**中国科学院软件研究所“点亮计划”专项**

**课题任务书**

**摘要（约500字）**

**一、主要指标**

1、（总体目标及关键时间节点）

（1）开源许可证检测工具：针对开源软件复用，开发开源许可证检测工具，建立开源许可证数据库，实现对开源软件中使用的开源许可协议的检测识别。

给定软件项目文件，分析给定文件中存在或者使用的开源许可协议。许可证检测工具支持C/C++、Java、Python等多种语言开源项目的检测。同时支持源代码、压缩格式文件等多种类型的文件进行检测。建立开源许可证数据库，并保持数据更新。时间点：2020年10月份

（2）开源许可证兼容性检测工具：针对开源软件存在的多种许可证，开发开源许可证兼容性检测工具，实现对开源许可证发生冲突或不合规的情况的检测。

调研开源许可证，分析许可协议内容条款，识别不同许可证两两之间的兼容性，判断产品中使用的开源软件是否存在License冲突，对冲突原因进行分析，对存在冲突的许可证逐项列出冲突的具体项目条款。时间点：2020年11月份

（3）开源许可证风险分析工具：针对开源许可证不同条款规定，开发开源许可证风险分析工具，识别由于开源许可证引入的法律风险。

调研开源许可证，分析许可协议内容条款，识别开源许可证中各种条款的权利，限制和义务。若许可证信息发生变化，管理识别由开源许可证引入的法律风险，需要保持对应内容更新。具体条款分析要求说明，包含以下条目但不限于以下条目：商业用途、分发要求、修改要求、商标使用、专利使用、私人使用、公开来源（开源代码）、许可和版权声明、许可和版权声明来源、 网络分发使用开源要求、相同或相似许可证要求、代码修改记录要求、责任限制。时间点：2020年12月份

2、主要考核指标（见表2）

（1）开源许可证检测工具：

* 对主流的编程语言代码支持检测，支持5种以上的编程语言。优先支持C、C++、C#、Go、Java、JavaScript、Kotlin、Node.js、Objective-C、Perl、PHP、Python、R、Ruby、Scala、.NET语言。
* 覆盖的开源软件License应在50种以上，检测准确率大于80%。优先支持包含OSI认证的许可证，SPDX License List以及木兰许可证。
* 支持不低于15个检测任务并发执行；检测速度不低于20个文件每秒。
* 支持源代码、多种压缩格式文件（zip、rar、tar、gz、bz2等）等多种类型的文件进行检测。
* 支持多种使用方式，包括：命令行、可视化工具。
* 开源许可证数据库包含开源许可证超过50种，优先支持包含OSI认证的许可证，SPDX License List以及木兰许可证。支持数据进行持续的扩展与更新。

（2）开源许可证兼容性检测工具：

* 覆盖的开源软件License应在50种以上。检测准确率大于90%。优先支持包含OSI认证的许可证，SPDX License List以及木兰许可证。特别是OSI认证的流行和广泛使用的许可证。
* 可以检测出待测文件直接或间接依赖的许可证冲突。
* 支持多种使用方式，包括：命令行、可视化工具。

（3）开源许可证风险分析工具：

* 覆盖的开源软件License应在50种以上。检测准确率大于90%。优先支持包含OSI认证的许可证，SPDX License List以及木兰许可证。特别是OSI认证的流行和广泛使用的许可证。
* 支持多种使用方式，包括：命令行、可视化工具。
* 风险分析对许可证具体条款分析要求说明，包含以下条目但不限于以下条目：商业用途、分发要求、修改要求、商标使用、专利使用、私人使用、公开来源（开源代码）、许可和版权声明、许可和版权声明来源、 网络分发使用开源要求、相同或相似许可证要求、代码修改记录要求、责任限制。
* 开源许可证数据库包含开源许可证超过50种，优先支持包含OSI认证的许可证，SPDX License List以及木兰许可证。支持数据进行持续的扩展与更新。

3、验收标准

（1）开源许可证检测工具：命令行、可视化界面开源许可证检测工具、开源许可证数据库、使用手册，提供源代码、压缩格式文件标准格式输出（如json）检测结果、（HTML、WORD、PDF等形式）检测报告。

（2）开源许可证兼容性检测工具：命令行、可视化界面许可证兼容性检测工具、使用手册，提供开源许可证名称列表/标准格式输出（如json）风险识别结果、（HTML、WORD、PDF等形式）检测报告。

（3）开源许可证风险分析工具：命令行、可视化界面许可证风险识别工具、使用手册，提供开源许可证名称列表/标准格式输出（如json）风险识别结果、（HTML、WORD、PDF等形式）检测报告。

**二、研究方案**

1、研究内容及技术路线

开源许可证是软件许可证的一种特殊形式，其授权一般是针对软件的开放源代码，允许用户在承认软件原作的著作权的基础上对软件源代码的使用、修改、私用等权益，以保证开源软件能够合法地被软件开发者自由使用和共享。

开源许可证的检测可以根据“LICENSE” 文件、源代码注释中、 readme 文件中等查看软件许可证，给出在开源许可证声明的条款涉及软件权益信息，主要包括商业用途、分发、修改、专利授权、私用、公开源码、放置许可协议与版权信息、使用网络分发、使用相同协议、声明变更、承担责任、使用商标等。

另外，许可证之间存在兼容性问题和法律风险问题。根据开源许可证的法律依据和效用，参考不同开源许可证的条款，在使用开源软件时，为避免产生法律问题，主要需要考虑几方面的问题:开源要求、修改声明、专利允许、许可证兼容、商标使用、网络分发。因此开源许可证的风险可分为六个方面：商业化风险、许可证兼容风险、专利侵权风险、产权归属风险、商标使用风险、服务提供风险。

基于上述分析结果，本项目设计工具提供以下功能: 查找开源软件中包含的所有许可证和权益信息、指出开源软件中包含的不同许可证兼容 问题、给出开源发布软件的风险提示、给出商业化发布软件风险提示、给出作为服务提供使用软件风险提示、给出其他权益风险提示、自定义开源许可证添加、自定义许可证风险添加等。

图1工具整体架构，包括输入层、文件构建层、扫描检测层、报告生成层。图2是工具技术路线

**输入层**完成对源代码文件、多种压缩格式文件（zip、rar、tar、gz、bz2等）等文件管理工作，并留出扩展接口，未来可处理二进制文件、镜像文件、jar文件、rpm文件、deb文件、bin文件、apk文件等其他类型的文件。

**文件构建层**包含文件解压缩器、仓库文件下载器、文件目录构建器。其中，文件解压缩器实现对zip、rar、tar、gz、bz2等格式的文件进行解压缩。仓库文件下载器实现从远程仓库到本地的复制。文件目录构建器对于整个文件夹进行遍历，然后构建文件夹的树形结构，以 JSON 数据形式保存。

**扫描检测层**包含了扫描调度器、文件扫描器、许可证语义构建器、冲突及风险检测器。文件扫描器根据许可证语义构建器提供的关键字，对扫描调度器指定的文件进行语义匹配，匹配到对应的关键字之后， 将关键字的位置及许可证或者版权的标示保存到结果中。

扫描调度器根据文件目录构建器给出的文件结构，按照深度优先的策略，调度文件扫描器进行扫描， 并将文件名称和扫描的结果以字典的形式保存在字典中，供冲突及风险检测器使用。为了提高文件扫描效率和速度，扫描调度器会启动多个文件扫描器的线程，按照MapReduce思想，由文件树最底层叶子节点开始，向上分别对不同的文件进行扫描，待所有文件扫描器的线程将所有文件扫描结束后，产生中间结果; 然后，扫描调度器会启动多个结果合并器的线程，以文件路径作为键，扫描结果作为值，将扫描结果进行合并，得到输出。许可证语义构建器，首先在许可证目录中将所有许可证逐个按照内容进行切分。切分完成后，将许可证名称和对应的切分内容按照键和内容的形式构建字典，待所有许可证都切分并且构建完毕之后，将字典返回给文件扫描器供实际扫描过程中使用。 冲突及风险检测器首先根据我们的研究结果构建了许可证风险库和不兼容库，实际使用中，用户可以对这个库进行自定义的添加、修改、删除等操作。风险库包含了商业化风险、专利侵权风险、产权归属风险、商标使用风险、服务提供风险，风险库中包含了此种风险的名称和其对应的许可证的键。兼容库中包含了所有强许可证( 即衍生作品必须使用本许可证并且要求开源) 的键和兼容风险名称。冲突及风险检测器根据扫描调度器提供的字典，在库中对风险和兼容性规则进行匹配，匹配到风险或者兼容性问题后便将许可证的键和风险类型保存在字典中，待所有扫描调度器提供的字典扫描完毕后，将结果的返回给报告生成器。

**报告生成层**输出扫描检测分析结果，即开源许可证检测结果、兼容性检测结果、风险识别结果，将结果填充到 HTML、WORD、PDF模板，形成最终检测报告。报告包含了开源软件许可证兼容问题、开源发布软 件风险、商业化发布软件风险、作为服务提供使用软件风险、其他权益风险以及文件中包含的许可证详细的官方信息等。

对于三个检测工具都设计对应的命令行版本和用户交互界面版本，用户界面设计可采用QT、Duilib或Web页面。

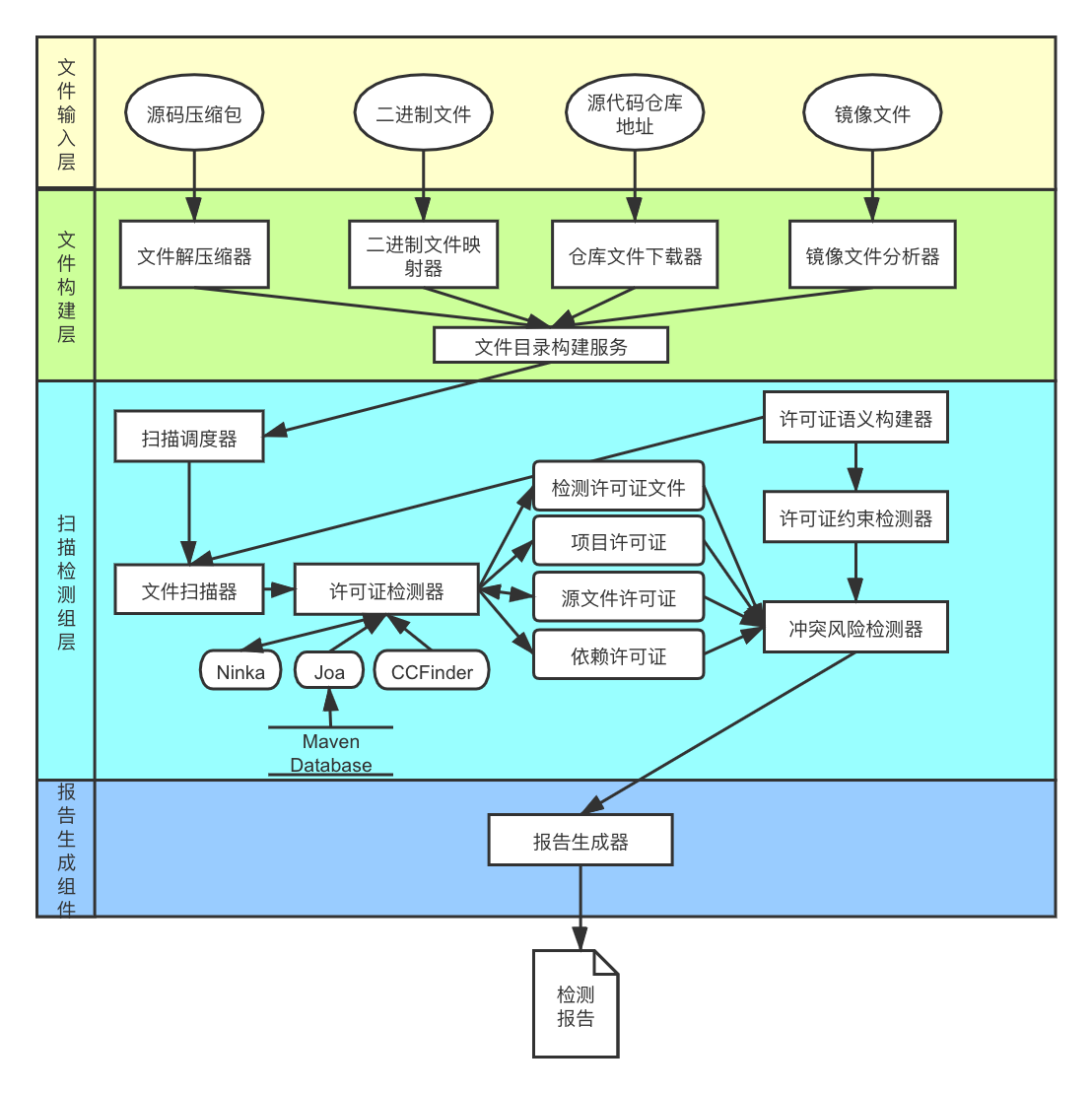


图1 工具技术整体架构

2、（核心科学技术问题与创新点）

**（1）开源许可证高精度检测方法**

对于开源许可证的检测，现有技术中一般基于关键字匹配的方法实现，存在识别精度不高的问题，会出现许可证漏查、误差的情况。

采用基于正则表达式的方法，在一定程度上可以提高开源许可证识别精度，但是运用正则表达式的方法需要人为设置大量规则，且易出现对所识别开源许可证无法分类的情况。本项目提出基于文本相似度的高精度开源许可证识别方法，识别潜在开源许可证信息，降低开源许可证漏查、误查的发生机率。

现有的基于文本相似度识别方法，通常做法是：针对不同开源许可证的文本特性定义将文本表示为K-shingle的集合，并使用Jaccard 算法计算检测文本与数据库中各开源许可证的相似度。但Shingle对文本语义特征的捕捉能力依赖于K参数，K越小，K-shingle对文本语义的捕捉能力越差，K越大，不同词汇的组合能反应出越来越多的语义层面的特征，此外，随着K增大，文本特征向量的维数也急剧增加，由于特征向量维数的增加，大大降低了特征向量之间距离的计算效率。对海量文本而言，K-shingle算法得到的特征向量是超高维的，导致该算法非常巨大的时间复杂度和空间复杂度，而且对海量文档集计算两两之间的Jaccard相似度是不现实的。因此，本项目的创新点之一是设计低复杂度高精度源许可证识别方法，其核心思路是设计了一种最小哈希函数，将原始超高维的稀疏向量转化为低维的稠密向量，降低计算的空间复杂度，同时，对转换后的稠密向量进行分段索引，缩小潜在相似许可证文本范围，降低计算的时间复杂度，并提高识别精度。

**（2）开源许可证兼容性建模**

两个自由／开源软件许可证兼容是指如果包含两个许可证组件的软件可以在许可证V2下获得许可证，可以认为许可证V1与许可证V2“单向兼容”。因此，根据这一基本原理，可以使用有向图G（V，E）表示已知许可及其兼容性，每个顶点V代表一个特定许可证及其版本，每个边E连接两个兼容许可证V1和V2（方向从V1到V2）。

由于许可证数量巨大，各自的许可证文本又存在很大差异，在理想情况下，检查OSI和自由软件基金会（FSF）认可的整套许可证的兼容性在实践中并不可行。因此，如何建立海量许可证兼容性模型是兼容性检测要解决的关键科学问题。本项目在海量许可证兼容性建模方面的创新点是，基于许可证热门度构建许可证兼容性图，通过遍历兼容性图来定位许可证冲突。主要做法是，首先根据机器学习算法选择流行许可证，即在相关在线存储库（如SourceForge和BlackDuck的知识库）中是热门的，然后基于这些热门许可证创建了许可证兼容性图，进而在许可证兼容性图遍历定位许可证冲突。

难点在于，**一是构建图的兼容性分析的信息来源是多样异构的**，其中包括维基百科关于许可证的文章条目、来自许可证文本和论坛的信息，以及GNU托管的关于许可证与GPL兼容的页面等；另一难点是在**许可证兼容性方面存在不明确的情况**，即不同的组织对许可文本提供了不同的解释，将许可划分为不同的类别（弱许可，强许可等）；另外，由于数据库的许可证兼容性冲突规则库中规则是不断增加的，且**同一规则可能存在多种变体，因此每次分析都遍历整个许可证兼容性冲突规则库使得检测分析的效率十分低下**。我们通过建立分类索引的方式对分析方式进行优化，根据涉及开源许可证类型的不同对冲突规则库设置分类索引，通过该索引列表快速定位与特定开源许可证相关联的规则集合，提升分析效率，具体的分类索引方法可以采用基于图的索引结构，用邻接表的形式存储图，定义表头为开源许可证或其他实体，边为不同开源许可证之间的规则。

检查器检查利用所构建的许可证兼容图，即对给定图中的两个顶点V1和V2，确定从V1到Vx的路径p1和从V2到Vx的p2是否都存在的方式来判断一个许可证是否可以与另一个许可证相结合，或者两个许可证是否可以在这两个许可证中的一个许可证下或另一个不同的许可证组合到一个新的软件包中。

1. 开源许可证风险分析策略

建立开源治理平台辅助开源治理规范，在开源软件许可证识别和兼容性分析的基础上，管理开源软件的使用情况及依赖情况，掌握现有系统中开源软件引入信息，为后续管理和治理打好基础。

通过平台自动化实现企业内部开源软件的引入评测、使用管理、运行维护等，跟踪和记录开源软件全生命周期动态，同时提升开源治理工作效率和准确性。支持定期维护开源软件的情况，根据开源软件的社区情况、版本更新情况、开源许可证变更情况、社区发展情况及安全漏洞情况等，定期对相关信息进行维护、分析、评估和处置。如出现开源软件许可证发生变更，尤其由宽松型许可证变更为严格型许可证时，及时评估许可证变更带来的各方面风险，并对其传染性进行评估和判断。

（4）灵活可定制的报告生成

根据本系统各个环节均有报告生成的需求，提出一种通用报告自动生成的机制，可满足不同业务过程对报告生成的需求，供不同产品复用。该报告生成功能具有灵活可定制的特点，避免二次开发数据获取的代码，提升生成报告效率。图中给出了通用报告生成功能的流程，工具提供模板描述语法、数据集模型、数据源配置文件及报告生成器（HTML、Word或Pdf）几个部分（绿色标识）。此外，查询接口用于实现对数据源的自定义检索，产生基于数据集模型的数据集。





参考项目：

1. OSLC（Open Source License Checker）
   1. 2007年
   2. java
2. ScanCode-Toolkit（<https://github.com/nexB/scancode-toolkit>）
   1. 基本能够满足（1）、（2）部分的要求
   2. Python