundell B. On licensing and other conditions for contributing to widely used open source projects: An exploratory analysis. In: Proc. of the 13th Int'l Symp. on Open Collaboration. 2017. 1−14.

[10] Harutyunyan N, Bauer A, Riehle D. Industry requirements for FLOSS governance tools to facilitate the use of open source software in commercial products. Journal of Systems and Software, 2019, 158: 110390.

[11] He DJ, Song H, Wang Q, *et al*. A study of open source license and its detection software. Computer Applications and Software, 2018, 35(6): 28−35, 53 (in Chinese with English abstract).

[12] Kapitsaki GM, Tselikas ND, Foukarakis IE. An insight into license tools for open source software systems. Elsevier Science Inc., 2015.

[13] Kapitsaki G, Charalambous G. Modeling and recommending open source licenses with findOSSLicense. IEEE Trans. on Software Engineering, 2019.

[14] Xu HB, Yang HH, Li DJ, *et al*. Study of open source license tracking process. Application Research of Computers, 2010, 27(8): 2972−2975 (in Chinese with English abstract).

[15] Vendome C, Linares-Vásquez M, Bavota G, *et al*. Machine learning-based detection of open source license exceptions. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). IEEE, 2017. 118−129.

[16] Vendome C, Linares-Vásquez M, Bavota G, *et al*. When and why developers adopt and change software licenses. In: Proc. of the 2015 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). 2015. 31−40. [doi: 10.1109/ICSM.2015.7332449]

[17] Carver BW. Share and share alike: Understanding and enforcing open source and free software licenses. Berkeley Technology Law Journal, 2010(1).

[18] Raymond E. The cathedral and the bazaar. Knowledge Technology & Policy, 1999, 12(3): 23−49.

[19] Kennedy DM. A primer on open source licensing legal issues: Copyright, copyleft and copyfuture. Louis University Public Law Review, 2001, 20: 345.

[20] Weber S . The success of open source. Library Review, 2004, 13(100): 1−6.

[21] Patterson MR. Must licenses be contracts-consent and notice in intellectual property. Florida State University Law Review, 2012, 40: 105.

[22] Mikko V. Dual licensing in open source software industry. SSRN Electronic Journal, 2003.

[23] Comino S, Manenti FM. Dual licensing in open source software markets. Information Economics and Policy, 2011, 23(3-4): 234−242.

[24] Horne NT. Open source software licensing: Using copyright law to encourage free use. 2001.

[25] Hammouda I, Mikkonen T, Oksanen V, *et al*. Open source legality patterns: architectural design decisions motivated by legal concerns. In: Proc. of the MindTrek 2010, ACM. 2010. 207−214.

[26] Zhang P, Zhang TL. The possible problems and countermeasures of the combination of open source software and commercial software. Netinfo Security, 2006(3): 69−70, 74. (in Chinese).

[27] Gobeille R. The FOSSology project. In: Proc. of the 2008 Int'l Working Conf. on Mining Software Repositories (MSR 2008) (Co-located with ICSE), Leipzig: DBLP, 2008.

[28] Obrenovic Z, Gasevic D. Open source software: All you do is put it together. IEEE Software, 2007, 24(5): 86−95.

[29] Kechagia M, Spinellis D, Androutsellis-Theotokis S. Open source licensing across package dependencies. In: Proc. of the Panhellenic Conf. on Informatics. IEEE Computer Society, 2010.

[30] German DM, Manabe Y, Inoue K. A sentence-matching method for automatic license identification of source code files. In: Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering. ACM, 2010. 437−446.

[31] Jaeger MC, Fendt O, Gobeille R, *et al*. The FOSSology project: 10 years of license scanning. Int'l Free and Open Source Software Law Review, 2018, 9(1): 9−18.

[32] Kapitsaki, Georgia M, Paschalides D. Identifying terms in open source software license texts. In: Proc. of the 24th Asia-Pacific Software Engineering Conf. (APSEC). IEEE, 2017.

[33] Feng M, Mao W, Yuan Z, *et al*. Open-source license violations of binary software at large scale. In: Proc. of the 26th Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2019.

[34] Zhang H. Comparison of open source license scanning tools. 2020.

[35] Monden A, Okahara S, Manabe Y, *et al*. Guilty or not guilty: Using clone metrics to determine open source licensing violations. IEEE Software, 2011, 28(2): 42−47.

[36] Kapitsaki GM, Kramer F, Tselikas ND. Automating the license compatibility process in open source software with SPDX. Journal of Systems and Software, 2016, 131: 386−401.

[37] Wheeler, David A. The free-libre/open source software (FLOSS) license slide. 2007. http://www.dwheeler.com/essays/floss-license-slide.pdf

[38] McLeod M. A new model of "dual licensing" for open source. Software World, 2006(14): 56−57 (in Chinese).

[39] Alspaugh TA, Asuncion HU, Scacchi W. Intellectual property rights requirements for heterogeneously-licensed systems. In: Proc. of the 17th IEEE Int'l Requirements Engineering Conf. IEEE, 2009. 24−33.

[40] Kapitsaki GM , Charalambous G. Find your open source license now! In: Proc. of the Software Engineering Conf. IEEE, 2017.

[41] Zahoor F, Bajwa IS. Automatic extraction of catchphrases from software license agreement. In: Proc. of the 6th Int'l Conf. on Intelligent Human-Machine Systems and Cybernetics, Vol.2. IEEE, 2014. 189−193.

[42] German DM, Hassan AE. License integration patterns: Addressing license mismatches in component-based development. In: Proc. of the 31st Int'l Conf. on Software Engineering. IEEE, 2009. 188−198.

[43] Gordon T. Report on prototype decision support system for oss license compatibility issues. Qualipso, 2010, 79: 80.

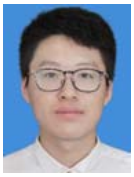
[44] Singh PV, Phelps C. Networks, social influence, and the choice among competing innovations: Insights from open source software licenses. Social Science Electronic Publishing, 2013, 24(3): 539−560.

[45] Lerner J, Tirole J. The scope of open source licensing. Journal of Law, Economics, and Organization, 2005, 21(1): 20−56.

[46] German DM, González-Barahona JM. An empirical study of the reuse of software licensed under the GNU general public license. In: Proc. of the IFIP Int'l Conf. on Open Source Systems. Berlin, Heidelberg: Springer-Verlag, 2009.
[47] Moglen E, Choudhary M. Software freeedom law center guide to GPL compliance. Software Freedom Law Center, 2014. https://softwarefreedom.org/resources/2014/SFLC-Guide_to_GPL_Compliance_2d_ed.html
[48] Wu X, Wu JY, Zhou MH, *et al*. Selection of open source license: Challenges and influencing factors. Ruan Jian Xue Bao/Journal of Software, 2021 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6279.htm [doi: 10.13328/j.cnki.jos.006279]
[49] Wang ZQ. The compliance use research on open source software licenses based on spdx license list [MS. Thesis]. Chongqing: Southwest University, 2021 (in Chinese with English abstract). [doi:10.27684/d.cnki.gxndx.2021.001811]
[50] Almeida DA, Murphy GC, Wilson G, *et al*. Do software developers understand open source licenses? In: Proc. of the 25th IEEE/ACM Int'l Conf. on Program Comprehension (ICPC). IEEE, 2017. 1–11.
[51] Cotton BJ. Impact of license selection on open source software quality [MS. Thesis]. Purdue University, 2014.
[52] Wang ZQ, Xiao GQ, Zhang ZL, *et al*. A novel model for automatic identification of open source software license terms. In: Proc. of the 4th Int'l Conf. on Computer and Communication Engineering Technology (CCET). IEEE, 2021. 212–219.
[53] Schoettle H. Open source license compliance—Why and how? Computer, 2019, 52(8): 63–67.

附中文参考文献:

[1] 金芝, 周明辉, 张宇霞. 开源软件与开源软件生态: 现状与趋势. 科技导报, 2016, 34(14): 42–48. [doi: 10.3981/j.issn.1000-7857.2016.14.005]
[3] 梅宏, 周明辉. 开源对软件人才培养带来的挑战. 计算机教育, 2017, (1): 2–5.
[11] 何东杰, 宋昊, 王琪, 等. 开源许可证及其检测工具研究. 计算机应用与软件, 2018, 35(6): 28–35, 53.
[14] 许洪波, 杨会会, 李德杰, 等. 开源许可证检测系统的研究. 计算机应用研究, 2010, 27(8): 2972–2975.
[26] 张平, 张韬略. 开源软件与商业软件结合可能产生的问题及其对策. 信息网络安全, 2006(3): 69–70, 74.
[38] McLeod M. "双重授权"的开源新模式. 软件世界, 2006(14): 56–57.
[48] 吴欣, 武健宇, 周明辉, 等. 开源许可证的选择: 挑战和影响因素. 软件学报, 2022, 33(1): 1–25. http://www.jos.org.cn/1000-9825/32/6279.htm [doi: 10.13328/j.cnki.jos.006279]
[49] 王志强. 面向 SPDX 许可证列表的开源软件许可证合规使用研究[硕士学位论文]. 重庆: 西南大学, 2021. [doi:10.27684/d.cnki.gxndx.2021.001811]



王志强(1995－), 男, 硕士生, 主要研究领域为开源软件, 开源许可证.

张自力(1964－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为人工智能, 数据分析.

伍胜(1976－), 男, 博士, 讲师, 主要研究领域为地理信息系统.

刘志有(1997－), 男, 硕士生, 主要研究领域为开源软件, 开源许可证.

肖国强(1965－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为计算机视觉, 机器学习, 大数据技术, 物联网技术.

彭景(1995－), 女, 硕士, 主要研究领域为开源生态模式及机制研究.