单位代码 10006
学　　号 38020326
分 类 号 TN953

北京航空航天大學
BEIHANG UNIVERSITY

毕业设计(论文)

软件架构设计图与代码追踪工具的设计与实现

学 院 名 称　计算机学院
专 业 名 称　计算机科学与技术
学 生 姓 名　赖慧琴
指 导 教 师　连小利

2024 年 9 月

论文封面书脊

软件架构设计图与代码追踪工具的设计与实现

赖慧琴

北京航空航天大学

北京航空航天大学

本科生毕业设计（论文）任务书

Ⅰ、毕业设计（论文）题目：

软件架构设计图与代码追踪工具的设计与实现

Ⅱ、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

使用的原始资料是在 Gitee 平台上发布的开源代码包括 RSE-CDI、RSE-SDI、CDTC 及 SDTC 算法。设计技术要求图必须符合 UML2.5 标准，结合上述算法并使用 Web 技术开发符合用户习惯且容易上手的追踪工具。

Ⅲ、毕业设计（论文）工作内容：

本文主要的工作内容如下：

(1)调研设计模型图语义元素识别技术，以确保能够识别具有语义、符合 UML2.5 标准并以位图形式存储的类图和顺序图。

(2)调研设计模型图与代码追踪构建技术，以便能够构建类图与 Java 代码间的追踪关系以及构建顺序图与 Java 代码间的追踪关系，并生成追踪报告。

(3)调研具有类似功能的工具，分析它们的优劣，并提出本工具的创新点。

(4)调研 Web 框架与库，选择更合适的 Web 技术以实现追踪工具的两大核心功能：设计模型图语义元素识别功能和设计模型图与代码追踪构建功能。

(5)确定了所使用的图识别算法及图代码追踪算法后，集成这些算法，使算

法可以与本 Web 工具交换数据。

Ⅳ、主要参考资料：

[1] F Chen, L Zhang, X Lian, et al. Automatically recognizing the semantic elements from UML class diagram images[J]. Journal of Systems and Software, 2022, 193: 111431.

[2] F Chen, L Zhang, X Lian. CDTC: Automatically establishing the trace links between class diagrams in design phase and source code[J]. Softw: Pract Exper. 2024; 54(2): 281-307.

计算机 学院 计算机科学与技术 专业类 200615 班

学生 赖慧琴

毕业设计（论文）时间：2023 年 12 月 01 日至 2024 年 09 月 05 日

答辩时间：2024 年 09 月 05 日

成　　绩：

指导教师：连小利

兼职教师或答疑教师（并指出所负责部分）：

6 系（教研室） 主任（签字）：

注：任务书应该附在已完成的毕业设计（论文）的首页

本人声明

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者：赖慧琴

签字： 

时间：2024 年 9 月

软件架构设计图与代码追踪工具的设计与实现

学　　生：赖慧琴

指导教师：连小利

摘　　要

架构设计图在软件开发过程中占据着重要的位置，它不仅为系统提供一个清晰的蓝图，确定了系统的总体架构，也关注到了模块和组件的具体实现细节，它是实现高质量软件产品的方法之一，对于要了解系统总体架构的开发人员来说更是一个利器。

通过调研开源社区发现，大项目的设计图多以位图形式存储且具备高抽象层次的特点。同时，目前没有公开工具可以自动建立这种特点的设计图与代码之间的追踪关系，开发人员需手动建立，费时费力。为节省时间成本，本文结合了图识别算法及图与代码间追踪算法，分析了现有工具的优劣，开发了软件架构设计图与代码追踪工具。

由于图识别算法不能完整且准确地将图片识别出来，本文研究了 GoJS 库的使用，提出了图编辑功能，让用户可以编辑图片，进一步修改错误的图信息。此外，为了减少错误的图信息，本文重点识别由 StarUML 绘制的符合 UML2.5 标准的类图，对 RSE-CDI 算法进行优化，通过评估结果证实优化算法提升了特定类图识别的准确率和召回率。

本文开发的是 Web 工具，而算法仅支持处理本地文件。为了让工具和算法之间能够互相传输数据，本文在算法的基础上搭建了一个 Tomcat 服务器。转换访问文件的方式后需要考虑几个问题：(1)算法中原先的文件格式是否仍然支持；(2)大项目的文件数量通常很多，而网络传输的文件数量有限，在传输过程中是否会过载。分析了上述问题后，本文对数据格式进行了转换，并进行了数据预处理和数据批处理，更好地解决了 Web 工具与算法之间的数据传输问题。

为实现本工具的功能，本文做出了以下贡献：(1)GoJS 库具有高定制性，但是有关资料较少，尝试了各种方法才得以实现复杂的图编辑功能；(2)采用 Zustand 库管理数据状态，以实现标签头与标签页、文件路径名称与文件内容的一一对应效果，在进行此处理之前，需要将用户上传的文件列表转换成 Map 结构；(3)为方便用户查看展示区域，开发了折叠面板组件及拖拽组件，并将其封装成开源库。

综上所述，本文开发了一个半自动工具，需要依靠用户手动编辑图片，但是与手动建立追踪相比，节省了更多的时间和人力成本。同时，本工具为建立高抽象层次设计图与代码追踪方面的工具奠定了基础，开拓了新市场。

关键词：软件架构设计图，UML 图，类图和顺序图，Java 编程语言，设计图和代码间追踪

Design and Implementation of Software Architecture Design Diagram and Code Tracing Tool

Author : LAI Hui-qin
Tutor : LIAN Xiao-li

Abstract

The architectural design diagram holds an important position in the software development process. It not only provides a clear blueprint for the system and defines the overall system architecture but also addresses the specific implementation details of modules and components. As one of the methods to achieve high-quality software products, it is a powerful tool for developers who need to understand the system's overall architecture.

Through research into the open-source community, it was found that design diagrams of large projects are often stored as bitmaps and typically exhibit a high level of abstraction. Currently, there is no public tool that can automatically establish a traceability relationship between such design diagrams and the code. Developers are required to establish this relationship manually, which is time-consuming and labor-intensive. To reduce time costs, this paper combines image recognition algorithms with diagram-to-code traceability algorithms, analyzes the advantages and disadvantages of existing tools, and develops a software architecture design diagram and code traceability tool.

Since image recognition algorithms cannot fully and accurately recognize images, this paper explores the use of the GoJS library and introduces a diagram editing feature that allows users to manually edit images and further correct erroneous diagram information. Moreover, to minimize incorrect diagram information, this study focuses on recognizing class diagrams drawn using StarUML that conform to the UML2.5 standard. The RSE-CDI algorithm is optimized, and evaluation results confirm that the optimized algorithm improves the accuracy and recall rate of specific class diagram recognition.

The tool developed in this study is a web-based tool, but the algorithm only supports

processing local files. To enable data transfer between the tool and the algorithm, this paper sets up a Tomcat server on top of the algorithm. After modifying the way files are accessed, several issues need to be considered: (1) whether the original file format used by the algorithm is still supported, and (2) large projects often involve numerous files, and there are limitations on the number of files that can be transferred over the network. Could this cause an overload during transmission? After analyzing these issues, this study converts the data format and implements data preprocessing and batch processing to better address the data transmission issues between the web tool and the algorithm.

To implement the tool’s functionality, this paper makes the following contributions: (1) The GoJS library offers a high degree of customizability, but there is limited documentation available. Various methods were attempted to implement the complex diagram editing functionality. (2) The Zustand library is used to manage data states, ensuring a one-to-one correspondence between tab headers and tab pages, as well as between file path names and file contents. Before doing this, the user-uploaded file list must be converted into a Map structure. (3) To improve the user experience, a collapsible panel component and drag-and-drop component were developed, both of which were packaged into an open-source library.

In conclusion, this paper developed a semi-automatic tool that requires users to manually edit images. However, compared to manually establishing traceability, this tool significantly saves time and labor costs. Additionally, this tool lays the foundation for establishing traceability between highly abstract design diagrams and code, opening up new market opportunities.

Key words: Software Architecture design diagram, UML diagram, Class diagram and sequence diagram, Java programming language, tracing between design diagram and code

目　　录

1 绪论

1.1 研究背景与意义

在软件开发过程中，软件架构设计非常重要。一个良好的软件架构设计可以使开发人员更了解系统的总体架构、要开发的模块和组件的具体细节，使项目管理者和开发团队对系统的关键决策和设计有共同的理解[7]，使项目更易于维护和可扩展。软件架构设计至上而下分为概要设计、详细设计两层，越上层的越抽象，越下层的越具体。概要设计抽象层次较高，主要为系统提供一个清晰的蓝图，确定系统的总体架构和组织方式，更关注模块间的逻辑关系。而详细设计则在概要设计的基础上进一步细化模块和组件的具体实现细节。这两个阶段是相互依赖、逐步深入的，共同确保软件系统的设计既符合高层次的业务需求和技术目标，也关注到了实现的具体细节，最终实现高质量的软件产品。

事实上，通过开源社区可发现大规模项目多以位图形式保存设计模型图且所设计的模型图通常抽象层次较高，并不会与代码一一对应，因为对于开发人员来说迭代开发是常态，通常只更新代码，并不会耗费时间更新模型图。同时没有必要使模型图过于具体，它不但会增加模型图的复杂度让人难以阅读，也不利于后续新来的开发人员快速了解系统的总体架构。

据调研，近 10 年来发表在主流期刊（如 ESE、JSS、IST、SPE[12]等）上的一些研究提供了来自业界（如 Microsoft、IBM 等）和开源社区的知名组织（如 Eclipse、Apache 等）的许多案例，这些案例普遍是基于非模型驱动开发，且常采用 UML 等半形式化模型做设计，最终生成的设计模型多以位图形式嵌入文档中，并缺乏与代码之间的追踪关系[6]，而市场上没有公开工具来自动建立设计模型与代码的追踪关系，对于想要快速熟悉代码的人员来说，只能人工建立追踪关系，费时费力，同时代码规模大且模型与代码间可能是多对多映射，导致手动建立设计模型与代码之间的追踪关系容易出错，劝退了无数人。

总的来说，目前没有公开工具可识别以位图形式存储且具有语义的设计模型图，也没有公开工具可以在设计模型图与代码存在层次差异的情况下建立模型图与代码之

间的追踪关系，进而生成追踪结果。所以本文为了让开发人员可以快速了解大规模项目的总体架构，决定开发软件架构设计图与代码追踪工具。由于软件架构设计图与代码所涉及的范围广阔，不易于初步研究，因此本文采用了“先精后广”策略，先确定具体的研究对象，琢磨图与代码追踪的具体细节，后续再扩大研究对象，让本工具可以支持更多的软件架构设计图与代码。首先，通过 F Chen 等人的调研结果，知道目前常用的设计模型图主要是类图和顺序图，而广泛使用的编程语言是 Java，主流建模工具包括 StarUML、Enterprise Architect、Rational Rose。其次，在实际项目中，还存在许多不同画法的设计模型图。例如，实线和实心三角形、实线和箭头等画法在不同的项目中可能表示不同的关系类型。因此，本文将选择一个标准作为基准，以确定图信息语义的一致性。最终，本文决定聚焦于由 StarUML 建模工具绘制的、符合 UML2.5 标准的类图和顺序图，优先解决这些图与 Java 代码之间的追踪问题。

1.2 相关研究现状

针对所确定的研究对象，本文将展开以下研究工作：

(1)调研设计模型图语义元素识别技术，以确保能够识别具有语义、符合 UML2.5 标准并以位图形式存储的类图和顺序图。

(2)调研设计模型图与代码追踪构建技术，以便能够构建类图与 Java 代码间的追踪关系以及构建顺序图与 Java 代码间的追踪关系，并生成追踪报告。

(3)调研具有类似功能的工具，分析它们的优势与劣势，并提出本工具的创新点。

1.2.1 设计模型图语义元素识别技术

在软件设计的早期阶段，软件工程师通常会在白板上非正式地绘制 UML 图形，为了解决从早期的草图过渡到电子形式问题，E. Lank 等人提出了在线 UML 图形识别系统[16]。论文中提到的系统接受的输入来源包括电子白板、数据平板或鼠标，该系统的核心是可重新定位的，为任何基于符号的图形符号提供一般的前端识别功能。系统特别扩展了用于 UML 图形的特定例程，包括分段、识别图形符号及其关系。

Hammond 等人提出了一个用于识别 UML 类图中几何图形草图的 Tahuti 系统[18]。该系统采用了多层次的识别框架，通过识别几何特性来识别多笔触对象，从而使用户能够自由地自然绘制，而不需要用户按照预定义的方式绘制对象。用户可以在查看原

始笔画或其解释版本时进行绘制和编辑，从而在草绘过程中产生用户自主性。

Karasneh 等人提出了 Img2UML 工具[17]，可以接受输入为 JPEG 等各种位图格式的 UML 图，并使用图像处理技术来识别图像中的 UML 符号和结构，将这些视觉元素转换为 XMI 格式或其他 UML 建模工具支持的格式，以便进一步分析和编辑 UML 模型格式。

为解决不同工具绘制的类图语义元素样式不一、图片模糊导致识别错误等问题，F Chen 等人提出了针对 UML 类图的 RSE-CDI 方法[1]和针对 UML 顺序图的 RSE-SDI 方法。这两种方法主要基于图像处理技术，专门设计用于从 UML 图的图像中提取语义元素，该方法考虑了 UML 图的特点，包括不同工具和样式的影响，以及低分辨率对检测的影响。

综合来看，目前的在线 UML 图形识别系统及 Tahuti 系统主要专注于识别通过电子白板或手绘形式绘制的 UML 草图。然而，这些系统与本文的需求不符，本文需要识别由建模工具绘制的 UML 图，而且 Tahuti 系统仅能识别 UML 类图，缺乏对其他类型 UML 图的支持，因此可扩展性较差。虽然 Img2UML 工具可以识别以建模工具绘制且以位图格式存储的 UML 图，但目前无法找到该工具的源码和安装地址，因此无法直接利用。RSE-CDI 和 RSE-SDI 这两种方法与本文的要求更加贴切，它们能够自动识别不同分辨率、使用主流 UML 图绘制工具绘制的类图和顺序图中的语义元素，并生成相应的语义模型。经过主流工具和开源社区的实验验证，这两种方法的召回率和准确率均达到了约 90%，相比其他方法，其识别准确率都较好。

1.2.2 设计模型图与代码追踪构建技术

Antoniol 等人[2]提出了一种半自动化方法，基于名称相似度来建立 UML 类图与 C++代码之间的追踪关系。该方法需要人工建立一个词典，包含设计文档与代码中的同义词。

CCUJ[19]可以建立 UML 类图与 Java 代码间的追踪关系，但该方法所涉及的类图与代码抽象层次十分接近（比如类的名称、甚至属性的名称几乎一样）。

F Chen 等人提出一种设计阶段类图与代码追踪方法 CDTC[3]，自动在设计阶段建立抽象层次较高的类图（或类图的实例对象图）与代码（其语义与类图有差距）之间的追踪关系。另外，CDTC 方法同样适用于类图与代码抽象层次差异不大的情形。同

时，也提出了一种顺序图-代码追踪方法 SDTC，自动建立设计阶段顺序图与代码（其语义与顺序图有差距）之间的追踪关系。

综合来看，半自动化方法需要依靠人工建立词典且其建立图与 C++代码之间的追踪关系，与本文要求的 Java 代码条件不符。CCUJ 仅能建立与代码抽象层次接近的追踪关系，与本文要求的高抽象层次条件不符。CDTC 方法通过对设计文档的统计分析以扩展 UML 类图的语义，自动建立设计阶段的类图与代码之间的高抽象层次追踪关系。在五个开源项目上的评估显示，CDTC 方法的总召回率超过 94%，F2 值超过 88%；SDTC 方法专门针对顺序图与代码之间的追踪，考虑到消息发送方式的差异和语义不一致问题，自动建立设计阶段的顺序图与代码之间的追踪关系。在三个开源项目的顺序图评估中，SDTC 方法的召回率和 F2 值均超过 87%。CDTC 和 SDTC 这两个方法对比其他方法，更符合本文要求，其评估结果也在可接受范围内。

1.2.3 代码分析工具

Hammad 等人开发了 srcTracer 工具[4]，该工具可以自动确定给定源代码更改是否影响系统设计（即 UML 类图）。随着源代码的发展，代码到设计的可追溯性能够得到一致的维护。该方法使用源代码更改的轻量级分析和语法差异来确定更改是否改变了抽象设计上下文中的类图，最终生成一个追踪结果报告。

曾一等人针对模型与代码之间不一致性问题，提出了一种基于 UML 模型与 Java 代码的一致性检测方法[10]，用于检测类间静态信息以及时序调用图 SD-CG 与方法调用图 CG 间的一致性。最后，通过开发 UML 模型与 Java 源代码一致性检测工具，验证了该方法的有效性。

王雷等人针对模型与代码一致性检测缺少动态分析问题，提出了一种基于有限状态机（FSM）的 UML 模型与代码一致动态性检测方法[9]，通过执行源代码捕获方法调用跟踪，并将其与由 UML 顺序图转换得到的 FSM 匹配，以检测动态行为的一致性。

余双双等人针对 UML 模型与代码生成过程中常见的不一致问题，特别是多态性导致的执行路径不确定性，提出了基于 UML 模型的多态性与 Java 接口代码信息一致性检测的方法[8]。该方法以 UML 模型为基准，通过解析预处理 UML 模型类图、时序图和 Java 接口代码，获取并扩展时序调用图和代码调用图以考虑多态性。通过比对模型与代码信息，能够准确检测并修正不一致问题，从而提高模型与代码一致性检测的

有效性和精确度。

上述工具的主要功能是生成结果报告，没有其它功能。而且一致性分析工具的核心思想是基于与代码抽象层次接近的 UML 图信息，通过对比图信息与代码信息来生成一致性结果。

1.3 研究目标

总结上述研究意义与研究现状，本文的研究是为了协助用户快速了解项目的总体架构，解决无公开工具建立抽象层次高且以位图形式存储的设计模型图与代码之间的追踪关系的问题，并锁定研究对象为设计模型图中符合 UML2.5 标准的类图和顺序图，编程语言中的 Java，主流建模工具中的 StarUML，旨在开发一款简单易用、符合用户习惯、扩展性强的设计模型图与代码追踪工具。借助 RSE-CDI 和 RSE-SDI 算法实现设计模型图语义元素识别功能；借助 CDTC 和 SDTC 算法实现设计模型图与代码追踪构建功能。该工具相比其它工具具有以下创新点：

(1)其它工具没有图形用户界面，仅生成结果报告，而本工具是一个提供了图形用户界面的 Web 工具，用户可以通过直观的图形界面在网页上进行交互操作。

(2)srcTracer 基于代码变更自动检测设计图是否需要变更，而本 Web 工具支持用户编辑设计模型图、设计文档或代码文件，并自主决定在什么时候进行设计模型图与代码追踪构建。

(3)一致性分析工具基于确定的 UML 图信息自动检测代码与图信息是否一致，只能执行设计模型图与代码抽象层次接近的一致性分析工作，而本 Web 工具不仅可以执行设计模型图与代码抽象层次接近的一致性分析，还可以执行设计模型图与代码抽象层次较高的追踪工作。

1.4 论文的组织结构及研究内容

本文的组织结构如下：

第一章 绪论。主要介绍了研究背景与意义，透露了软件架构设计在软件开发过程中的重要性，其是如何协助开发人员有条不紊地推进工作；锁定了具体的研究对象及调查了目前的相关研究，确定了研究方向和目标。

第二章 RSE-CDI 算法精度优化。RSE-CDI 算法旨在包容各种建模工具所绘制的

类图画法。然而，这种广泛的包容性会降低对特定类图画法识别的准确率和召回率。为了遵循本文一开始提到的“先精后广”策略，本文决定首先针对特定类图进行识别，以提升特定类图的准确率和召回率，通过分析 RSE-CDI 算法的逻辑，重点识别由 StarUML 工具绘制的符合 UML2.5 标准的类图，发现其中尚有改进之处，于是分别对矩形识别、关系线识别及关系符号识别进行了优化。

第三章 需求分析与架构设计。首先基于用户需求和创新点，进行需求分析，主要有项目模块、设计模型图语义元素识别模块及设计模型图与代码追踪构建模块。接着根据需求分析，介绍了工具的架构设计，描述了功能模块层、数据传输层以及技术支撑层。

第四章 实现与测试。首先借助功能流程图及 UI 界面详细介绍了各个功能的具体实现。接着进行了测试设计并对各个功能进行测试，通过测试结果确保工具的完整性和可用性。最后验证 RSE-CDI 算法优化有效性及使用工具后的效率有效性。

结论。总结了本文的核心思想，并介绍了本文解决的问题以及所做的贡献。

2 RSE-CDI 算法精度优化

RSE-CDI 算法[1]是本工具实现类图识别功能的核心算法，其核心思想是从各种建模工具所绘制的 UML 类图中识别关键语义要素，例如类及类之间的关系。算法设计时考虑包容不同的类图编辑工具的输出，可是这种做法会导致特定类图的识别效果较差。因此，本章旨在通过研究图识别知识，转换其核心思想，采用“先精后广”策略，针对特定类图识别进行优化，以提高特定类图的准确率及召回率。本文选择的特定类图是由 StarUML 建模工具绘制的符合 UML2.5 标准的类图。

2.1 RSE-CDI 算法概述

RSE-CDI 算法的核心思路主要包括两部分：(1)对 UML 类图中类的识别。包括代表类的矩形以及类矩形中的文字（包括名称、属性和方法）；(2)对类之间关系的识别。包括关系线识别、对关系与其所属类的整合以及关系类型识别。

由于图片的清晰程度可能不够从而影响识别，所以在进行识别之前，需要先对图片进行预处理，从而可以降噪（如关系线旁边的噪点、类矩形旁边的短线段等），提升识别效果。

在预处理之后，按以下顺序识别图中的语义元素：首先识别类矩形，然后识别关系线，最后识别关系符号。由于关系类型符号所属的关系需要在识别完关系线之后才能确定，因此关系识别划分为对关系线的识别和对关系类型符号的识别这两个阶段。

考虑到不同的类由于内容差别而导致其可能由不同数目的小矩形组成，从而使类的内容识别受到挑战，F Chen 等人提出一种矩形聚合方法，将小矩形按照语义顺序聚合为一个完整的类，聚合后的矩形将按照从上到下的顺序依次包含类的名称、属性、方法。

由于目前的方法难以识别折线（特别是倾斜的折线），所以 F Chen 等人提出一种基于斜率检测技术的折线聚合方法来识别类图中倾斜的折线。完成对关系线的识别之后，再通过识别关系类型符号的方式识别每条关系的类型，将关系类型符号与关系线组合成为一条完整的关系。F Chen 等人提出一种能够识别不同保存质量的（包括分辨率低、或形状不标准等，比如继承关系的符号保存为不标准的三角形）、不同形状的

关系类型符号的方法。最后识别类图中每个类的文字内容，并将识别出来的文字内容与其所属类整合为一个完整的类。在识别完所有语义元素之后，F Chen 等人将识别结果整合并记录到一个文本文件中，从而代表所输出的类图的语义模型。

以上为 RSE-CDI 算法的核心思路，同时 RSE-CDI 算法借助了 Tesseract-OCR 库来识别文字以及 OpenCV 图像处理库来识别图形。对原有算法逻辑及图识别知识有了简单的认识之后，本文选取由 StarUML 绘制的具有透明背景且符合 UML2.5 标准的一张类图，进行实验，通过调整阈值，修改算法中的部分方法，以提升具有这种特性的类图的召回率和准确率为目标，对算法进行精度优化，主要针对矩形识别、关系线识别及关系符号识别三个部分进行精度优化。

2.2 矩形识别优化

不同的建模工具有不同的样式，StarUML 建模工具绘制的 UML 类图有多种背景包括透明背景和白色背景。在 StarUML 中，如果透明背景被设置为默认背景，则类图中类的矩形外围有阴影效果，阴影的功能主要是增强视觉效果。而 RSE-CDI 算法忽略了 StarUML 类图的透明背景的阴影效果，导致在识别过程中默认将阴影效果识别成了矩形形状，从而影响了类名、类属性、类方法三个矩形的划分以及在单个矩形上文字识别库的识别。下面将对这个问题进行优化。

2.2.1 原有算法逻辑

由于 F Chen 等人的识别工作主要关注于形状、线条、文字的大小和位置，而图形的背景颜色不提供有用的语义信息，所以算法中会先对图片做灰度处理，将每个像素的 RGB 三个通道上不同的值转换为一个通道上的值即灰度值。灰度处理后的图片仍然保留了对语义识别而言必要的信息（比如形状、大小等），为下一步识别工作奠定了基础。

2.2.2 优化方法

原有算法忽略了图像可能包含透明背景（alpha 通道）且含有阴影效果的情况，导致 OpenCV 在灰度处理时将透明背景转换成黑色，将阴影效果转换成白色，形成一种矩形的形状，从而影响后续的识别工作。

OpenCV 将透明背景转换成黑色的原因如下：OpenCV 在处理彩色图像时，通常

将透明部分视为 0 值（即黑色）。当将透明背景的图像转换为灰度图像时，透明部分的像素值会保持为 0，因此呈现为黑色。这是因为在处理透明度时，OpenCV 通常将透明度值与图像的像素值相乘，以确定最终的像素值；OpenCV 将阴影效果转换成白色的原因如下：OpenCV 在灰度处理时，会将整个图像中的每个像素都考虑在内，包括阴影效果。阴影效果通常是由灰色的像素组成的，而且这些像素与类图的其他部分相比可能具有更高的像素值。因此在灰度处理过程中，阴影效果的像素值可能会被映射到较高的灰度级别，最终呈现为白色。

本文的优化方法是在灰度处理之前，检查图片是否是四个通道 RGBA，RGBA 颜色模型中的 R 表示颜色中的红色成分，G 表示颜色中的绿色成分，B 表示颜色中的蓝色成分，A 表示颜色的透明度。如果是四个通道 RGBA，则将透明背景转换成白色背景，覆盖阴影效果，如此在灰度处理中，就不会将透明背景的阴影效果识别成矩形的形状，从而避免产生矩形划分错误导致文字识别错误问题，提高了类图识别的准确率和召回率。

2.3 关系线识别优化

RSE-CDI 算法在识别线条的过程中会因为噪点未清除干净等因素识别出多条线条，可能是同一条关系线但是产出了多条相似且不必要的线条，所以需要去除冗余线，核心思路是将两条相似的线条合并成一条新的“长直线”。但是在处理过程中忽略了新的线条可能会与之前的线条相似并且没有进行合并造成冗余线未去除干净问题，进而影响了类关系识别结果。下面将对这个问题进行优化。

2.3.1 原有算法逻辑

原有算法逻辑提到在线条识别过程中会遇到非常多难以预知的情况，这些情况都是由各种形式的噪点引起的，而这些噪点会引发冗余线、交叉线、断裂线等不是真实线条的情况出现。由于噪点的出现难以控制，所以这些情况也非常难预测。为了应对这些情况，F Chen 等人提出了多种解决方案。本文仅介绍其中的两种情况：(1)冗余线；(2)交叉线。针对这两种情况，采用斜率和长度检查方法，并基于经验设置阈值大小，判断两条线的斜率是否接近、两端是否靠近、是否重合，从而决定是否将其合并为一条线。下文将斜率接近或两端靠近的两条线条统称为相似的线条。

如算法 2.1 所示，将两条相似的线条合并为“最长”的一条线，即以线条 i 为目标，若线条 j(i < j)与线条 i 相似，则线条 i 与线条 j 合并成新的一条线并代替线条 i。这个操作使线条 i 的斜率和长度发生了微小的变化，可能会与前面的线条（如 i - 1）相似。在原有算法逻辑中，继续往后执行，取下标 i + 1 的线条和下标 j(i + 1 < j)的线条作相似性比较，忽略了线条 i + 1 可能与线条 i - 1 相似的情况，导致有多余的线条出现，进而影响了类关系识别。

算法 2.1 关系线识别的部分原有逻辑

输入: 数组 line_segments[1..n], 数组长度 length
输出: 去除冗余线的数组 line_segments[1..m]

```
For i ← 0 to length Do // 遍历 line_segments
  For j ← i + 1 to length Do
    If line_segments[i] == line_segments[j] Then // 两条线近似相等(斜率接近或两端接近)
      line_segments[j].should_be_del ← true;
    End
  End For
  j ← i + 1;
  While j < length Do
    If line_segments[j].should_be_del Then // 线被标记为需要删除
      line_segments[i] ← merge(line_segments[i], line_segments[j]); // 将两条相似的线合并成“最长”的一条线
      line_segments.remove(line_segments[j]); // 删除线条
      length ← line_segments.length;
    Else
      j ← j + 1;
    End
  End While
End For
```

2.3.2 优化算法

上文已提到原有算法逻辑忽略了线条 i + 1 可能与线条 i - 1 相似的情况，导致有多余的线条出现，进而影响了类关系识别。所以本文提出了以下解决方案，如算法 2.2 所示，在循环比较任意两条线条是否相似的过程中，若还有两条相似的线条合并为“最长”的一条线这个操作，则利用 flag 变量做标记，标识有合并线条操作，即合并过程中产出了新线条。在循环比较结束后，需要再次进行循环比较，以免遗漏相似线，确保最终所有线条都不相似，没有冗余线。

算法 2.2 关系线识别的部分优化逻辑

输入： 数组 line_segments[1..n], 数组长度 length
输出： 去除冗余线的数组 line_segments[1..m]

```
flag ← true;
While flag Do // 若还有相似线合并操作，则重新循环比较，避免遗漏相似线
  flag ← false;
  For i ← 0 to length Do
    j ← i + 1;
    While j < length Do
      If line_segments[i] == line_segments[j] Then // 两条线近似相等
        line_segments[i] ← merge(line_segments[i], line_segments[j]); // 将两条相似的线合并成“最长”的一条线
        line_segments.remove(line_segments[j]); // 删除线条
        length ← line_segments.length;
        flag ← true;
      Else
        j ← j + 1;
      End
    End While
  End For
End While
```

2.4 关系符号识别优化

算法旨在识别各种建模工具所绘制的类图，以包容不同的类图画法。然而，用户的画法多种多样，有些不符合 UML2.5 标准，同时算法对于特定类图的关系符号识别不够精确，影响了关系类型的判断。为此，本文决定转换思路，首先提高特定类图画法的准确率和召回率，重点识别由 StarUML 建模工具绘制的符合 UML2.5 标准的类图。下面将根据 UML2.5 标准[5]：实心方块加上实线为聚合关系；三角形加上实线为继承关系；三角形加上虚线为实现关系；箭头加上虚线为依赖关系。基于原有算法逻辑，重新梳理关系符号识别算法，以提高特定类图识别的准确率和召回率。

2.4.1 原有算法逻辑

在关系符号识别过程中，F Chen 等人采用形态学处理中的膨胀和腐蚀方法。先腐蚀后膨胀，这个操作称为开运算，它可以有效地处理图像中的噪声和结构，使得后续的图像分析和识别更加准确。然后识别图中所有轮廓，筛选出符合面积尺寸的轮廓，并采用二段逼近识别方法检测关系符号，二段逼近识别方法的原则如下：关系符号虽

然都带有一定缺陷、但都是近似标准形状的，因此可以轻微的“拉伸”一下关系符号，然后再精细化的“缩紧”它们，从而可以将缺陷抹除掉、同时保持其大致的形状不变，提高识别的准确度。

2.4.2 优化算法

执行关系符号识别步骤时，图像仅剩余三角形或实心方块等关系符号。在原有算法逻辑中采用了轮廓检测方法检测这些关系符号，而本文为了提高轮廓检测精度，决定在检测轮廓之前使用 Canny 边缘检测算法，它能够更好地突出图像中的边缘，后续的轮廓检测步骤可以更容易地检测到准确的轮廓，此外它进行了高斯滤波，平滑图像，减少了噪点的影响。

经过测试得出，原有算法逻辑目前存在两个问题：

(1)原有算法逻辑中的开运算方法，虽然可以有效地处理图像中的噪声和结构，但是由于阈值设置的不够恰当导致线条宽度变大，使三边形在后续识别过程中误识别成四边形，如图 2.1(a)所示。

(2)由于线条宽度过大导致一个三角形可能有外部三边形和内部三边形。原有算法逻辑提取图中所有轮廓的方法会将内部三边形的轮廓也提取出来，如图 2.1(b)所示，从而产生了冗余的多边形轮廓。

针对上述两个问题，本文提出了以下优化方法：

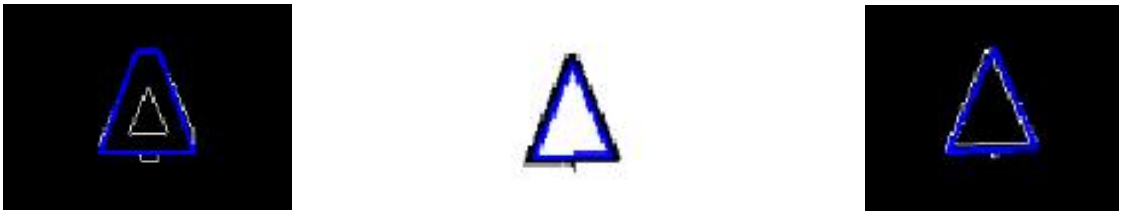
(1)由于之前的识别工作已经尽可能地减少噪点以及原有算法逻辑中过滤不符合面积尺寸的轮廓行为已经排除部分噪点，所以在关系符号识别中所存的噪点可忽略不计，因此本文决定删除开运算方法，避免线条宽度变大引起误识别的问题出现。

(2)修改检测轮廓方法的参数，使其从图中提取所有轮廓的方式转换至仅提取图中外部轮廓的方式。即 Imgproc.RETR_LIST 转换至 Imgproc.RETR_EXTERNAL。这个参数用于指定在查找轮廓时，只检索最外层的轮廓，而忽略嵌套在其中的内层轮廓。

最终效果如图 2.1(c)所示，优化算法如算法 2.3 所示，首先采用 Canny 边缘检测算法，以便提供连续且清晰的边界线，帮助更准确地识别形状。接着提取外部轮廓，忽略嵌套在其中的内层轮廓，再筛选出符合面积尺寸的轮廓。最后对符合要求的轮廓执行二段逼近识别方法以检测出所有关系符号。

本文结合了逼近多边形得到的顶点数量以及 UML2.5 标准来进一步确定关系的类

型。根据 UML2.5 标准可知，依赖关系的符号未闭合，不能使用二段逼近得到形状，所以默认关系为依赖关系，即没有获得精确形状的关系符号都认为是依赖关系。能够获得精确形状的关系符号根据顶点个数判断其关系类型：若顶点个数为 3 且为实线，则关系类型为继承关系；若顶点个数为 3 且为虚线，则关系类型为实线关系；若顶点个数为 4 且为实线，则关系类型为聚合关系。



(a) 识别成四边形　　(b) 识别出内部三边形　　(c) 正确识别出三边形

图 2.1 关系符号识别

算法 2.3 关系符号识别的部分优化逻辑

输入: cls_diagram, dash_lines, solid_lines

```
cls_diagram ← Imgproc.Canny(cls_diagram); // Canny 边缘检测算法，以便提供连续且清晰的边界线
contours ← Imgproc.findContours(cls_diagram, Imgproc.RETR_EXTERNAL); // 仅提取外部轮廓
For i ← 0 to contours.length Do
  contour_area ← Imgproc.contourArea(contours[i]);
  If contour_area < min_area Or contour_area > max_area Then // 过滤不符合面积尺寸的轮廓
    continue;
  End
  curve ← new MatOfPoint2f(contours[i]); // 曲线
  // 对“轮廓”做二次逼近
  curve ← Imgproc.approxPolyDP(curve, 0.05 * Imgproc.arcLength(curve, false)); // 第一次逼近，误差大一点，得到大致形状
  curve ← Imgproc.approxPolyDP(curve, 0.01 * Imgproc.arcLength(curve, true)); // 第二次逼近，误差小一点，得到精确形状
  If curve.toArray().length == 4 Then // 形状的顶点个数为 4
    For j ← 0 to solid_lines.length Do
      // 默认关系为依赖
      If solid_lines[j].type != “依赖” Then // 关系类型已经判断完成，跳过
        continue;
      End
      polygon ← new Polygon(curve); // 图符的多边形
      If polygon.contains(solid_lines[j]) Then // 延长线两端，若触碰到多边形，则执行以下步骤
        solid_lines[j].type = “聚合”; // 顶点个数为 4 且为实线
        break;
      End
```

```
    End For
  Else If curve.toArray().length == 3 Then // 形状的顶点个数为 3
    flag ← false;
    For j ← 0 to solid_lines.length Do
      If solid_lines[j].type != “依赖” Then
        continue;
      End
      polygon ← new Polygon(curve);
      If polygon.contains(solid_lines[j]) Then
        solid_lines[j].type = “继承”; // 顶点个数为 3 且为实线
        flag ← true;
        break;
      End
    End For
    If !flag Then
      For j ← 0 to dash_lines.length Do
        If dash_lines[j].type != “依赖” Then
          continue;
        End
        polygon ← new Polygon(curve);
        If polygon.contains(solid_lines[j]) Then
          solid_lines[j].type = “实现”; // 顶点个数为 3 且为虚线
          break;
        End
      End For
    End
  End
End For
```

3 需求分析与架构设计

3.1 需求分析

3.1.1 概述

为帮助开发人员节省手动建立图与代码间追踪关系的时间成本，本文旨在开发一个满足用户需求、符合用户习惯及易上手的追踪工具。

3.1.2 功能需求

本文主要实现一个半自动或自动的追踪工具以节省时间和人力成本，基于上述调研最终确定本工具的两大核心功能：采用 RSE-CDI 和 RSE-SDI 算法实现设计模型图语义元素识别功能以及采用 CDTC 和 SDTC 算法实现设计模型图与代码追踪构建功能。需要注意的是，设计模型图与代码追踪构建功能需要基于设计模型图语义元素识别功能生成的识别结果才能进行。为了保障两大核心功能的使用，本文决定增加项目管理功能，使用户使用起来更加方便。具体而言，本工具允许用户创建项目，在该项目中上传设计模型图进行设计模型图语义元素识别以便获取识别结果，进而进行设计模型图代码追踪构建工作获取追踪结果，从而满足用户需求，如图 3.1 所示。

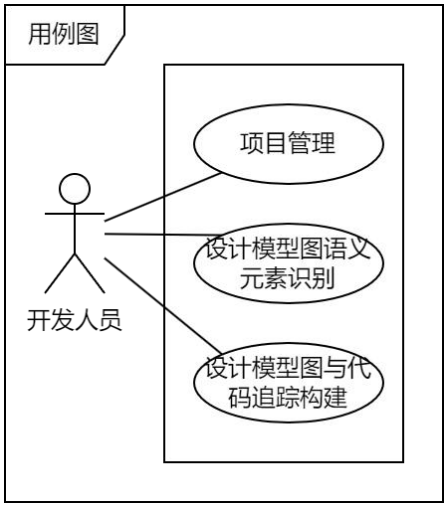


图 3.1 用例图

但是据调研结果可知，RSE-CDI 和 RSE-SDI 算法无法完全准确地识别图像，并获取完整的图信息，因此本文决定在设计模型图语义元素识别工作中增加编辑图片及保

存图片功能，以便用户能够手动编辑图片，将识别的图信息恢复至与原图信息一致，使后续的设计模型图与代码追踪构建工作可以更顺利地进行下去。

了解了功能的基本样貌后，本文根据功能所处理的特定操作进一步将功能划分为三个模块：项目模块、设计模型图语义元素识别模块及设计模型图与代码追踪构建模块，图 3.2 为每个模块的用例图，并设定用户角色为开发人员。

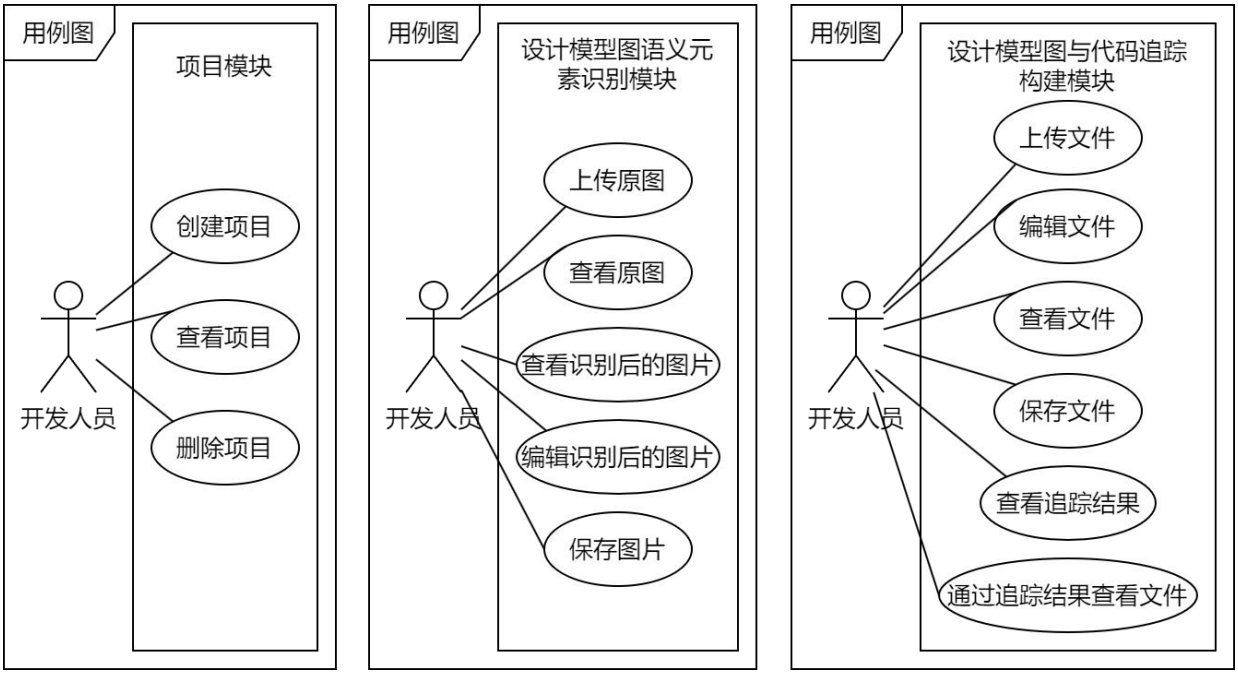


图 3.2 每个模块的用例图

项目模块的功能需求如图 3.2(左)所示。下文提到的项目是指类图语义元素识别、顺序图语义元素识别、类图与 Java 代码追踪构建或顺序图与 Java 代码追踪构建。下文提到的项目名称是指用户取的项目名称。下文提到的项目内容是指执行项目操作的页面。

(1)用户可以创建多个项目，每个项目表示一个项目名称与项目内容；

(2)用户可以查看项目，即点击项目名称，出现对应的项目内容；

(3)用户可以删除项目。

设计模型图语义元素识别模块的功能要求如图 3.2(中)所示。

(1)用户可以上传原图，这里的原图是指符合 UML2.5 标准且以 PNG/JPG 格式存储的图片；

(2)用户可以查看原图；

(3)用户可以查看识别后的图片，即上传的原图经过图语义元素识别后所展示的图

片；

(4)用户可以编辑识别后的图片，修改经过图语义元素识别后的图片信息；

(5)用户可以保存图片，即经过图语义元素识别后以 SVG 格式存储的图片。

设计模型图与代码追踪构建模块的功能要求如图 2.2(右)所示。下文提到的文件是指经过图语义元素识别并以 SVG 格式存储的图片、以 TXT/XLS/JSON 格式存储的文档、以 JAVA 格式存储的代码文件。

(1)用户可以上传文件；

(2)用户可以编辑文件；

(3)用户可以查看文件，即点击文档名称，出现对应的文档内容或点击代码文件名称，出现对应的代码内容；

(4)用户可以保存文件，包括以 SVG 格式存储的图片，以 ZIP 格式存储的所有设计文档，以 ZIP 格式存储的所有代码文件；

(5)用户可以查看追踪结果，即上传的文件经过图与代码追踪构建后得到追踪结果；

(6)用户可以通过追踪结果查看文件，即点击代码文件路径，出现对应的代码内容。

3.1.3 非功能需求

为了有良好的用户体验以及使工具具有稳定性，有如下非功能需求：

(1)界面设计：界面应该简单美观，增加用户喜爱度。

(2)易用性：工具应具备使用教程，让用户可以快速理解功能的使用方法。

(3)可用性：工具应具有高可用性，保证系统可以正常运行，不会出现异常情况。

(4)可扩展性：工具应具有良好的扩展性，以支持将来的功能扩展。

3.2 架构设计

3.2.1 概述

本章节在 3.1 需求分析的基础上进行了架构设计，并介绍了本工具的设计思路。如图 3.3 所示，架构设计图分为功能模块层、数据传输层及技术支撑层。首先，基于功能需求提到的三个模块以及非功能需求提到的易用性，本文确定了功能模块层的内容，其中包含了两大模块即基础模块和核心模块，基础模块分为教程模块和项目模块，核心模块分为设计模型图语义元素识别模块和设计模型图与代码追踪构建模块。核心

模块是本文实现核心功能即设计模型图语义元素识别功能以及设计模型图与代码追踪构建功能的重点内容，基础模块则是协助核心模块实现功能且为了方便用户使用功能而出现的内容。接着，由于本文更倾向于开发 Web 工具以及考虑到开发团队通常会将项目代码存储在托管平台上例如 Github 以便更高效地管理和协作项目代码，所以为了工具的扩展性，本文最终决定开发 Web 工具，而这个决定可能会造成本 Web 工具在传输数据给算法接口时，出现数据无法解析或格式不符合要求问题，因此为确保本工具与算法可以顺利地交换数据，不会报错，本文定义了数据传输格式规则，由此产生了数据传输层。最后，为了实现功能模块层，需要有良好的技术支撑，于是本文调研目前主流技术，选取了社区活跃的相关技术来实现功能模块，所采用的技术归为技术支撑层。

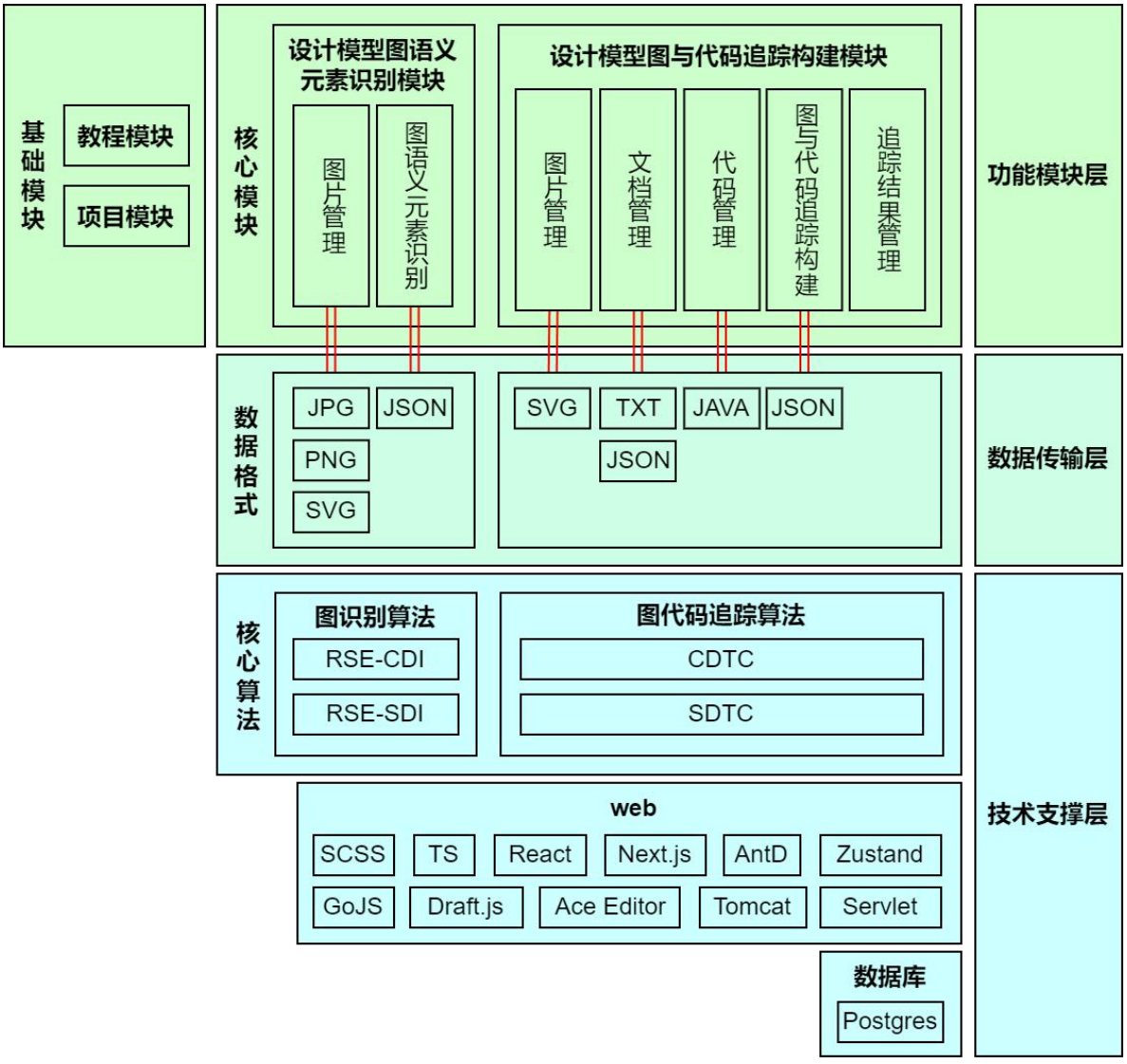


图 3.3 架构设计图

3.2.2 功能模块层

本节主要介绍功能模块的设计细节，以及为实现这些功能所选择的技术。

如图 3.4 所示，本工具主要分为基础模块和核心模块。基础模块是为了良好的用户体验所提供的基础功能。基础模块中的教程模块用来指导用户使用核心功能，让用户得以快速上手；基础模块中的项目模块用来管理项目，用户通过创建项目来进行核心功能。核心模块是为了满足用户需求所提供的核心功能。核心模块中的设计模型图语义元素识别模块用来识别类图语义元素及顺序图语义元素；设计模型图与代码追踪构建模块用来构建类图与 Java 代码追踪及顺序图与 Java 代码追踪。为了便于叙述，下文将对某些名称进行简化，具体简称见下图。

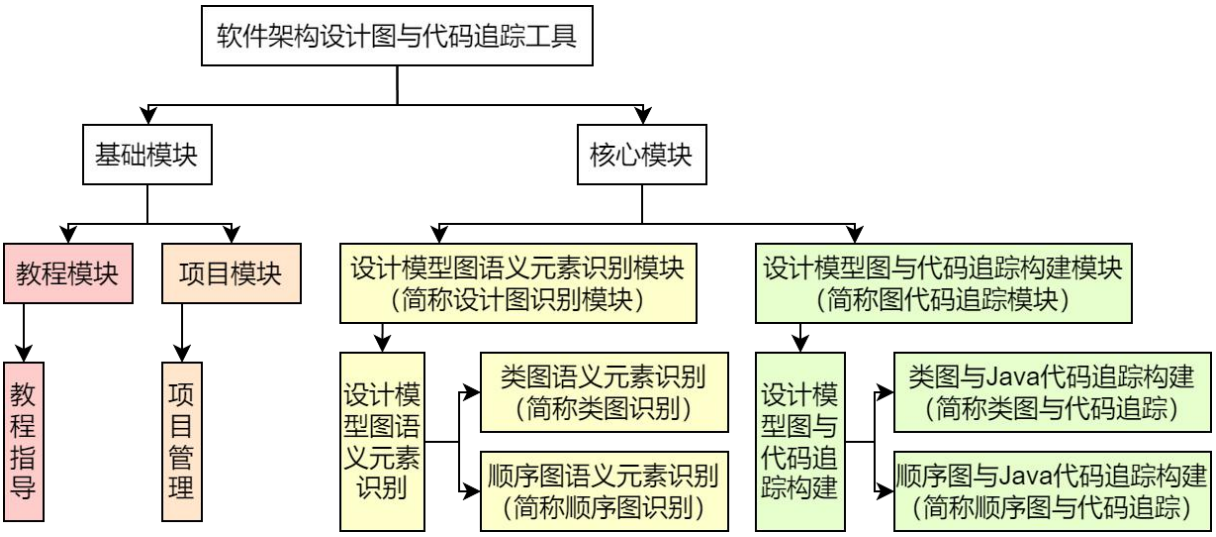


图 3.4 软件架构设计图与代码追踪工具的功能模块

对基础模块有了基本认识后，下文将进一步介绍教程模块和项目模块的设计细节。

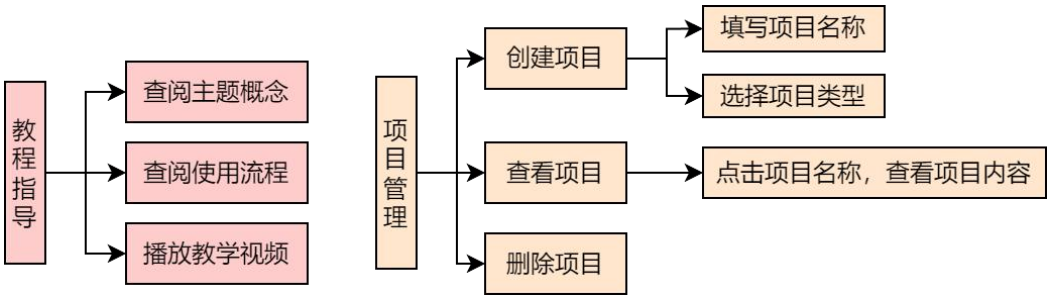


图 3.5 教程模块和项目模块的功能分解图

(1)教程模块：负责指导用户使用功能。为工具本身及功能提供简要描述，说明其概念及使用流程，并提供教学视频，旨在让用户快速上手。从工具本身、类图识别、类图与代码追踪、顺序图识别、顺序图与代码追踪等 5 个方面展开介绍。由于介绍内

容及教学视频属于静态数据，所以本文将它们存储在客户端中，不必再向服务端请求数据，避免了不必要的网络请求，从而节省网络资源。

(2)项目模块：负责管理项目信息。基于需求分析可知，用户可以创建多个项目。创建单个项目需要填写项目名称和选择项目类型。共有 4 个项目类型如类图识别、类图与代码追踪、顺序图识别、顺序图与代码追踪。同时，每个项目表示一个项目名称与项目内容。用户点击项目名称，可以查看对应的项目内容。为实现这个功能，本文选择了 AntD 组件库的 Tabs 组件，它很好地反映出项目名称与项目内容一一对应的关系。它不仅支持新增和关闭选项，还可以使用`closable={false}`禁止关闭页签。它提供的这些功能与项目模块的功能需求非常贴切。此外，每个标签页都有其独立的项目内容，项目内容包含项目数据如类图识别项目中的图片数据或类图与代码追踪项目中的图片、文档及代码数据。因此，在切换标签页时，需要考虑如何保存项目内容的数据状态，以确保切换回来时的数据状态仍然一样。为此本文选择了 Zustand 状态管理库存储每个标签页的数据状态，从而解决了这个问题。具体解决方法是项目名称作为 key，项目内容的数据作为 value，以 Map 结构的形式存储在 JavaScript 运行时内存空间。

对核心模块有了基本认识后，下文将进一步介绍设计图识别模块和图代码追踪模块的设计细节。

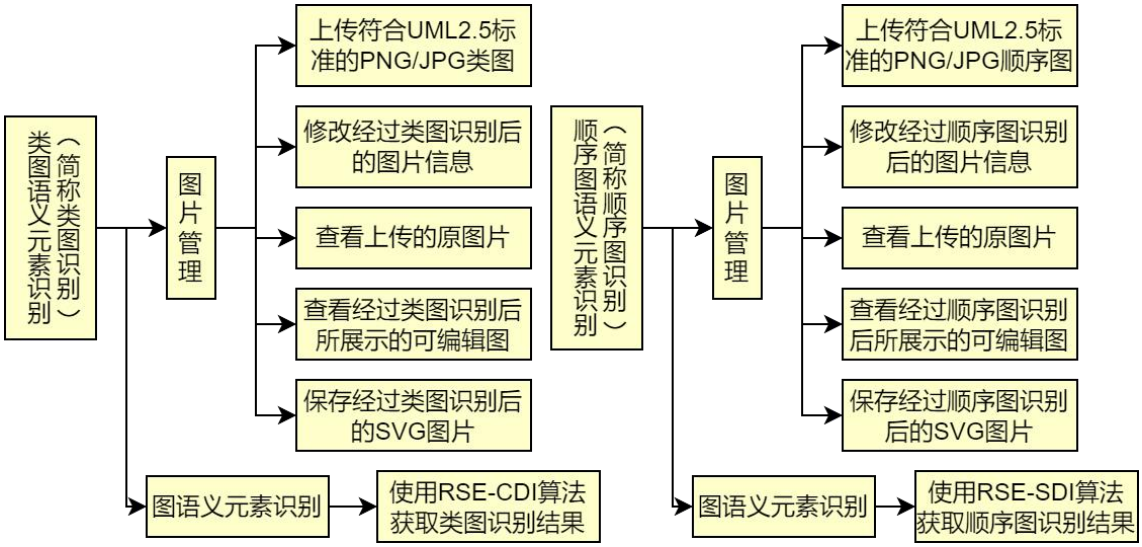


图 3.6 设计图识别模块的功能分解图

(1)设计图识别模块：负责识别类图语义元素及顺序图语义元素。由于类图语义元素识别与顺序图语义元素识别的功能类似，唯一的区别在于图片信息与图识别算法不同，为方便叙述，下文将类图和顺序图统称为图。用户可以在已创建的图识别项目标

签页上上传符合要求的图，浏览器将图发送给服务器，服务端调用图识别算法生成识别结果返回给浏览器，客户端解析识别结果后展示图编辑区域，用户可以查看原图，通过修改图信息将识别后的图信息恢复至与原图信息一致，编辑结束后可以将图保存至本地电脑。为实现具有编辑 UML2.5 标准的图功能，本文选择了 GoJS 图表库来定制化编辑图功能。

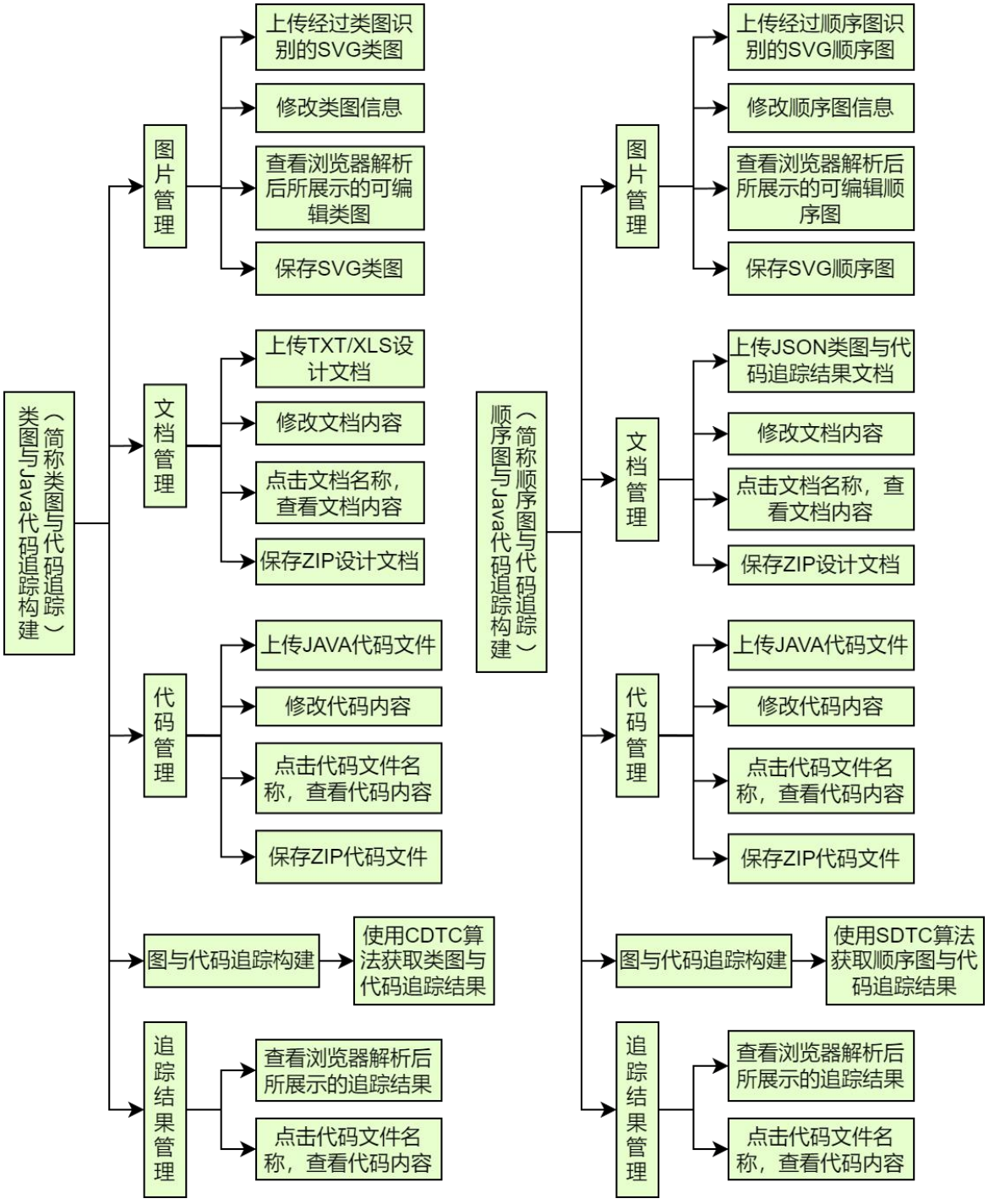


图 3.7 图代码追踪模块的功能分解图

(2)图代码追踪模块：负责构建类图与 Java 代码追踪及顺序图与 Java 代码追踪。由于构建类图与 Java 代码追踪与构建顺序图与 Java 代码追踪的功能类似，唯一的区别在于图片信息、上传的文档类型以及图代码追踪算法不同，为方便叙述，下文将类图和顺序图统称为图。用户可以在已创建的图与代码追踪项目标签页上上传符合要求的图、文档及代码，客户端解析这些文件后展示图编辑区域、文档编辑区域、代码编辑区域。用户可自主决定在什么时候获取追踪结果。用户点击“获取追踪结果”按钮后，浏览器将这些文件数据发送给服务器，服务端调用图代码追踪算法生成追踪结果返回给浏览器，客户端解析追踪结果后展示追踪结果区域，用户可点击追踪到的代码文件名称，查看对应的代码内容。用户可保存修改后的文件信息，为下次上传同样的文件节省不必要的重复修改。用户可保存追踪结果至本地电脑，方便反复查看。为实现文档编辑功能，本文选择了 Draft.js 文档编辑库；为实现代码编辑功能，本文选择了 Ace Editor 代码编辑库。这两个库的功能齐全，满足了本工具的所有需求，而且功能种类繁多，扩展性极强。

3.2.3 数据传输层

基于上述核心模块的介绍可以发现，算法仅提供了接口，不能与浏览器交换数据，为了解决浏览器与算法接口交换数据问题，本文选择了 Tomcat 服务器以及 Servlet API 技术，在算法基础上搭建了 Tomcat 服务器，使得浏览器可以将数据传输给 Tomcat 服务器，服务端再调用算法接口，并将输出数据返回给浏览器。

算法接口能正常访问数据后，还需考虑数据传输的格式，以免在数据传输过程中因数据格式不符而报错，而且混乱的数据格式会影响本 Web 工具的可用性以及稳定性。因此，本文严格定义了数据传输格式规则，首先本文介绍了核心功能与算法交换数据的格式要求，接着再从技术层面的角度剖析原因。

图识别功能要求用户上传 JPG 或 PNG 格式的图片，通过图识别算法返回的数据要求是 JSON 格式，以便作为 GoJS 图表库的图编辑组件的输入，从而展示图编辑区域。用户可以保存以 SVG 格式存储的图片至本地电脑。图代码追踪功能要求用户上传 SVG 格式的图片，TXT 格式或 JSON 格式的文档及 JAVA 格式的代码文件，通过图代码追踪算法返回的数据要求是 JSON 格式。

显而易见的数据格式要求本节不做解释，仅对重要的数据格式要求进行讲解。在

设计图识别模块的介绍中提到了 GoJS 图表库的应用以及“客户端解析识别结果后展示编辑图区域”的描述，这意味着客户端负责将识别结果处理成 GoJS 图表库的图编辑组件能够接受的输入格式即 JSON 格式。在图 3.7 中要求用户“上传经过图识别的图片”，这意味着图代码追踪功能要求用户上传的图片是上一步图识别功能所保存的图片，而且这个图片必须携带识别结果信息，而以 SVG 格式存储的图片意指这个图片既可以携带描述二维矢量图片的信息又可以携带 JSON 信息，也就是说，它既方便用户打开查看图片，又可以保存图识别结果信息。在图 3.7 的类图与代码追踪中提到了“上传 XLS 的设计文档”，即表示了用户可能上传一些意想不到的文件格式，本文统一将除 TXT 文本格式外的格式转换成 JSON 格式，之所以选择转换成 JSON 格式是因为它具有格式简洁、广泛支持及高效解析等优点，是 Web 开发中数据交换的首选格式。

3.2.4 技术支撑层

为支撑功能模块的实现，本文首先选择了 React 框架，并以它为主，调研客户端技术。为支撑核心功能的实现，本文选择了 RSE-CDI、RSE-SDI、CDTC、SDTC 算法，并以它们为主，调研服务端技术，主要从技术之间的兼容性、技术的易用性、技术的扩展性、技术在社区的活跃度四个层面来选择客户端技术、服务端技术及数据库技术。最终选择的技术如图 3.8 所示。

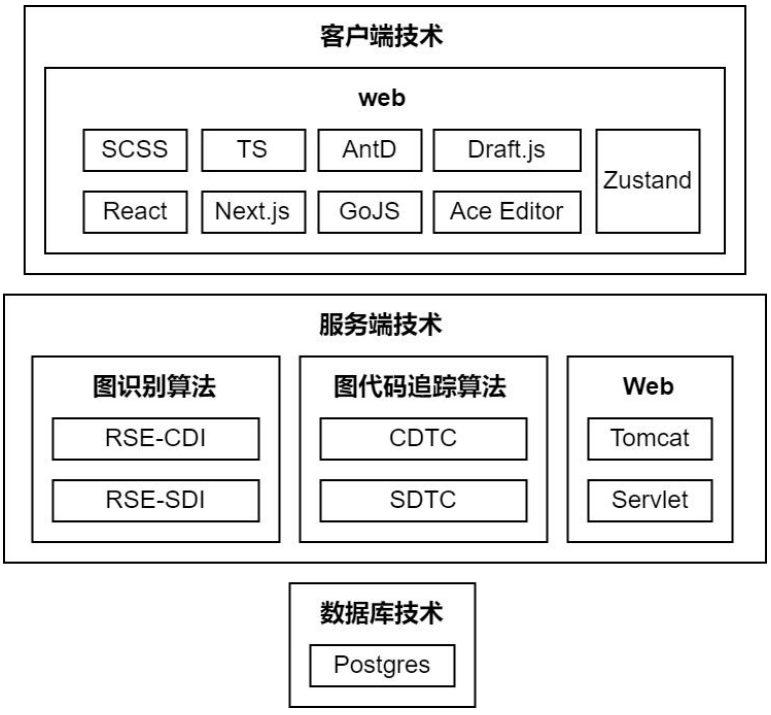


图 3.8 选择的所有技术

客户端技术主要为 Sassy CSS（SCSS）、TypeScript（TS）、React 框架、基于 React 的 Zustand 状态管理库、基于 React 的 AntD 组件库[13]、基于 React 的 GoJS 图表库[14]、基于 React 的 Draft.js 文本编辑库[20]、基于 React 的 Ace Editor 代码编辑库[21]及用于构建 React 应用的 Next.js 框架[15]。它们提供简单且灵活的 API，有丰富的社区及详细的文档。

服务端技术主要为 Gitee 平台上发布的开源代码如 RSE-CDI、RSE-SDI、CDTC 及 SDTC 算法，还有为搭建 Web 服务器所采用的 Tomcat 及 Servlet 应用程序。它们互相兼容，能够有效地将识别结果和追踪结果返回给客户端。

数据库技术主要使用 PostgreSQL 数据库。PostgreSQL 高度兼容 SQL 标准，对于熟悉 SQL 的人员来说，学习时间短，可以快速上手。它支持用户自定义数据类型、函数等，可扩展性强，对于后续迭代开发很有帮助。它可以将查询的数据封装成 JSON 格式，有利于 GoJS 库使用。它还可以和 Next.js 结合使用，Next.js 的 API Routes 可以直接与 PostgreSQL 交互。

总的来说，本文通过调研并选择了合适的前端、后端和数据库技术，以支持功能模块的实现。前端技术以 React 框架为主，结合了 SCSS、TS 及多种 React 生态的库，如 Zustand、AntD、GoJS、Draft.js 和 Ace Editor 等，形成了灵活且高效的客户端架构。后端方面，选用了 RSE-CDI、RSE-SDI、CDTC 和 SDTC 等开源算法，基于 Tomcat 和 Servlet 构建 Web 服务器，实现了对识别和追踪结果的有效处理与响应。数据库使用了 PostgreSQL，其能够与前端技术良好配合。

4 实现与测试

4.1 功能实现

4.1.1 概述

3.2 架构设计章节已经描述了设计的具体细节，接下来通过实操将设计内容完整地实现出来，所产出的产物为软件架构设计图与代码追踪工具。下文将借助功能流程图及 UI 界面对功能实现进行介绍。此外，本文将项目代码存储在了 Github 平台，感兴趣的读者可以自行了解代码实现[22]。

4.1.2 基础功能-UI 设计

(1) 教程模块

如图 4.1 所示，上方为标签头列表，第一个标签为主页表示教程界面，默认打开且不可删除。“UML&Code Trace”是本工具的名称，左上角的图标为本工具的 LOGO。左边的列表从两个维度对五个方面：工具本身（UML&Code Trace）、类图识别功能（Class Diagram Recognize，简称 CR）、顺序图识别功能（Sequence Diagram Recognize，简称 SR）、类图代码追踪功能（Class Diagram Trace，简称 CT）、顺序图代码追踪功能（Sequence Diagram Trace，简称 ST）进行介绍，详情见下文。

“UML&Code Trace”的“介绍”描述了工具的背景以及目前可支持的 UML 图与代码。“核心功能”描述了功能的流程图。

“CR”的“介绍”描述了类图的标准以及具体的使用流程。“操作指南”描述了类图编辑的使用快捷键并提供了操作视频。

“CT”的“介绍”描述了上传文件的格式要求以及具体的使用流程。“操作指南”描述了类图编辑的使用快捷键并提供了操作视频。

“SR”的“介绍”描述了顺序图的标准以及具体的使用流程。“操作指南”描述了顺序图编辑的使用快捷键并提供了操作视频。

“ST”的“介绍”描述了上传文件的格式要求以及具体的使用流程。“操作指南”描述了顺序图编辑的使用快捷键并提供了操作视频。

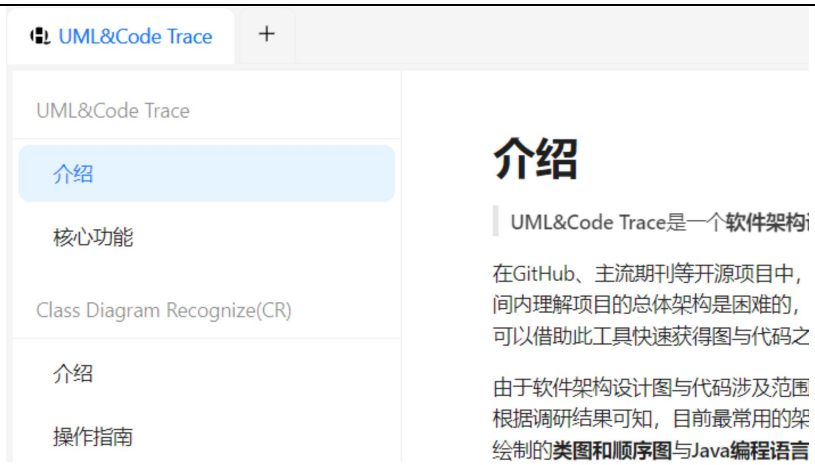


图 4.1 教程界面

(2) 项目模块

如图 4.2 所示，上方标签头列表除第一个主页外，皆为项目，即用户可以创建多个项目，并从中选择要查看的项目。点击项目名称右边的“×”符号可以删除项目。点击“+”符号新增项目，可以创建项目，填写项目名称，选择项目类型。用户可以按照自己的需求，自主选择从图识别步骤或从图与代码追踪步骤开始执行。

下面对上方标签头列表的处理过程进行介绍：上方标签头列表个数会随着用户创建项目而增加，这时在切换标签头时，标签页的数据会跟着改变。为了确保切换标签头时，其标签页显示的数据状态是正确的，本文借助了 Zustand 状态管理库将数据以 Map 结构（key:标签头名称, value:标签页内容）的形式存储在 JS 运行时内存空间中。当切换标签头时，从内存中提取标签页的数据以便传递给各个组件，最终在标签页中显示正确的数据。

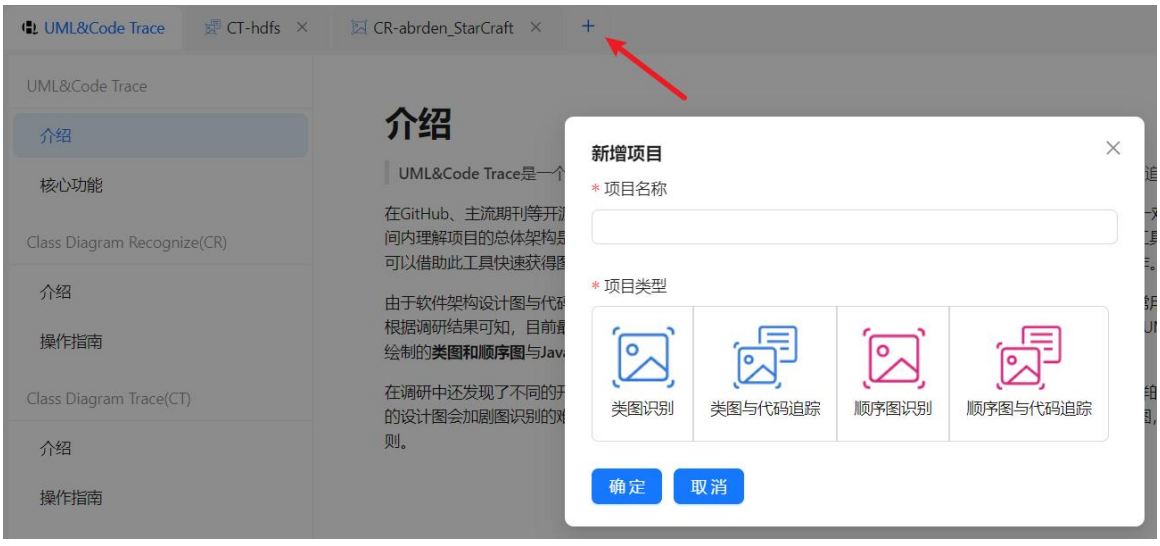


图 4.2 新增项目功能界面展示

4.1.3 核心功能-UI 设计

(1) 概述

该工具的核心功能可分为四个部分（红色字体），两大模块（蓝色字体）如图 4.3 所示。下面将以类图举例描述两大模块的具体功能。

设计图识别模块：用户上传符合 UML2.5 标准的类图，通过 RSE-CDI 识别算法获取图信息并生成新的可编辑图供用户编辑以恢复缺失的图信息，确保与原图一致后，保存至本地，为下一步使用。综上所述，输入为 JPG/PNG 格式图片，输出为编辑后正确的 SVG 格式图片。

图代码追踪模块：用户上传编辑后的 SVG 图片，TXT/XLS 设计文档（可选），JAVA 代码，通过 CDTC 追踪算法获取所有类名与代码间的追踪关系。追踪结果可能是多对多关系，不易用户查看，本文将追踪结果转换为一对多关系，即一个类名对应多个追踪文件路径。用户点击类名后，可获取追踪到的所有文件路径，点击文件路径可打开相应文件，方便对比，分析追踪结果。综上所述，输入为 SVG 格式的图片、TXT 或 JSON 格式的文档及 JAVA 格式的代码，输出为 JSON 格式的追踪结果。

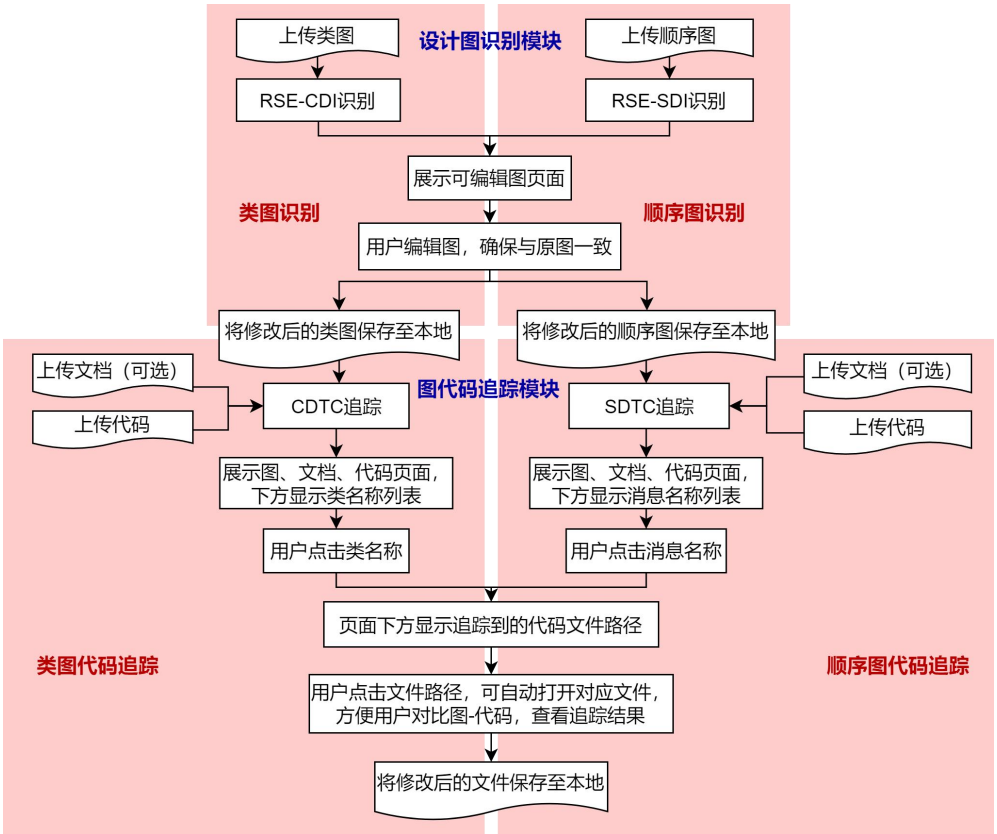


图 4.3 自定义总流程图

(2) 设计图识别模块

如图 4.4 所示，详细介绍了图识别功能的流程图。以类图为例，用户上传 UML 类图，通过 RSE-CDI 图识别算法获取图识别结果并返回给浏览器，浏览器会展示图 4.5 的页面：左侧展示可编辑图，右侧展示原图。方便用户进行对比，修正错误内容；用户可通过上方的工具栏拖动元素，添加类矩形或类关系线条；通过双击修改类名、类属性或类方法的文字；通过右击增加类属性、类方法或方法参数；通过右击删除指定类属性、类方法或方法参数。通过右击转换类关系的目标方向；修改完成后，可点击保存按钮将图片以 SVG 格式保存至本地电脑，为下一步图代码追踪功能使用。同理，顺序图也具备这些功能。

下面对左侧编辑图区域及右侧原图区域的处理过程进行介绍：(1)浏览器将用户上传的类图发送至服务器，服务器调用识别算法并返回结果。浏览器接收识别结果后，借助 Zustand 状态管理库将结果以字符串形式存储在 JS 运行时内存空间中，当需要显示该图片时，从内存中提取数据并将其传递给 GoJS 库的编辑图组件，最终在编辑图区域中渲染该组件。(2)浏览器将用户上传的类图转换成 base64 格式并存储在 JS 运行时内存空间中，当需要显示该图片时，从内存中提取出 base64 数据，设置为 img 标签的 src 值，浏览器解析并渲染 base64 格式的图片，最终在指定区域中显示该图片。

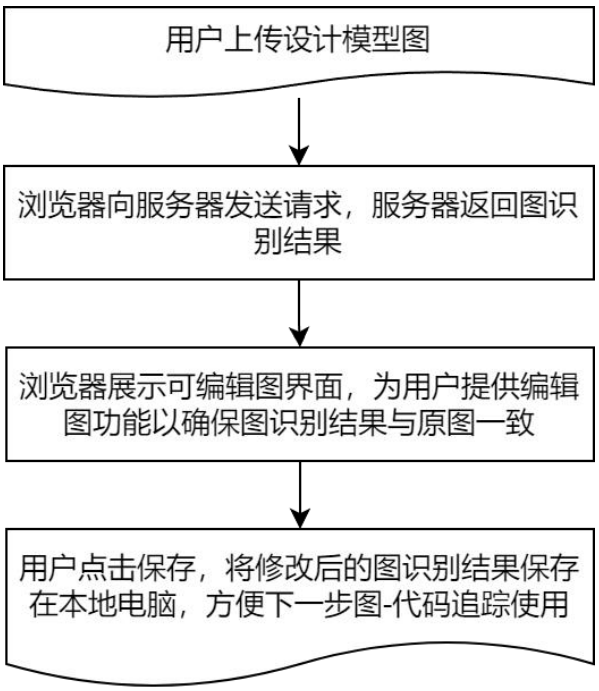


图 4.4 图识别功能流程图

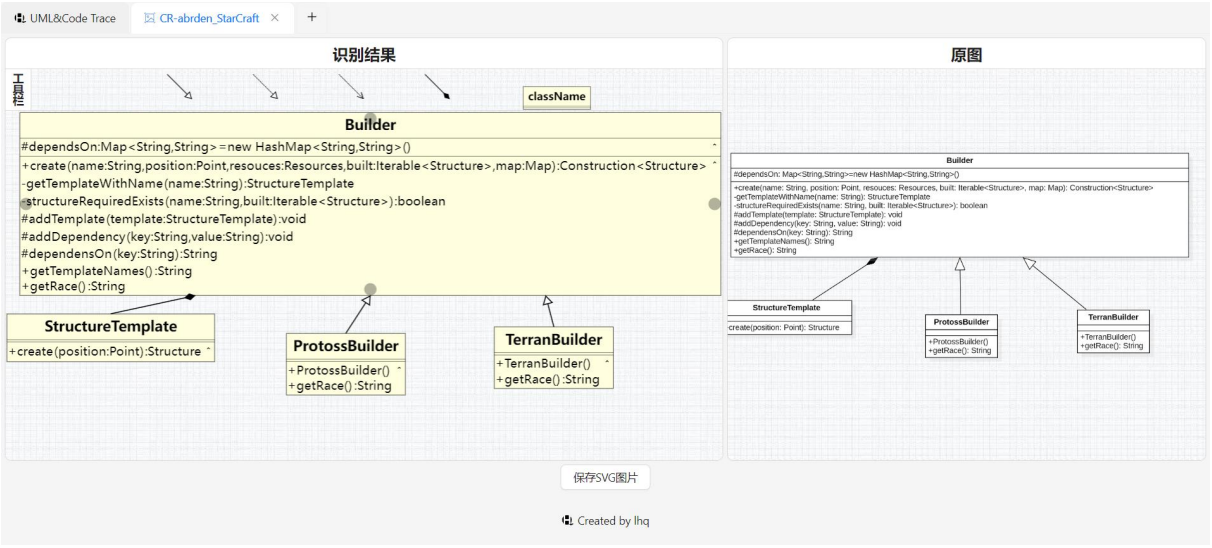


图 4.5 图识别功能界面

表 4.1 描述了用于获取类图识别结果和获取顺序图识别结果的接口定义，具体包含了上传图片后向服务器发送请求的结构体。接口定义详细说明了请求的格式和字段要求，以确保客户端和服务器之间的正确通信。

表 4.1 图识别功能的接口定义

操作	请求行		请求体		
	请求方法	URL	字段	类型	说明
获取类图识别结果	POST	/recog/class-diagram	picture	File	图片二进制文件，必填
获取顺序图识别结果	POST	/recog/sequence-diagram			

(3) 图代码追踪模块

如图 4.6 所示，详细介绍了图代码追踪功能的流程图。以类图为例，用户上传上一步包含图识别信息的 SVG 图片，设计文档（可选）与代码。浏览器会将其解析为三个区域分别是可编辑图、文件名称列表及对应的文档文件、文件名称树及对应的代码文件。点击“获取追踪结果”按钮后，通过 CDTC 算法获取追踪结果并返回给浏览器。浏览器会展示图 4.7 的页面，下方展示追踪结果：左侧为类名列表，右侧为类名追踪到的所有代码文件路径；可通过点击文件路径打开相应代码文件；编辑图片、文档或代码后，可点击保存按钮保存至本地电脑。同理，顺序图也具备这些功能。

下面对左侧编辑图区域、中间编辑文档区域及右侧编辑代码区域的处理过程进行

介绍：(1)浏览器负责从用户上传的类图中提取图信息并借助 Zustand 状态管理库将图信息以字符串形式存储在 JS 运行时内存空间中，当需要显示该图片时，从内存中提取数据并将其传递给 GoJS 库的编辑图组件，最终在编辑图区域中渲染该组件。(2)中间编辑文档区域分为上下两个部分，上方为文档名称列表，下方为文档文件内容。客户端将用户上传的文件列表转换成 Map 结构（key:文件名称, value:文件内容）并借助 Zustand 状态管理库将 Map 结构数据存储在 JS 运行时内存空间中。列表结构可直接传递给 AntD 库的分段控制器组件以呈现上方的文档名称列表。转换成 Map 结构是为了当需要显示该文档内容时，方便从内存中提取数据并将其传递给 Draft.js 库的编辑文档组件，最终在下方渲染该组件。(3)右侧编辑代码区域分为左右两个部分，左方为代码名称树，右方为代码文件内容。客户端将用户上传的文件列表转换成两种结构：树结构（文件名称树）和 Map 结构（key:文件名称, value:文件内容），并借助 Zustand 状态管理库将 Map 结构数据存储在 JS 运行时内存空间中。转换成树结构是为了方便传递给 AntD 库的树形控件组件以呈现左方的代码名称树。转换成 Map 结构是为了当需要显示该代码内容时，方便从内存中提取数据并将其传递给 Ace Editor 库的编辑代码组件，最终在右方渲染该组件。

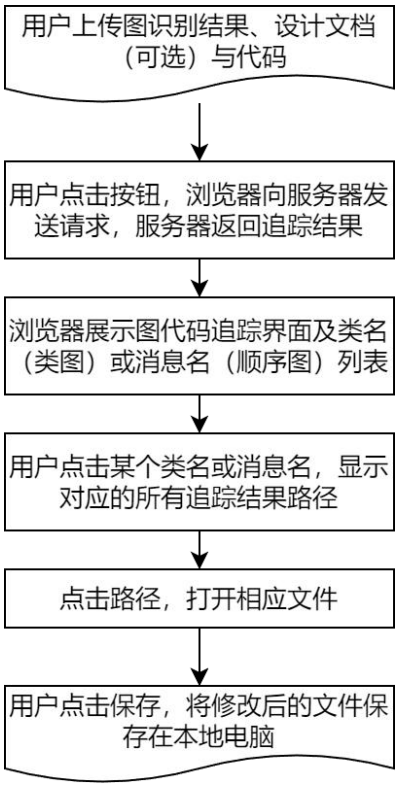


图 4.6 图代码追踪功能流程图

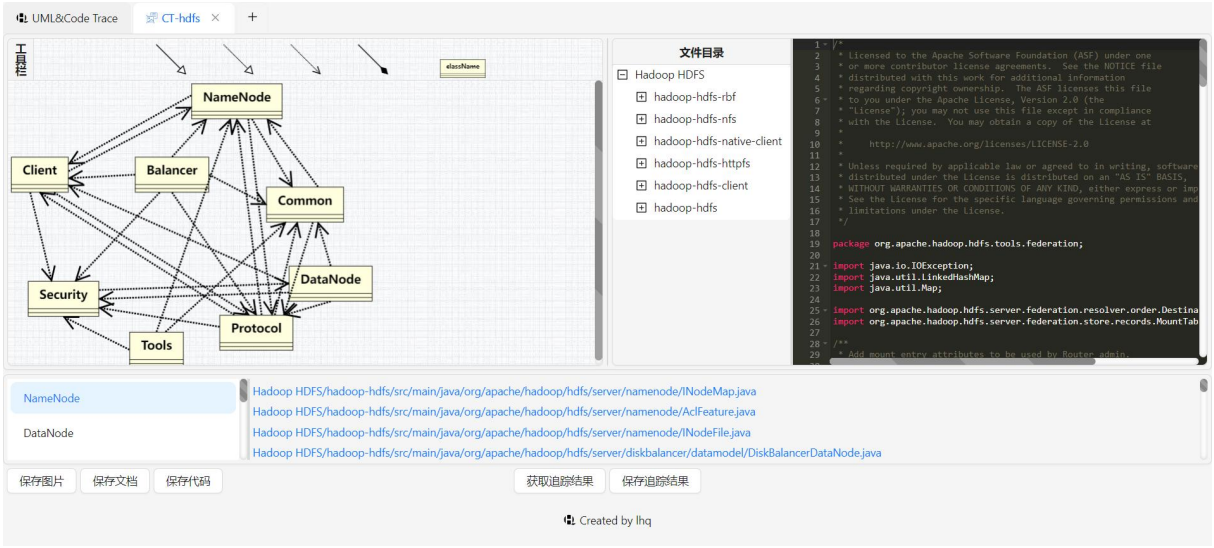


图 4.7 图代码追踪功能界面

表 4.2 描述了用于获取上传 ID、获取类图代码追踪结果及获取顺序图代码追踪结果的接口定义，具体包含了上传文件后向服务器发送请求的结构体。接口定义详细说明了请求的格式和字段要求，以确保客户端和服务器之间的正确通信。

获取上传 ID 操作只为了从服务器获取唯一标识，所以选用 GET 方法更为合适。获取类图代码追踪结果及获取顺序图代码追踪结果操作需要传输大量文件数据，而 GET 方法的 URL 具有长度限制且参数暴露在 URL 中，不适合传输敏感的大数据，所以为了数据的安全以及能够正常传输，更适合采用 POST 方法。

由于上传的文件数量较大，本文采用了分批处理，所以在请求体的字段中包含了 batchNumber 和 totalBatches 来携带分批处理信息。同时为了辨别批次来源于哪个请求，添加了 uploadId 字段作为标识，所以表 4.2 中包含了获取上传 ID 的接口定义。

表 4.2 图代码追踪功能的接口定义

操作	请求行		请求体		
	请求方法	URL	字段	类型	说明
获取上传 ID	GET	/generate-upload-id	—	—	—
获取类图代码追踪结果	POST	/trace/class-diagram	uploadId	UUID	上传 ID
			batchNumber	Number	批次号码
			totalBatches	Number	总批次数量
获取顺序图代	POST	/trace/sequence-diagram	diagFile	File	图片二进制文件，必填

码追踪结果	docFiles	File	文档二进制文件，必填
	codeFiles	File	代码二进制文件，必填

4.1.4 辅助功能

如图 4.7 所示，页面展示的内容包括图编辑区域、文档编辑区域、代码编辑区域及追踪结果区域，由于展示的内容较多且需符合窗口大小，展示区域不得不缩小，这会导致用户看的不清晰，所以为了提升用户体验，开发了折叠面板组件和拖拽组件。

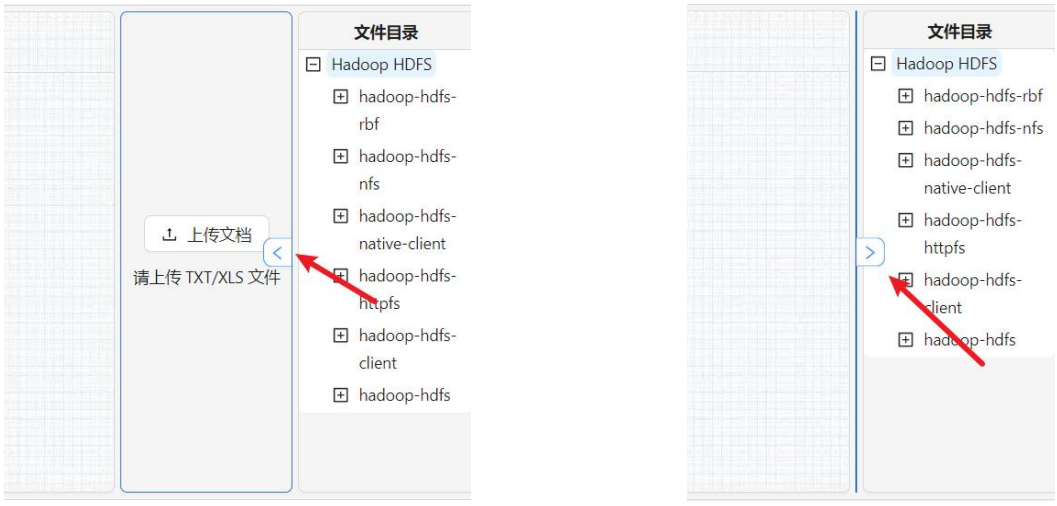
(1) 折叠面板组件

折叠面板分为纵向和横向两种。首先以纵向折叠面板中的类图区域举例，图 4.8(a)为展开状态，图 4.8(b)为收起状态。当处于展开状态时，鼠标靠近“收起”箭头，会触发蓝色边框样式，可通过点击箭头将类图区域收起，扩大代码的展示区域，方便用户编辑和查看。此功能已经封装成组件库，并提供了相应的接口（表 4.3），该接口允许用户输入需要展示的组件和它的默认宽度，确保组件首次加载时能够按照指定的宽度进行展示。

表 4.3 纵向折叠面板及拖拽组件的接口定义

字段	类型	说明
children	React.JSX.Element[]	React 元素，需要展示的组件
width	string[]	默认宽度

注: children 和 width 的个数必须一致.



(a) 展开状态 (b) 收起状态

图 4.8 纵向折叠面板组件界面展示

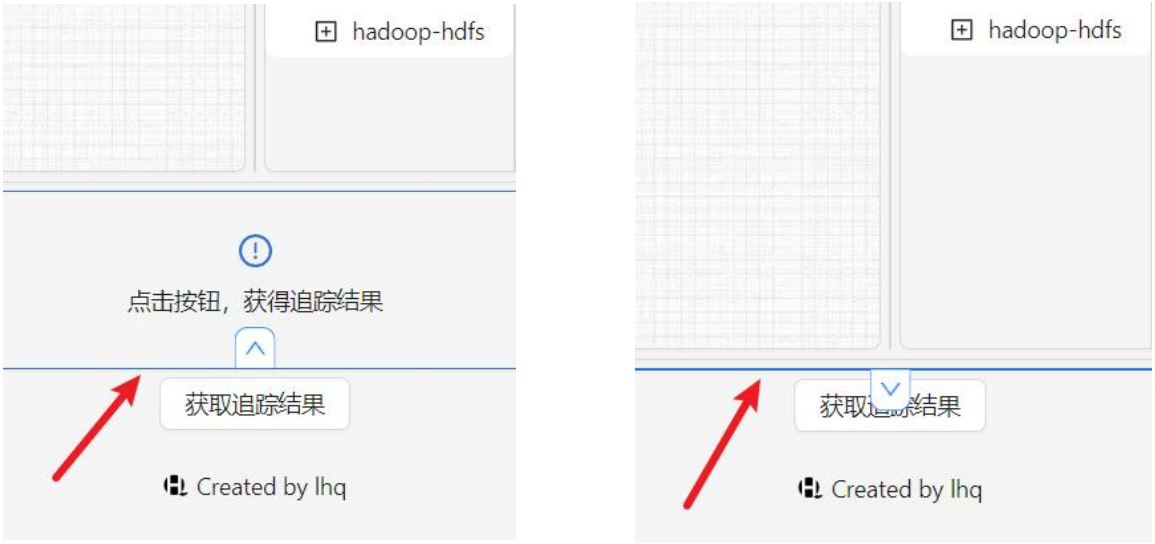
接着以横向折叠面板中的追踪结果区域举例，图 4.9(a)为展开状态，图 4.9(b)为收起状态。当处于展开状态时，鼠标靠近“收起”箭头，会触发蓝色边框样式，可通过点击箭头将追踪结果区域收起。当处于收起状态时，鼠标靠近“展开”箭头，会触发蓝色边框样式，可通过点击箭头将追踪结果区域展开，恢复至原来的高度。此功能已经封装成组件库，并提供了相应的接口（表 4.4），该接口允许用户输入需要展示的组件和它的默认高度，确保组件首次加载时能够按照指定的高度进行展示。

在使用折叠面板组件时，建议在其外部包裹一个父组件，并为父组件设置宽度和高度。这可以确保折叠面板组件在特定的尺寸范围内显示，从而避免布局问题并提高界面的美观性和一致性。

表 4.4 横向折叠面板的接口定义

字段	类型	说明
children	React.JSX.Element[]	React 元素，需要展示的组件
height	string[]	默认高度

注: children 和 height 的个数必须一致.



(a) 展开状态 (b) 收起状态

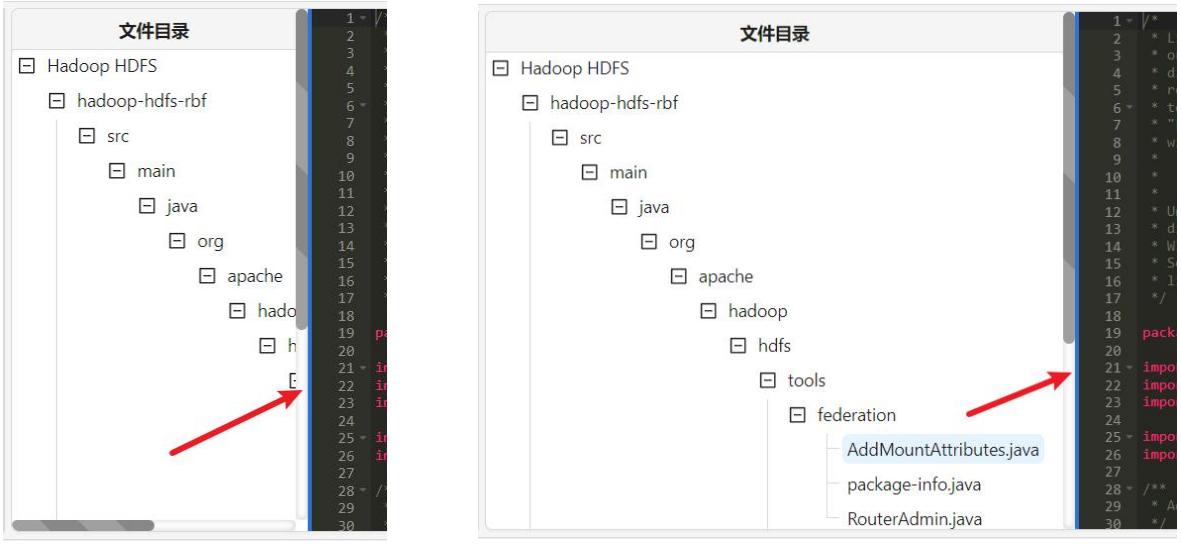
图 4.9 横向折叠面板组件界面展示

(2) 拖拽组件

图 4.10(a)为拖拽前状态，图 4.10(b)为拖拽后状态。鼠标靠近条框时，会触发蓝色条框以及拖拽鼠标样式，按住不放并拖动条框可以调整文件目录与代码间的展示区域，从而方便用户查看文件路径。此功能已经封装成组件库，并提供了相应的接口（表 4.3），

该接口允许用户输入需要展示的组件和它的默认宽度，确保组件首次加载时能够按照指定的宽度进行展示。

在使用拖拽组件时，建议在其外部包裹一个父组件，并为父组件设置宽度和高度。这可以确保拖拽组件在特定的尺寸范围内显示，从而避免布局问题并提高界面的美观性和一致性。



(a) 拖拽前状态　　(b) 拖拽后状态

图 4.10 拖拽组件界面展示

4.2 功能测试

4.2.1 测试设计

(1)测试目标

验证功能的完整性及可用性，预期软件架构设计图与代码追踪工具的所有功能都能正确运行，出现异常情况时有正确的提示。

(2)测试范围

功能模块的测试范围如下：教程指导功能、项目管理功能、类图识别功能、顺序图识别功能、类图与 Java 代码间的追踪功能、顺序图与 Java 代码间的追踪功能、界面交互功能。

(3)测试类型

单元测试（测试模块的原子函数）；集成测试（测试模块之间的交互性）。

(4)测试场景设计

基于上述确定的测试目标、测试范围及测试类型，考虑功能的正常及异常情况并进一步设计测试场景。需要注意的是，本文认为的正常情况是允许发生的情况，异常情况是不允许发生的情况。在异常情况下，工具需要有相应的提示，让用户知道这个操作是不被允许的。根据测试范围及测试类型设计了下文的测试场景，这些测试场景只是一个概括，具体细节见测试结果的操作描述。

表 4.5 教程指导功能的单元测试场景

序号	情况	测试场景
1	正常	查看教程文档，验证教程正常展示
2	正常	查看教程视频，验证视频正常播放

表 4.6 项目管理功能的单元测试场景

序号	情况	测试场景
1	正常	创建新项目，验证项目成功创建
2	正常	创建重名项目，验证项目成功创建
3	异常	创建空项目，验证工具提示错误
4	正常	删除项目，验证项目成功删除

对于类图识别功能的测试，要求类图遵守以下条件：

图片格式：JPG/PNG；UML2.5 标准，如图 4.11 所示。

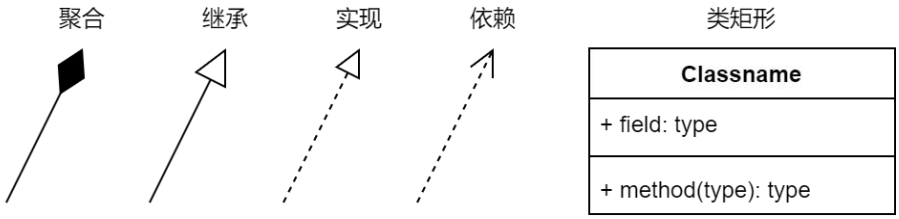


图 4.11 UML2.5 类图标准

对于顺序图识别功能的测试，要求顺序图遵守以下条件：

图片格式：JPG/PNG；UML2.5 标准，如图 4.12 所示。

图 4.12 UML2.5 顺序图标准

由于类图识别功能与顺序图识别功能的测试场景几乎一致，所以本文将它们放在一起介绍。

表 4.7 类图识别功能及顺序图识别功能的单元测试场景

序号	情况	测试场景
1	正常	上传符合格式的图片，验证图片成功上传
2	异常	上传不符合格式的图片，验证工具提示错误
3	正常	上传符合 UML2.5 标准的图片，验证图片正常识别
4	异常	上传不符合 UML2.5 标准的图片，验证工具提示错误
5	正常	编辑图片，验证所有编辑功能正常使用
6	正常	保存识别结果，验证结果保存成功

对于类图与 Java 代码间的追踪功能的测试，要求文件遵守以下条件：

图片格式：SVG；文档格式：TXT/XLS；代码格式：JAVA。

对于顺序图与 Java 代码间的追踪功能的测试，要求文件遵守以下条件：

图片格式：SVG；文档格式：JSON；代码格式：JAVA。

由于类图与 Java 代码间的追踪功能与顺序图与 Java 代码间的追踪功能的测试场景几乎一致，所以本文将它们放在一起介绍。

表 4.8 类图与 Java 代码间的追踪功能及顺序图与 Java 代码间的追踪功能的单元测试场景

序号	情况	测试场景
1	正常	上传符合格式的图片，验证图片成功上传
2	异常	上传不符合格式的图片，验证工具提示错误
3	正常	上传符合格式的文档，验证文档成功上传
4	正常	上传不符合格式的文档，验证工具正常运行
5	正常	上传符合格式的代码，验证代码成功上传
6	正常	上传不符合格式的代码，验证工具正常运行
7	正常	点击文档文件名称，查看文档内容
8	正常	点击代码文件名称，查看代码内容

9	正常	编辑图片或文档或代码，验证所有编辑功能正常使用
10	正常	保存图片或文档或代码或追踪结果，验证所有文件保存成功

表 4.9 界面交互功能的单元测试场景

序号	情况	测试场景
1	正常	点击纵向折叠面板，验证展开和收起功能正常使用
2	正常	点击横向折叠面板，验证展开和收起功能正常使用
3	正常	拖动条框，验证拖拽功能正常使用

本文在进行单元测试的同时也进行了集成测试，如表 4.10 为集成测试的测试场景。这些测试场景只是一个概括，具体细节见测试结果的前提条件及操作描述。

表 4.10 集成测试的测试场景

序号	情况	测试场景
1	正常	创建类图识别项目，上传符合要求的类图图片，编辑类图图片，保存类图识别结果，验证类图识别功能的完整性
2	正常	创建顺序图识别项目，上传符合要求的顺序图图片，编辑顺序图图片，保存顺序图识别结果，验证顺序图识别功能的完整性
3	正常	创建类图与 Java 代码间的追踪项目，上传符合要求的图片、文档及代码，编辑图片、文档及代码，生成类图与 Java 代码间的追踪结果，保存图片、文档、代码及追踪结果，验证类图与 Java 代码间的追踪功能的完整性
4	正常	创建顺序图与 Java 代码间的追踪项目，上传符合要求的图片、文档及代码，编辑图片、文档及代码，生成顺序图与 Java 代码间的追踪结果，保存图片、文档、代码及追踪结果，验证顺序图与 Java 代码间的追踪功能的完整性

4.2.2 测试环境

本工具尚未部署到任何远程服务器或云平台，只能在本地开发环境中运行。本文实验的环境配置如表 4.11 所示。

表 4.11 环境配置

CPU	AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx	2.30 GHz
内存	8.00 GB	

操作系统	64-bit operating system, x64-based processor
浏览器	Google Chrome 版本 125.0.6422.77（正式版本） （64 位）

4.2.3 测试结果

(1)教程指导功能

测试目的：验证教程指导功能的可用性。确保功能可以正常使用。

表 4.12 教程指导功能测试结果

编号	功能	前提条件	输入/操作	预期结果	测试结果
1	查阅文档	如图 4.1 所示，点击左侧列表栏的“介绍”	查看文档内容	内容显示正常	与预期结果相符
2	播放视频	如图 4.1 所示，点击左侧列表栏的“操作指南”	点击播放视频按钮	播放正常	与预期结果相符

(2)项目管理功能

测试目的：验证项目管理功能的可用性。确保功能可以正常使用。确保出现异常情况时，工具有相应的提示信息。

表 4.13 项目管理功能测试结果

编号	功能	前提条件	输入/操作	预期结果	测试结果
1	创建项目	如图 4.2 所示，点击“+”号	输入项目名称，选择项目类型	创建成功	与预期结果相符
			输入重名项目，选择项目类型	创建成功	与预期结果相符
			不输入项目名称，选择项目类型	提示用户“输入项目名称”	与预期结果相符
			输入项目名称，不选择项目类型	提示用户“选择项目类型”	与预期结果相符
			不输入项目名称，不选择项目类型	提示用户“输入项目名称以及选择项目类型”	与预期结果相符
2	删除项目	如图 4.2 所示，点击“×”号	点击删除项目按钮	删除成功	与预期结果相符

(3)类图识别功能

测试目的：验证类图识别功能的可用性及完整性。确保上传符合要求的图片时，功能可以正常使用。确保出现异常情况时，工具有相应的提示信息。

表 4.14 类图识别功能测试结果

编号	功能	前提条件	输入/操作	预期结果	测试结果
1	上传图片	创建类图识别项目	上传 JPG/PNG 图片	上传成功	与预期结果相符
			上传 TXT 文档	提示用户“上传 JPG/PNG 图片”	与预期结果相符
2	识别图片	创建类图识别项目并上传 JPG/PNG 图片	将符合 UML2.5 标准图片发送至服务器	左侧正确展示识别结果，右侧正确展示原图	与预期结果相符
			将不符合 UML2.5 标准图片发送至服务器	左侧不展示识别结果并提示用户“图片不符合要求，识别失败”，右侧正确展示原图	与预期结果相符
3	编辑图片	创建类图识别项目并上传符合 UML2.5 标准的 JPG/PNG 图片	编辑类、类属性、类方法及方法参数文字	编辑成功	与预期结果相符
			添加类、类属性、类方法及方法参数	添加成功	与预期结果相符
			删除类、类属性、类方法及方法参数	删除成功	与预期结果相符
			从工具栏拖动线条或矩形至画布上	拖动功能有效，画布上能够正常显示	与预期结果相符
4	保存识别结果	创建类图识别项目，然后上传符合 UML2.5 标准的 JPG/PNG 图片，并将图信息恢复到与原图一致	点击保存 SVG 图片按钮	成功保存 SVG 图片至本地	与预期结果相符

(4)顺序图识别功能

测试目的：验证顺序图识别功能的可用性及完整性。确保上传符合要求的图片时，功能可以正常使用。确保出现异常情况时，工具有相应的提示信息。

表 4.15 顺序图识别功能测试结果

编号	功能	前提条件	输入/操作	预期结果	测试结果
1	上传图片	创建顺序图识别项目	上传 JPG/PNG 图片	上传成功	与预期结果相符
			上传 TXT 文档	提示用户“上传 JPG/PNG 图片”	与预期结果相符
2	识别图片	创建顺序图识别项目并上传 JPG/PNG 图片	将符合 UML2.5 标准图片发送至服务器	左侧正确展示识别结果，右侧正确展示原图	与预期结果相符
			将不符合 UML2.5 标准图片发送至服务器	左侧不展示识别结果并提示用户“图片不符合要求，识别失败”，右侧正确展示原图	与预期结果相符
3	编辑图片	创建顺序图识别项目并上传符合 UML2.5 标准的 JPG/PNG 图片	编辑对象及信息文字	编辑成功	与预期结果相符
			添加对象及信息	添加成功	与预期结果相符
			删除对象及信息	删除成功	与预期结果相符
4	保存识别结果	创建顺序图识别项目，然后上传符合 UML2.5 标准的 JPG/PNG 图片，并将图信息恢复到与原图一致	点击保存 SVG 图片按钮	成功保存 SVG 图片至本地	与预期结果相符

(5)类图与 Java 代码间的追踪功能

测试目的：验证类图与 Java 代码间的追踪功能的可用性及完整性。确保上传符合要求的文件时，功能可以正常使用。确保出现异常情况时，工具有相应的提示信息。

表 4.16 类图与 Java 代码间的追踪功能测试结果

编号	功能	前提条件	输入/操作	预期结果	测试结果
1	上传文件	创建类图与 Java 代码间的追踪项目	上传 SVG 图片	上传成功	与预期结果相符

			上传 TXT 文档	提示用户“上传 SVG 图片”	与预期结果相符
			上传 TXT/XLS 文档	上传成功，客户端负责将 XLS 文档转换成 JSON 格式，同时将上传的文件列表转换成 Map 结构	与预期结果相符
			上传 MD 文档	客户端过滤掉不符合格式要求的文档，同时将上传的文件列表转换成 Map 结构	与预期结果相符
			上传 JAVA 代码	上传成功，客户端负责将上传的文件列表转换成树结构再转换成 Map 结构	与预期结果相符
			上传 C 代码	客户端过滤掉不符合格式要求的代码，同时将上传的文件列表转换成树结构再转换成 Map 结构	与预期结果相符
2	查看文件	创建类图与 Java 代码间的追踪项目并上传符合要求的文件	点击文档文件名称	文件名称为 key，文件内容为 value，通过 Map 结构提取文档内容，能够正常显示文档内容	与预期结果相符
			点击代码文件名称	文件名称为 key，文件内容为 value，通过 Map 结构提取代码内容，能够正常显示代码内容	与预期结果相符
3	获取追踪结果	创建类图与 Java 代码间的追踪项目并上传符合要求的文件	点击获取追踪结果按钮	正确显示追踪信息	与预期结果相符
			点击追踪结果的文件名称	客户端负责打开对应代码文件，通过 Map 结构提取代码内容，能够正常显	与预期结果相符

				示代码内容	
4	编辑文件	创建类图与 Java 代码间的追踪项目并上传符合要求的文件	编辑图片	编辑成功	与预期结果相符
			编辑文档	编辑成功	与预期结果相符
			编辑代码	编辑成功	与预期结果相符
5	保存文件	创建类图与 Java 代码间的追踪项目并上传符合要求的文件	点击保存图片按钮	成功保存 SVG 图片至本地	与预期结果相符
			点击保存文档按钮	成功保存压缩后的 ZIP 文档至本地	与预期结果相符
			点击保存代码按钮	成功保存压缩后的 ZIP 代码至本地	与预期结果相符
6	保存追踪结果	创建类图与 Java 代码间的追踪项目，然后上传符合要求的文件，并生成追踪结果	点击保存追踪结果按钮	成功保存 JSON 格式的追踪结果至本地	与预期结果相符

(6)顺序图与 Java 代码间的追踪功能

测试目的：验证顺序图与 Java 代码间的追踪功能的可用性及完整性。确保上传符合要求的文件时，功能可以正常使用。确保出现异常情况时，工具有相应的提示信息。

表 4.17 顺序图与 Java 代码间的追踪功能测试结果

编号	功能	前提条件	输入/操作	预期结果	测试结果
1	上传文件	创建顺序图与 Java 代码间的追踪项目	上传 SVG 图片	上传成功	与预期结果相符
			上传 TXT 文档	提示用户“上传 SVG 图片”	与预期结果相符
			上传 JSON 文档	上传成功，客户端负责将上传的文件列表转换成 Map 结构	与预期结果相符
			上传 MD 文档	客户端过滤掉不符合格式要求的文档，同时将上传的文件列表转换成	与预期结果相符

				Map 结构	
			上传 JAVA 代码	上传成功，客户端负责将上传的文件列表转换成树结构再转换成 Map 结构	与预期结果相符
			上传 C 代码	客户端过滤掉不符合格式要求的代码，同时将上传的文件列表转换成树结构再转换成 Map 结构	与预期结果相符
2	查看文件	创建顺序图与 Java 代码间的追踪项目并上传符合要求的文件	点击文档文件名称	文件名称为 key，文件内容为 value，通过 Map 结构提取文档内容，能够正常显示文档内容	与预期结果相符
			点击代码文件名称	文件名称为 key，文件内容为 value，通过 Map 结构提取代码内容，能够正常显示代码内容	与预期结果相符
3	获取追踪结果	创建顺序图与 Java 代码间的追踪项目并上传符合要求的文件	点击获取追踪结果按钮	正确显示追踪信息	与预期结果相符
			点击追踪结果的文件名称	客户端负责打开对应代码文件，通过 Map 结构提取代码内容，能够正常显示代码内容	与预期结果相符
4	编辑文件	创建顺序图与 Java 代码间的追踪项目并上传符合要求的文件	编辑图片	编辑成功	与预期结果相符
			编辑文档	编辑成功	与预期结果相符
			编辑代码	编辑成功	与预期结果相符
5	保存文件	创建顺序图与 Java 代码间的追踪项目并上	点击保存图片按钮	成功保存 SVG 图片至本地	与预期结果相符

		传符合要求的文件	点击保存文档按钮	成功保存压缩后的 ZIP 文档至本地	与预期结果相符
			点击保存代码按钮	成功保存压缩后的 ZIP 代码至本地	与预期结果相符
6	保存追踪结果	创建顺序图与 Java 代码间的追踪项目，然后上传符合要求的文件，并生成追踪结果	点击保存追踪结果按钮	成功保存 JSON 格式的追踪结果至本地	与预期结果相符

(7)界面交互功能

测试目的：验证折叠面板组件和拖拽组件功能的可用性，确保功能可以正常使用。

表 4.18 界面交互功能测试结果

编号	说明	前提条件	输入/操作	预期结果	测试结果
1	纵向折叠面板功能	创建类图与 Java 代码间的追踪项目或创建顺序图与 Java 代码间的追踪项目	点击收起按钮	折叠相应的展示区域	与预期结果相符
			点击展开按钮	展开相应的展示区域	与预期结果相符
2	横向折叠面板功能	创建类图与 Java 代码间的追踪项目或创建顺序图与 Java 代码间的追踪项目	点击收起按钮	折叠相应的展示区域	与预期结果相符
			点击展开按钮	展开相应的展示区域	与预期结果相符
3	拖拽功能	创建类图与 Java 代码间的追踪项目或创建顺序图与 Java 代码间的追踪项目	拖动条框	调整展示区域间各占的宽度尺寸	与预期结果相符

4.3 RSE-CDI 算法优化实验评估

4.3.1 评估设计

(1)评估目标

为了证实第 2 章 RSE-CDI 算法精度优化的矩形识别优化、关系线识别优化及关系符号识别优化的有效性，本文将构造测试数据集，并按照评估算法的指标标准，分析原有算法逻辑以及优化算法的测试结果，以验证优化算法相对于原有算法是否提升了召回率和准确率。

(2)测试数据集

用于评估算法性能的数据集，规定样本是由 StarUML 建模工具绘制的符合 UML2.5 标准的类图。

根据优化方向，需要关注以下维度：对于矩形识别优化需要关注矩形的数量；对于关系线识别优化需要关注线条包括实线和虚线的数量；对于关系符号识别优化需要关注关系符号包括继承关系、实现关系、依赖关系和聚合关系的数量。

本文将记录算法识别的结果，并人工检查识别结果以记录算法正确识别结果、算法漏检结果及算法误检结果。结果记录需要关注 7 个维度：矩形的数量记作 *RECT*、实线的数量记作 *SOL*、虚线的数量记作 *DASH*、继承关系的数量记作 *GEN*、实现关系的数量记作 *RLZ*、依赖关系的数量记作 *DPD*、聚合关系的数量记作 *AGG*。

算法正确识别结果记录为 $RECT_{TP}$、SOL_{TP}、$DASH_{TP}$、GEN_{TP}、RLZ_{TP}、DPD_{TP}、AGG_{TP}；算法漏检结果记录为 $RECT_{FN}$、SOL_{FN}、$DASH_{FN}$、GEN_{FN}、RLZ_{FN}、DPD_{FN}、AGG_{FN}；算法误检结果记录为 $RECT_{FP}$、SOL_{FP}、$DASH_{FP}$、GEN_{FP}、RLZ_{FP}、DPD_{FP}、AGG_{FP}。

为了方便理解，下面以识别矩形的结果为例，说明其召回率和准确率的计算方法。

例子：采用图识别算法识别 1 张 StarUML 类图（透明背景）的矩形，其中有 40 个是真实存在的矩形。

TP：算法正确识别的矩形数量。

FP：算法错误识别的矩形数量。

FN：算法遗漏识别的矩形数量。

如果实际存在的矩形有 40 个（TP + FN = 40），其中 30 个被正确识别，10 个被漏识别（FN = 10），则召回率为：

$$Recall = \frac{RECT_{TP}}{RECT_{TP} + RECT_{FN}} = \frac{30}{30 + 10} = 0.75$$

如果算法识别出了 50 个矩形，其中 40 个是真实存在的（TP = 40），但误识别了 10 个（FP = 10），则准确率为：

$$Precision = \frac{RECT_{TP}}{RECT_{TP} + RECT_{FP}} = \frac{40}{40 + 10} = 0.8$$

计算关系线及关系符号的召回率和准确率的方法与上述相同。不同之处在于，识别矩形的类图要求是透明背景。

F Chen 等人的工作提供了各种建模工具绘制的类图。为了方便评估，本文统一从

中选取了 5 张由 StarUML 建模工具绘制的带透明背景的类图作为数据集，并统计了这 5 张类图在 7 个维度上的数量分别为矩形数量、实线数量、虚线数量、继承关系数量、实现关系数量、依赖关系数量及聚合关系数量，见表 4.19。

表 4.19 5 张测试类图在 7 个维度上的数量

	矩形	实线	虚线	继承关系	实现关系	依赖关系	聚合关系
原图	48	20	26	15	3	23	5

(3)评估指标

本文选择了召回率和准确率来度量算法性能。主要原因在于它们能够有效衡量算法在正样本（即需要识别的目标）检测方面的表现。

召回率高意味着算法能识别出大多数的目标对象，即漏检率低。对于图识别任务来说，召回率能够反映算法在检测所有目标对象方面的能力，尤其适用于对漏检容忍度低的应用场景。

准确率高意味着算法误检的样本较少，即识别结果更为可靠和精确。

对于图识别任务来说，准确率能够反映算法在准确识别目标对象方面的能力，尤其适用于对误检容忍度低的应用场景。召回率和准确率的定义和公式如表 4.21 所示。在了解它们之间需要先了解 TP、TN、FP、FN 的含义如表 4.20 所示。

表 4.20 TP、TN、FP、FN 的含义

	含义
TP(True Positive)	真正例。指实际为正例的样本被正确分类为正例的数量。
TN(True Negative)	真负例。指实际为负例的样本被正确分类为负例的数量。
FP(False Positive)	假正例。指实际为负例的样本被错误分类为正例的数量。
FN(False Negative)	假负例。指实际为正例的样本被错误分类为负例的数量。

表 4.21 召回率和准确率

	定义	公式
召回率	召回率是指在所有实际为正的样本中，被正确识别为正的样本所占的比例	$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$
准确率	准确率是指在所有被识别为正的样本中，实际为正的样本所占的比例	$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$

4.3.2 评估结果

使用了原有算法及优化算法对所选数据集进行识别，得出并统计了这 5 张类图的识别结果，如表 4.22 所示。

表 4.22 原有算法及优化算法识别结果在 7 个维度上的数量

	矩形	实线	虚线	继承关系	实现关系	依赖关系	聚合关系
原有算法识别结果	88	3	1	2	0	0	0
优化算法识别结果	51	15	29	4	0	37	2

有了所有数据后，按照表 4.21 公式计算出原有算法及优化算法的召回率和准确率。有一点需要注意的是，不同的图片有不同的阈值，若阈值设置的恰当，则召回率和准确率会相应提升，而本次实验是在同个阈值情况下，对 5 张类图进行评估得出的结果，如表 4.23 所示。相比原有算法，优化算法的召回率显著提升，其原因在于关系线漏检的可能性较小，因此关系符号漏检的可能性降低了。相比原有算法，优化算法的准确率接近，其原因在于矩形误检的可能性较小，但是虚线误检的可能性较高，造成依赖关系误检的可能性增加了。这个现象说明了优化算法在不影响准确率的前提下，其召回率有所提升，它能识别出大多数的目标对象，特别适用于漏检容忍度低的应用场景。

表 4.23 原有算法及优化算法的召回率和准确率

		矩形	实线	虚线	继承关系	实现关系	依赖关系	聚合关系
召回率	原有算法	=100%	=15%	≈4%	≈13%	=0%	=0%	=0%
	优化算法	=100%	=75%	=100%	≈27%	=0%	=100%	=40%
准确率	原有算法	≈55%	=100%	=100%	=100%	=100%	=100%	=100%
	优化算法	≈94%	=100%	≈90%	=100%	=100%	≈62%	=100%

4.4 工具效率分析

为了验证使用本工具后有效地节省了时间和人力成本，本文将构造符合测试要求的数据集，通过实验结果证实工具的效率有效性。

由于从类图与代码间的追踪测试结果就可以验证使用工具后效率是否有所提升，所以本文只对类图与代码间的追踪进行测试。首先构造符合测试要求的数据集，本工具基于遵守 UML2.5 标准的图片、设计文档及代码来建立类图与代码间的追踪关系。根据这个原则，本文在 F Chen 等人发布的开源项目中发现其中保存了大量的测试数据集，于是从中选取了三个项目分别为 Hadoop HDFS、ASM 及 Jetty 作为数据集，项目规模如表 4.24 所示。但需要注意的是，它们所保存的设计模型图不规范，需要使用 StarUML 工具重新绘制。还有需要补充的一点是借助文档能使建立的追踪关系更为准确，可是会更耗时。

表 4.24 项目规模

项目名称	图片				文档		代码	
	类	属性	方法	关系	数量	总字符数	数量	总行数
Hadoop HDFS	8	0	0	25	1	9545	2424	509748
ASM	17	141	0	27	1	1428	310	50078
Jetty	9	2	8	10	2	345990	3019	441918

接着分析使用工具后所花费的时间，假设开发人员使用工具，那么他们只需要手动编辑图片，并能够借助工具自动建立追踪关系，因此仅需考虑图片的复杂性以及网络延迟对自动建立追踪关系所造成的影响两个因素。为了提供一个概括性的测试结果，本文对上述选取的三个项目分别做了三次实验，得出它们每个阶段所花费的时间后再计算出它们的平均值，见附录 A。三次实验皆由本人亲自操作，为了减少误差所以本人进行了三次而不是一次，于手动编辑图工作而言，以加载图片完成作为开始时间，编辑图片完成作为结束时间，得出所花费的时间。于自动建立类图与代码间的追踪关系工作而言，以发送请求作为开始时间，接收响应作为结束时间，得出所花费的时间。最终通过两个工作所花费的平均时间之和得出总花费时间，如表 4.25。

表 4.25 使用工具后所花费的时间

项目名称	工作流程	是否借助文档	花费时间
Hadoop HDFS	手动编辑图，自动建立类图与代码间的追踪关系	是	$313.33 + 163.67 = 477$ 秒 ≈ 8 分钟
		否	$313.33 + 34.67 = 348$ 秒 ≈ 6 分钟
ASM	手动编辑图，自动建立类	是	$1122.67 + 9.67 = 1132.34$ 秒 ≈ 19 分钟

	图与代码间的追踪关系	否	$1122.67 + 9.67 = 1132.34$ 秒 ≈ 19 分钟
Jetty	手动编辑图，自动建立类图与代码间的追踪关系	是	$199.33 + 88.33 = 287.66$ 秒 ≈ 5 分钟
		否	$199.33 + 30 = 229.33$ 秒 ≈ 4 分钟

得出使用工具后所花费的时间后，本文从结果中选取了花费最短及花费最长的时间作为最终花费的时间范围，得出了表 4.26 的结果。已知使用工具后所花费的时间以及基于项目代码量和人类的劳动力，可以确定使用工具前所花费的时间会大于使用工具后所花费的时间，因此得以证实使用本工具后，有效地节省了时间和人力成本。

表 4.26 效率分析

	工作流程	是否借助文档	花费时间范围
使用工具后	手动编辑图，自动建立图与代码间的追踪关系	是	$5 \sim 19$ 分钟
		否	$4 \sim 19$ 分钟

总结与展望

目前，本工具采用了如下技术以支持功能需求的实现：(1)采用 SCSS 编写样式文件，(2)采用 TS 定义变量及函数的类型，(3)采用 AntD 库实现部分功能，如按钮组件实现点击功能及标签页组件实现创建多个项目功能，(4)采用 GoJS 库实现类图编辑功能和顺序图编辑功能，(5)采用 Draft.js 库实现文档编辑功能，(6)采用 Ace Editor 库实现代码编辑功能，(7)采用 Next.js 官方提供的脚手架工具创建了 Next.js 应用，(8)使用 JSX 语法编写 React 组件，(9)采用 Zustand 库管理模块的状态，(10)采用 RSE-CDI 和 RSE-SDI 算法实现图识别功能，(11)采用 CDTC 和 SDTC 算法实现图与代码追踪功能，(12)在算法基础上搭建 Tomcat 服务器，Tomcat 服务器负责托管和运行 Servlet，并提供 HTTP 服务器功能，而 Servlet 程序负责处理客户端的请求并生成响应。

目前，本工具暂未使用数据库技术，因为现有的功能需求不需要数据库支持。具体而言：(1)当前工具的所有功能均可通过非数据库技术实现，并且能够满足用户需求，(2)目前的功能不需要存储用户信息，工具主要提供可视化界面，方便用户操作。

虽然当前的功能需求已经得到满足，但为了确保未来迭代开发的顺利进行，并增强工具的可扩展性，本文通过代码实现验证了 PostgreSQL 数据库与目前所用技术的高适配度。在后续开发中，如果需要使用数据库，推荐选择 Postgres 数据库来存储数据。

同样地，当前的功能需求未能真正发挥出 Next.js 框架的优势（如 SSG 和 SSR）。然而，考虑到后续的发展，本工具选择了这些技术以支撑未来的计划。

最后，作为一个软件架构设计图与代码追踪工具，在后续的工作中可以往四个方向发展：(1)扩展软件架构设计图种类，(2)扩展代码种类，(3)扩展或优化图识别算法，(4)扩展或优化图代码追踪算法。(1)和(2)可以使工具适应各种应用场景，更有包容度。(3)和(4)可以使识别结果和追踪结果更为全面且准确。

致谢

时间飞逝，一转眼就快要毕业了。回顾这四年的大学生活，我经历了巨大的压力、无数的难关和各种各样的挑战。在此，我非常感谢一路上帮助我的所有人，他们给予了我力量，使我无所畏惧地面对困难，一次又一次地跨过障碍，坚持到最后，并顺利完成了学业。

感谢指导我毕业论文的连小利老师，在我多次对题目定位感到模糊时，老师总是耐心地解释，帮助我抓住重点，将我被误导的思路引回正确的道路。在我毫无头绪时，会提供一些解决方案，这些方案帮助我拓宽思路并探索更多可能性，同时针对我的弱项，给予针对性指导，让我受益匪浅。我非常感激连小利老师，在老师身上，我不仅学到了宝贵的理论知识，还学到了待人处事的态度。

感谢国际学院的张燕老师，疫情期间，面对我无法到校只能上网课的窘境，是她积极和课程指导老师沟通，建立了微信群组，使我能够与老师们保持联系，并顺利上课。同时，我也要感谢计算机学院的老师和助教们，他们在微信群组中分享电子资料、视频录播等课程资源，时刻关注我们的学习进度，并耐心解答我们的疑惑，即使是在线上，他们也尽力为我们创造一个良好的学习环境，让我能够顺利完成所有课程。老师们的无私奉献和尽职尽责的态度，协助我们克服了线上学习的诸多困难，是支撑我不断学习的动力。

感谢陈劲安学长毫无保留地分享他在学习上所获得的宝贵经验，这让我面临相同问题时有了兜底方案，避免了慌张和手足无措。当我面临心理挑战或学习困境时，他会鼓励我，并主动伸出援手，言传身教地教导我。在我认为只能依靠自己突破种种难关时，他给予了我极大的信心，让我觉得还有一个人可以协助我渡过难关。感谢已毕业的陈方伟师兄，在百忙之中抽空帮我整理思路并指出关键点，让我能在短时间内理解论文的整体结构并掌握知识轮廓。他通俗易懂的解答方式，多次强调核心观点，帮我强化了概念，同时提出了多种可扩展的方案，为我提供了一些创新思路。

感谢我的同胞们在生活上给予的诸多帮助。感谢同学们在群组和学习论坛中积极表达自己的观点，解答疑问，这让我能从多个角度思考问题。

感谢我的父母提供经济支持以及不反对我跨洋来到中国留学，这使我得以开阔眼界，提升认知，进而激发我的各种可能性。也感谢我的姐姐在北京的陪伴，让我勇敢地走出国界，不至于只身一人担惊受怕。

回头看，轻舟已过万重山。我闯过了无数关卡，经历了种种磨练，这使我心灵和认知层面更上一层楼。我的成长离不开身边任何一个人的帮助，再次郑重地感谢大家！

参考文献

[1] F Chen, L Zhang, X Lian, et al. Automatically recognizing the semantic elements from UML class diagram images[J]. Journal of Systems and Software, 2022, 193: 111431.

[2] Antoniol, Giulio, et al. Design-code traceability for object-oriented systems[J]. Annals of Software Engineering 9.1 (2000): 35-58.

[3] F Chen, L Zhang, X Lian. CDTC: Automatically establishing the trace links between class diagrams in design phase and source code[J]. Softw: Pract Exper. 2024; 54(2): 281-307.

[4] M. Hammad, M. L. Collard and J. I. Maletic. Automatically identifying changes that impact code-to-design traceability[C]. 2009 IEEE 17th International Conference on Program Comprehension, Vancouver, BC, Canada, 2009, pp. 20-29.

[5] Selic, B., et al. Omg unified modeling language (version 2.5)[S]. Object Management Group, MA, USA (2015).

[6] Perry, Dewayne E., and Alexander L. Wolf. Foundations for the study of software architecture[J]. ACM SIGSOFT Software engineering notes 17.4 (1992): 40-52.

[7] 刘敏. 分层软件架构设计及其应用研究[J]. 移动信息, 2018,(08): 148-149.

[8] 余双双, 曾一, 刘慧君等. 基于 UML 模型的多态性与 Java 接口代码信息一致性检测的方法[J]. 计算机应用与软件, 2017, 34(02): 8-13.

[9] 王雷, 张帅, 宋慧娜. 基于 FSM 的 UML 模型与代码一致性动态检测[J]. 延安大学学报：自然科学版, 2021, 40(04): 54-60.

[10] 曾一, 李函逾, 刘慧君等. UML 模型和 Java 代码之间的一致性检测方法[J]. 计算机科学, 2015, 42(04): 151-155.

[11] 黄兴荣. 基于 B/S 架构模式的三层结构设计与实现[J]. 电脑知识与技术, 2015, 21(11X): 52-53.

[12] Arora, Ritu, Sanjay Goel, et al. Supporting collaborative software development over GitHub[DB/OL]. Software: Practice and Experience 47.10 (2017): 1393-1416.

[13] 蚂蚁集团. Ant Design 开源社区[Z]. Ant Design Documentation. https://ant.design/components/overview-cn/, 2024-02-26.

[14] Northwoods Software. GoJS Diagramming Components[Z]. https://gojs.net/latest/api/index.html. 2024-02-26.

[15] Vercel. Next.js Documentation[Z]. https://nextjs.org/docs. 2024-02-26.

[16] Lank, Edward et al. On-line recognition of UML diagrams[C]. Proceedings of Sixth International Conference on Document Analysis and Recognition (2001): 356-360.

[17] Karasneh, Bilal, and Michel RV Chaudron. Img2uml: A system for extracting uml models from images[C]. 2013 39th Euromicro Conference on Software Engineering and Advanced Applications. IEEE, 2013.

[18] Hammond, Tracy Anne and Randall Davis. Tahuti: a geometrical sketch recognition system for UML class diagrams[J]. ACM SIGGRAPH 2006 Courses (2006): n. pag.

[19] Chavez, Hector M., et al. An approach to checking consistency between UML class model and its Java implementation[C]. IEEE Transactions on software engineering 42.4 (2015): 322-344.

[20] Facebook. Draft.js Docs[Z]. https://draftjs.org/docs/getting-started/. 2020-10-16.

[21] Securingsincity. React-Ace[Z]. https://securingsincity.github.io/react-ace/. 2024-03-24.

[22] H Lai. Uml-Code-Consistency[Z]. https://github.com/011015/uml-code-consistency. 2024-06-05.

附录 A 使用工具后每个阶段所花费的时间

表 A1 Hadoop HDFS 每个阶段所花费的时间

	手动编辑图片	自动建立类图与代码追踪关系	
		有文档	无文档
第一次	310 秒	162 秒	42 秒
第二次	305 秒	153 秒	31 秒
第三次	325 秒	176 秒	31 秒
平均值	≈313.33 秒	≈163.67 秒	≈34.67 秒

表 A2 ASM 每个阶段所花费的时间

	手动编辑图片	自动建立类图与代码追踪关系	
		有文档	无文档
第一次	1157 秒	10 秒	10 秒
第二次	1048 秒	9 秒	9 秒
第三次	1163 秒	10 秒	10 秒
平均值	≈1122.67 秒	≈9.67 秒	≈9.67 秒

表 A3 Jetty 每个阶段所花费的时间

	手动编辑图片	自动建立类图与代码追踪关系	
		有文档	无文档
第一次	200 秒	88 秒	31 秒
第二次	195 秒	89 秒	31 秒
第三次	203 秒	88 秒	28 秒
平均值	≈199.33 秒	≈88.33 秒	=30 秒