2021 届本科生学士学位论文　　学校代码：10269

華東師範大學

East China Normal University

本科生毕业论文

面向开放组织的迷你项目看板

Mini Project Kanban for Open Organizations

姓　　名：梅佳奕

学　　号：10165300206

学　　院：数据科学与工程学院

专　　业：数据科学与大数据技术

指导教师：王伟

职　　称：研究员

2021　年　6　月

华东师范大学学位论文诚信承诺

本毕业论文是本人在导师指导下独立完成的，内容真实、可靠。本人在撰写毕业论文过程中不存在请人代写、抄袭或者剽窃他人作品、伪造或者篡改数据以及其他学位论文作假行为。

本人清楚知道学位论文作假行为将会导致行为人受到不授予/撤销学位、开除学籍等处理（处分）决定。本人如果被查证在撰写本毕业论文过程中存在学位论文作假行为，愿意接受学校依法作出的处理（处分）决定。

承诺人签名:　　日期：　　年　月　日

华东师范大学学位论文使用授权说明

本论文的研究成果归华东师范大学所有，本论文的研究内容不得以其它单位的名义发表。本学位论文作者和指导教师完全了解华东师范大学有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权华东师范大学可以将论文的全部或部分内容编入有关数据库进行检索、交流，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

保密的毕业论文（设计）在解密后应遵守此规定。

作者签名:　　导师签名：　　日期：　　年　月　日

目录

面向开放组织的迷你项目看板

摘要：

随着互联网技术的发展，越来越多的企业开始逐渐从传统的封闭式组织转变为开放式组织，而开源作为一种典型的开放式组织管理形式，正在软件开发领域发挥巨大的作用，推动着传统生产方式的变革。随之而来的是对开放组织与开源社区治理的更大需求，于是，解读开源社区现状、推动社区机制完善成为了目前极具价值的研究方向。

Hypertrons 是 X-lab 开放实验室发起的一个开源的跨平台 RPA 平台，在提供跨平台交互能力的同时，为了能更好的做出决策，也采集了各平台的开放数据作为决策基础。这些数据同时也为社区的数字化运营分析提供了有效的支撑。本文介绍了 Hypertrons 在前端方面的功能延申部分——一个面向开放组织的迷你看板，来更好地以交互的方式展现对开放组织与开源社区的数据分析结果、辅助用户决策。

面向开放组织的迷你看板采用了浏览器插件的形式，通过在特定网址的网页中插入可视化组件并提供交互手段，在无需深度改变用户的使用习惯的同时，为开放组织提供有效的监控和运营助力。

关键词： 开源，可视化

Mini Project Kanban for Open Organizations

Abstract:

With the development of Internet technology, companies are now gradually transforming their management mode from the traditional closed type to the open type, which is called open source. Open source platform plays an important role in the field of software development, and it contribute to the transformation of traditional production methods. These transformation leads to a greater demand for the governance of open organizations and open source communities. Therefore, explaining the current situation of the open source community and promoting the improvement of the community mechanism has become a valuable research direction.

Hypertrons is an open source RPA(Robotic Processing Automation) platform initiated by X-lab. While providing cross-platform interactive functions, it also collects open data from each platform as the basis of decision-making. These data effectively support the analysis of digital operations in open source communities. In this article, we will introduce the front-end extension of Hypertrons , which is a mini kanban for open organizations, aiming at better data visualization features.

The mini kanban for open organizations is developed as a browser extension. It inserts visual components into web pages with specific URLs as well as provides interactive methods. As a result, it can provide effective monitoring and operational assistance for open organizations without the need to change user habits greatly.

Keywords： Open Source, Visualization

1. 绪论

1.1 背景

随着互联网技术的发展，越来越多的企业开始逐渐从传统的封闭式组织转变为开放式组织。传统的自上而下、封闭式、命令式与控制式的组织管理方式正在走向终结，而开放透明、高效敏捷的开放式组织管理方式正在成为新的潮流。一方面，互联网技术为开放访问、开放协议以及开放协作的文化提供了基础保障。技术的发展使得全世界的知识型人才能够通过互联网进行沟通和交流，并使得他们能够平等和自由的相互分享和贡献自己的盈余资源，如想法、创意以及才智等。大家在自发交互沟通中达成共识并创造出群体智慧，产生了如维基百科、开源软件开发平台（如 GitHub）等创新型产品和服务[1]。另一方面，全球经济数字化浪潮正蓬勃发展，以移动通信、物联网、人工智能等为代表的新一代信息技术将人－业务－系统－组织－企业连接成一个网络，传统的业务系统从一个简单的机械系统演变成一个复杂的生态系统，企业在从局部优化走向全局优化的过程中，不断变迁的生产方式推动着企业管理模式的变化，而这些变化都驱动着企业组织不断被重构与再定义[2]。

近些年来，开源作为一种典型的开放式组织管理形式，在软件开发领域发挥了巨大作用，推动了现有的生产方式的变革。长期以来，产品、项目形式的专有软件开发是软件生产的主流，其特点是精英化（专业骨干制作软件）、计划性（预先规划需求/自顶向下开发）、封闭性（开发过程不对外开放）、许可证（商业模式通过销售软件产品获利）。这种软件生产方式的问题在于开发成本过高、开发周期较长、补丁漏洞防不胜防。与之相反，开源软件的开发者通过一种开放的软件许可协议保护其著作权，其源代码、设计文档、开发日志等数据允许用户自由的学习、修改和传播。开源软件的开发模式不仅是一种依托开放、开源的软件合作社群而且正向基于分享、交互与群体智能的同侪生产方式发展，其特点是开发过程从封闭到开放、开发人员从精英到大众、开发组织从工厂到社群、开发成果从产品到服务。在此方式下，使用者与设计者、开发者、维护者之间不再壁垒森严，维基百科等已证明了这种生产方式对于知识型制品的优越性。[3]

开源软件经过了 20 多年的发展，已经演化出诸如 Linux、MySQL、Hadoop、Kubernetes、TensorFlow、React、VS Code 这些数字化的基础设施，而且整个技术栈还在不断向上生长，吞噬着整个软件世界，并形成了一个开源生态，一个开源文明[4]。由代码、数据、开发者行为、社区规范等等所组成的开源数字生态系统，就像一个新物种一样，不断演化与成长，支持着整个数字世界的基础设施，也在逐渐改变人类的协作与行为方式。开源协作平台也将成为一个海量数据市场，在这个市场里，将有大量透明化的信息，同时拥有做出决策和进行协作的数字工具。这一切将产生巨大的影响，不只是对组织和管理者，而且对所有生态中的各类参与者，包括开发者、社区经理、工程师、还有消费者[5]。

开源软件开发者分布在全球不同位置，进行相对独立的软件开发，开发者的立项、讨论、评价、修改、测试等行为，主要是通过互联网相互沟通、讨论和协商实现[6]。开源软件开发过程中的交互方式不同于基于企业内部层级式组织的专有软件开发，而是依

靠自组织、分布式的开源社区，通过项目发起者协调，这就导致质量、成本这些因素成为在软件开发过程中受交互行为影响的外生变量，而不同于专有软件的厂商自己决定的内生变量。[3] 因此，开源需要有一整套的治理框架，基于此框架的数字化、自动化的治理工具来帮助开源项目及社区更高效的运作和管理。

开源的本质是开放式组织管理。开放式组织中的人员以兴趣爱好聚集，可快速地加入和退出，人员流动较快。但却不能缺少传统企业运营中的每一个环节，例如人员招聘、项目管理、人力绩效、工资分红、退出机制、宣传营销等[7]。因此，开放式组织需要有一套治理框架。框架中包含有松耦合的进入与退出机制、完善的数字化管理监控体系，从宣传、活动、用户、开发、管理、布道等各个方面进行数字化管理和常态化的社区角色快速转换，并有完善的自动化机制保证社区角色快速转换中足够低的学习成本和转换成本[8]。

本课题正是致力于完善开源社区中的数字化治理框架，结合机器人流程自动化的形式，实现一个浏览器插件的看板应用，协助开源社区中不同项目、不同视角的决策与运营。

1.2 相关工作

为了完善开源社区的架构、更好地治理开源社区，使其成为如今商业社会中更具价值的数字化生产社区，开源开发者们在社区治理方面做了许多工作，他们的力量回馈到开源社区，成为了新的发展方向。这其中，最显著的变化便是 Github Apps 自 2016 年推出以来的迅猛发展。

Github Apps 也称为自动协作机器人，它们本身是 Github 的独立账号，用户名以 [bot] 结尾，在开源社区中拥有独立的认证身份；但其实它们本质上是一种程序，能够自动化地帮助用户使用 Github 提供的认证信息、调用 Github API 来完成一系列相对复杂的操作，如检查代码合并、签署许可协议等，以减少重复的人工劳作。GitHub Apps 的优点有很多，一个比较突出的优势是，GitHub Apps 可以做到对权限的精细管理，比如负责持续集成的 GitHub Apps 可以请求对仓库内容的读取权限和对状态的写入权限，又比如一个 GitHub Apps 可能没有对代码的读取或写入权限，但仍能管理 issues、labels 和 milestones。

在目前 Github 平台全球最活跃开发者账号中，大部分为 GitHub Apps。据多年的数据统计，GitHub Apps 年活跃账号数量（活跃数量）与所产生日志总量占全年日志占比（日志占比）的变化如图 1.2 所示：从日志占比来看，2019 年相较于 2018 年提升了 288%，2020 年相较 2019 年增长 141%，达到了 12% 以上。总体而言，Github Apps 的流行趋势不可遏制，这也侧面反映着开源社区的发展进程对 Github Apps 这一类智能化治理的功能需求日益增加。

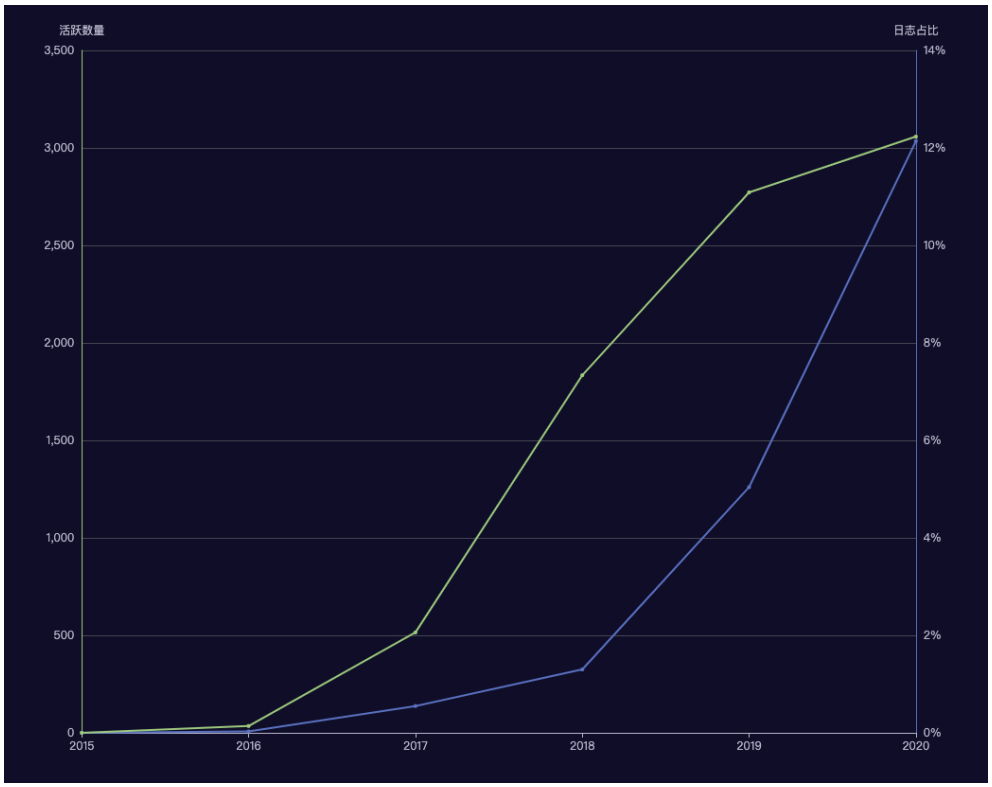

图 1-1: GitHub Apps 活跃账号数量与日志占比
Figure 1-1: The number of GitHub Apps active accounts and the proportion of logs

在这样的大环境之下，由华东师范大学与同济大学跨校组建的实验室 X-lab 设计并发起了一个面向开源项目的 RPA(Robotic Process Automation 机器人流程自动化）平台——Hypertrons，致力于向开源代码平台提供流程自动化定制与运行的能力。在开源社区中，Hypertrons 扮演着开源项目和开放组织中的一个智能的数字劳工的角色，承担一些流程的自动的构建与编排，为跨平台的组织管理工作提供便利。在功能上，它依托于对全域开源行为的开放数据分析结果，帮助开发者与社区管理者进行决策，其方式是提供一个高度可配、高度可移植的框架，供用户在项目级别自定义组件，以配合不同项目的管理流程与制度。在开发者通常使用的各个平台（如 Git、Gitlab、Github、Gitee、Slack、Jenkins、邮件、石墨文档和各类线上服务）之间，Hypertrons 提供了跨平台交互的能力，同时为了更好地做出决策，也采集各平台的开放数据作为决策基础，为社区的数字化运营分析提供了有效的支撑。

在 Hypertrons 项目中，不论是在项目层面对于用户行为数据的采集，还是将数据分析结果呈现给用户，都是一个无法避免地与用户产生联结、进行交互的过程。于是，为了更好地以交互的方式展现 Hypertrons 社区的数字化运营分析结果、辅助用户决策，面向开放组织的迷你看板项目应运而生。由于不希望深度改变用户的使用习惯，故发起一个浏览器插件项目，通过在特定网址的网页中插入迷你看板的方式为开放组织提供有效的监控和运营手段，这个项目是 Hypertrons 一大创新，也是 Hypertrons 的前端展示模块与功能延伸，即重要的组成部分。

1.3 拟解决的主要问题

一方面，迷你项目看板作为 Hypertrons 在浏览器端的入口之一，是 Hypertrons 的多平台集成与扩展能力的展示与增强。另一方面，通过该项目，可以非常方便且低成本地进行精细化市场运营：掌握用户的行为数据和兴趣偏好等重要信息，从而针对不同用户进行个性化的数字运营。

经调研，在目前的各大代码托管平台中，仅项目页面会显示一些低阶指标数据（如 Star 数、贡献者列表等）与有限的的高阶指标数据（如 GitHub Insights 页面）。使用这些平台自带的功能，用户无从了解更多的高阶数据指标（如项目活跃度、流行度、贡献者活跃度、Issue 平均响应时间等），从而无法很好或快速地掌握项目状况，进行管理或协同工作。而市面上已有的同类型看板应用，虽然也有用于项目健康度衡量分析的指标，但它们都以独立部署的平台、网站的形式呈现，无法很好地在此场景下为不同角色、不同平台的用户服务。于是，以浏览器插件形式存在的迷你看板成为了该场景下的一种更优解。

根据 Hypertrons 的需求与迷你项目看板的产生动机，本课题关注的主要问题有两个：一是如何将开源社区中的开放数据进行分析呈现，即，可视化给用户；二是提供一种简单易用的交互方法，接收用户管理与配置的定制化指令（也即用户行为数据）。

1.4 论文组织

本文在第一章介绍了课题的相关背景与产生动机，对应地提出拟解决的主要问题；在第二章中，将给出面向开放组织的数据挖掘框架，并定义相关方法和衡量指标；接下来的第三章，会根据前面的定义，对开放平台的开源数据进行探索、分析；第四章中主要对面向开放组织的迷你看板进行了整体设计；而第五章从技术的角度，详述了迷你看板作为浏览器插件的实现细节。

2. 面向开放组织的数据信息挖掘框架

对于开源开发者而言，通过个人行为数据的汇总挖掘，能够更好地自我了解与进行职业规划；对于开源项目而言，这些数据将提供不同的维度来评估其影响力、国际化程度，即便对于项目不熟悉的人，也能在很短的时间内快速了解项目状况，更可以方便项目管理与社区治理[9]。

2.1 活跃仓库与开发者统计方法

若被统计的代码仓库包含事件日志，仓库即被定义为活跃仓库；若开发者有任意代码仓库包含事件日志，开发者即被定义为活跃开发者。

2.2 开发者活跃度

开发者活跃度，其定义为某特定 GitHub 账号在一段时间内在某特定 GitHub 项目中的活跃评价指标。其活跃度由该账号在该项目中的行为数据决定，本文中所关心的行为包含如下几种：[10]

- Issue comment：在 issue 中参与讨论是最基本的行为，每个评论计入 1 次。
- Open issue：在项目中发起一个 issue，无论是讨论、bug 报告或提问，对项目都是带来活跃的，每个发起的 issue 计入 1 次。
- Open pull request：为项目提交一个 PR，表示已对该项目进行源码贡献，则每次发起一个 PR 计入 1 次。
- Pull request review comment：对项目中的 PR 进行 review 和讨论，需要对项目有相当的了解，并且对项目源码的质量有极大帮助，每个评论计入 1 次
  注：仅通过代码 review 对特定代码行的讨论记为 review，直接对 PR 的评论回复记为 issue comment 事件。
- Pull request merged：若有 PR 被项目合入，即便是很小的改动，也需要对项目有较为深入的理解，是帮助项目进步的真切贡献，则每有一个 PR 被合入计入 1 次，同时 PR 合入事件根据该 PR 的代码增加行数。
- Watch：用户 star 项目，被计入 1 次。注：Star 操作在 GitHub 日志事件中记为 Watch 事件。
- Fork：开发者 fork 该项目，被计入 1 次。

以上 7 个种行为各自独立计数，具有不一样的权重，根据专家经验，加权值分别为 1、2、3、4、2，1，2，即：

$$A_{u\_d} = C_{issue\_comment} + 2C_{open\_issue} + 3C_{open\_pr} + 4C_{review\_comment} + 2C_{pr\_merged} + C_{watch} + 2C_{fork}$$

其中，Pull request merged 的计数由分段函数决定[11]：

$$C_{pr\_merged} = \begin{cases} 0.8 + 0.002 \times loc & loc < 100 \\ 1 & 100 \leq loc < 300 \\ 2.5 - 0.005 \times loc & 300 \leq loc < 400 \\ 0.5 & loc \geq 400 \end{cases}$$

其中，loc 表示新增的代码行数。根据软件计量学经典数据统计，单次代码变更最佳在 200 行以内，超过 400 行的代码变更会导致审阅困难，故通过分段函数关联了代码新增行数与 PR 的权重指标。

2.3 项目活跃度

项目活跃度定义为某特定项目在一段时间内的活跃评价指标。其活跃度由单开发者在一天内的活跃度加总计算得到（时间点与时间段计算方式均基于开发者个人活跃度方式计算）：

$$A_r = \sum (A_{u\_d}/day\_count)$$

开发者活跃仓库数量，则在此基础上定义为，由每个开发者所产生的活跃度大于 0 的仓库数量。

2.4 开发者时区估计方法

根据开发者全年每小时产生日志的数量，计算获取其事件最多的连续 12 个小时，令其时间为该开发者本地时间的 9 时至 21 时，则可以大致确定该开发者的所属时区。因为日志所记录的时间为 UTC 标准时间[1]。考虑开发者一天连续 12 个小时产生日志数最多的最后一个小时，其最后一小时（即本地时间 21 时）对应的 UTC 时间记为 k, 则时区计算公式如下：

$$zone = \begin{cases} 20 - k^*, & k^* > 8 \quad \text{对应本地为东时区} \\ -4 - k^*, & k^* \leq 8 \quad \text{对应本地为西时区} \end{cases}$$

其中，数学公式: $k^* = argmax\{\sum_{i=k-11}^{k} c_i\}, \quad k = 0, 1, ..., 23$，$c_i$ 表示当前小时产生的日志数量。当 i 为负值时，意味着为前一天的时间，需进行转换，即使用数学公式: $i = i + 24$ 进行转换。

因单个开发者的行为具有偶发性与特殊性，本方法不可用于单开发者时区的精确估计，但其在统计维度上具有意义，在此仅考虑统计学意义。再者，活跃度较低的开发者，其行为发生也具有偶发特性，所以在对开发者时区判断时，剔除了 GitHub Apps 并只保留全域活跃度排名前 5 万名的开发者账号用于分析，得到的工作时间画像如图 2-3 所示。

[1]UTC 即协调世界时间，是国际通用的时间标准，将全球各地的时间进行同步协调。UTC 时间是经过平均太阳时（以格林威治时间 GMT 为准）、地轴运动修正后的新时标以及以秒为单位的国际原子时进行综合精算而得到的。世界时区使用 UTC 的正或负偏移量表示。最西端的时区使用 UTC-12，比 UTC 落后十二小时；最东部的时区使用 UTC+14，比 UTC 早 14 小时。出现 UTC+13，UTC+14 的原因如下：

1） 因为国际日期变更线，虽然 1884 年划定时避免在一个国家中同时存在着两种日期，但是 1979 年成立的基里巴斯领土却跨越了国际日期变更线。基里巴斯的 UTC 有：莱恩群岛 (UTC+14)、菲尼克斯群岛 (UTC+13)、吉尔伯特群岛 (UTC+12)，这样保证一个国家内的日期为同一天。

2） 因为夏令时，即天亮早的夏季人为将时间调快一小时。比如新西兰是 UTC+12，在夏时制改用 UTC+13。

北京时间是中国采用国际时区东八时区的区时作为标准时间，东八区（UTC+8）是比 UTC 早 8 小时的时区。

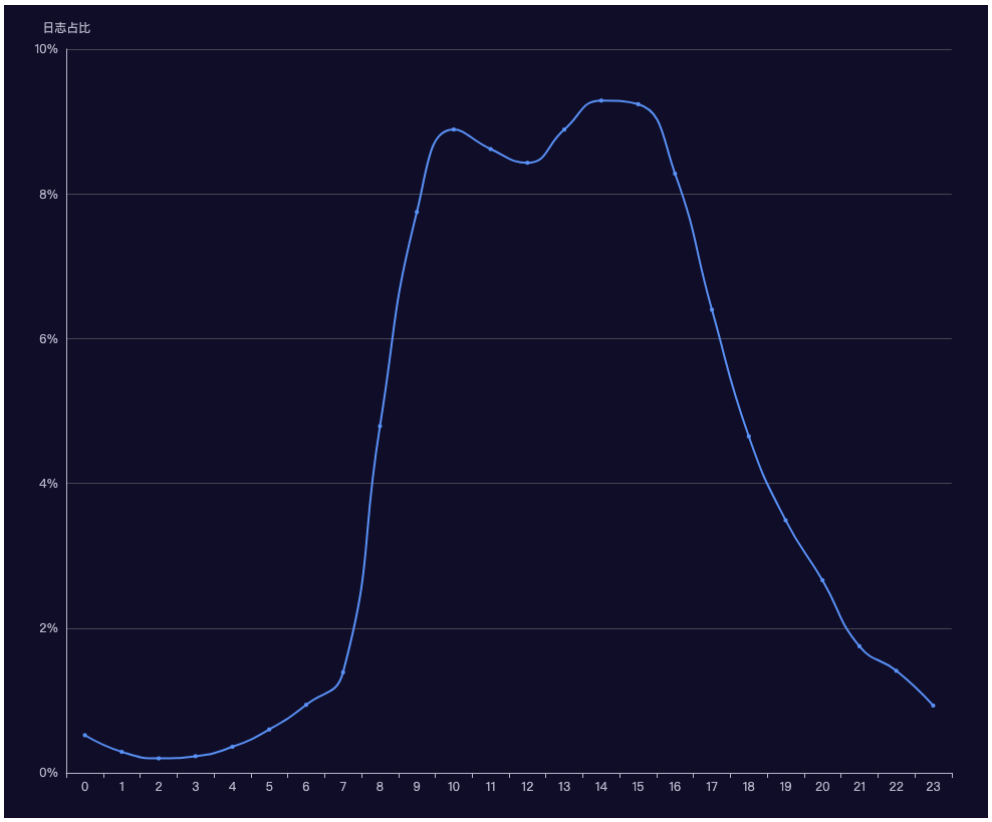

图 2-1: 典型开源开发者工作时间画像
Figure 2-1: Portrait of a typical open source developer's working hours

由图 2-3 所示，开发者通常会选择从当地时间早上 8 时开始工作，中午 11 - 13 时进行短暂的午休，随后在下午 15 - 16 时达到一个生产力高峰，并持续产出到晚间，较符合常见的工作时间情况；另一方面，可以看到开发者在晚间的工作产出比例还是较高的，甚至可以工作到凌晨 1 点，活跃程度应该明显高于一般职业。

在高活跃开发者中，UTC-8 至 UTC-3，即美洲（美国、加拿大、南美）开发者分布最多，虽然单时区的开发者比例不是最高，但总体开发者占比高达 33% 左右。而 UTC+1 至 UTC+3，即欧洲拥有最高的单时区开发者比例，UTC+1 时区高达近 10%，三个时区的总占比约为 26% 左右。总体而言，亚洲的开发者数量依然较少，但在 UTC+7 至 UTC+8 区域有一个小的波峰，说明中国、俄罗斯开发者相较其他国家还是有较高的开源活跃。而太平洋地区（UTC+9 至 UTC-9）则由于人口分布原因，开发者比例最低。

2.5 开发者使用语言统计方法

开发者使用语言定义为一个开发者账号在该年度使用频率最高的语言，其具体实现为一个开发者账号在该年度提交并合入 PR 最多的项目所使用的主要编程语言。

2.6 开发者协作网络构建方法

开发者协作网络是基于说明 2.1 中开发者活跃度的定义，通过多个开发者在项目中的协作关系进行构建的。具体构建过程为将每个开发者的全年活跃度计算精细化到具体

的 issue/PR 之上，则同时在同一个 issue/PR 上有过活跃的开发者认为其具有协作关系，而协作关系与两人在该 issue/PR 上的活跃度相关，具体计算方式为：

$$R_{ab} = \sum_i \frac{A_{ia} A_{ib}}{A_{ia} + A_{ib}}$$

其中 A_{ia}, A_{ib} 分别为开发者 a 和 b 在 issue/PR i 上的活跃度，计算方法遵循 2.1 中的开发者活跃度计算方法，R_{ab} 为开发者 a 和 b 在该项目上的协作关联度。即两个开发者在项目的协作关联度为其在所有共同活跃的 issue/PR 上的活跃度的调和平均值之和。

2.7 开源象限分析

提出一种开源象限（OpenQuadrant）的方法来刻画一个开源项目在影响力、全球化、社区规模三个核心特性方面的表现。基于该开源象限分析，使用散点图来表示，横纵两个维度为项目影响力指标和项目全球化指标，为了方便可视化，我们采用取对数的形式呈现上述两个指标，而使用散点图上的点的大小来刻画项目参与的活跃人数，用来反映一个项目的社区规模。[12]

以 2020 年 CNCF 基金会（Cloud Native Computing Foundation，即“云原生计算基金会”）下云原生领域的开源象限分析结果为例，如图 2-2 所示，

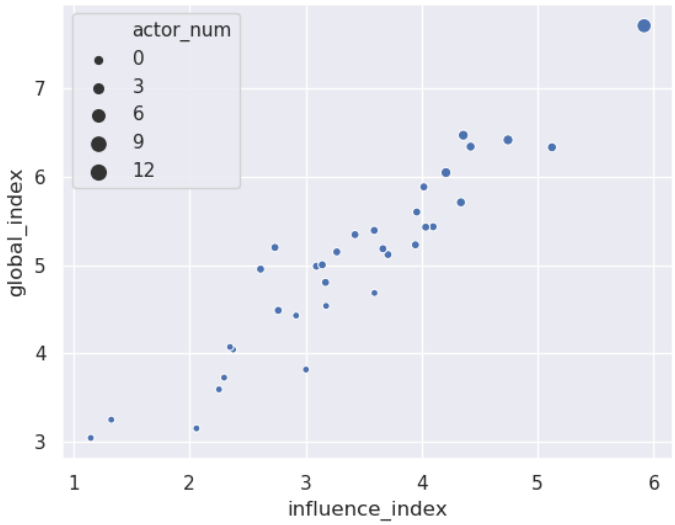

图 2-2: CNCF 下云原生领域的开源象限
Figure 2-2: Open source quadrant in the cloud native field under CNCF

基于以上，开源象限将整个平面分成了四块区域，分别是：

- 前瞻（Foresighted）：落在该区域的项目影响力强，同时项目全球化程度高；
- 引领（Leading）：落在该区域的项目影响力强，但项目全球化程度较低；
- 行动（Acting）：落在该区域的项目影响力较弱，但项目全球化程度高；

- 进入（Incubating）：落在该区域的项目影响力较弱、同时项目全球化程度也较低。

影响力、全球化、参与人数的具体计算方式如下所示。

2.7.1 领域项目影响力指标

领域项目的影响力计算结果是由全域项目的影响力计算结果给出，

2.7.2 领域项目全球化指标

全球化的影响因素众多，考虑地域、开发者人数这两个因素用于项目全球化指标的计算。针对地域因素，考虑计算参与该项目的开发者的时区分布情况，开发者时区分布趋于均匀分布，即标准差越小时，认为该项目的全球化程度越高。因此，首先通过判断项目上所有开发者的时区，之后计算了项目在 24 个时区上对应的开发者人数，在此基础上，计算项目关于时区人数的标准差。针对开发者人数因素，即该项目上参与的开发者总数，开发者人数越多，该项目全球化程度越高。考虑到开发者人数不多，但开发者的行为具有偶发特性的情况，该情况导致地域因素计算不准确，从而导致全球化指标的结果不准确。因此，使用如下全球化指标计算公式：

$$\sqrt{\frac{24}{\sum_{i=-12}^{11}(x_i-\bar{x})^2}}\bar{x}^2, \quad x_i\text{表示该项目在 i 时区的开发者人数。}$$

上述计算公式表明，全球化指标与项目时区开发者人数均值的平方成正比，与项目时区开发者人数的标准差成反比。该计算方式能够有效评估开发者人数较少的项目的全球化程度，另一方面，该计算方式对开发者人数较多的项目，全球化指标计算有正向作用。

2.8 项目参与人数

项目参与人数，即在项目上发生日志行为的参与者总数。在可视化散点图点的处理上, 我们对领域项目的参与人数使用线性最大最小归一化, 使得数值映射到了 1-10 区间范围内[13]。

3. 开源社区中的开放数据分析

3.1 数据的获取

在世界范围内，开源开发人员同时在贡献着数百万个项目，为项目编写代码和文档、修复 Bug、提交，而 GH Archive 就是其中一个起到了记录作用的项目。它会记录公共的 GitHub 时间线，即按时间顺序录入 GitHub 上所有行为活动的日志、存档，并使其能够被访问。本课题就使用了来自 GH Archive 的数据，开展进一步的研究。

开源平台 GitHub 提供了 20 多种事件类型，范围从新的 Commit 和 Fork 事件，到提交新的 Pull Request、Comment 以及向项目添加成员。这些事件被聚合到每小时的存档中，自动更新，能够通过 HTTP 的方式访问获取。

在本课题中，为了有效地进行开源项目或个人开发者的健康度衡量分析、呈现，在更新代价与数据有效性之间进行了折中选择，以 1 周为周期更新数据。

3.2 数据预处理

为了能够更全面的了解开源社区的数据特征，同时保证其对当下情况较好的描述作用，在此选择了开源平台 Github 的 2020 年度全域数据作为样本，进行预处理以待进一步探索分析。

在所有的数据中，由于 Github Apps 自动化协作机器人运行在服务端，可以同时服务于众多项目，从而具有了极高的活跃度和协作仓库数量，在后续统计开发者活跃度和活跃仓库数量时，对相关账号的协作行为进行了过滤。识别方法也很简单，即根据用户名后缀的 [bot] 进行账号识别。

3.3 开放数据分析

从总体数据来看，2020 年全年的 GitHub 全域事件日志数量总计约 8.6 亿条，较 2019 年 6.1 亿条增长约 42.6%，是近五年来增长最快的一年。经统计，2020 年 GitHub 全域活跃项目数量约 5,421 万个，活跃开发者账号约 1,454 万，分别较 2019 年增长 36.4% 与 21.8%。通过对全域开发者进行活跃度与活跃仓库数量的统计，可以得到 GitHub 全域开发者的活跃度分布情况和单个开发者活跃仓库数量分布情况如下：

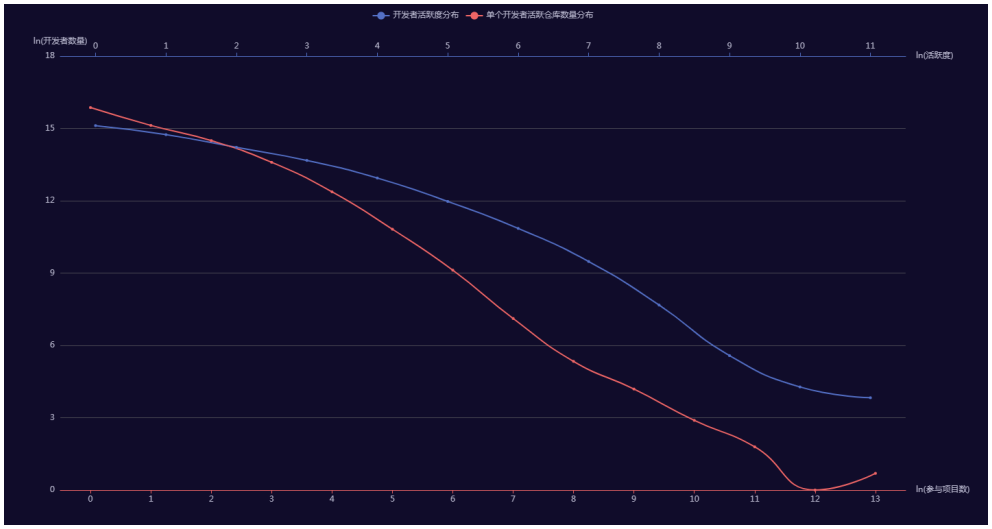

图 3-1: 开发者活跃度与活跃仓库数量分布图

Figure 3-1: Distribution diagram of developer activity and the number of active warehouses

图 2-1 使用双对数坐标系绘制，图上方的横坐标表示开发者活跃度，图下方横坐标表示开发者参与项目数量，纵坐标表示开发者数量，所谓双对数坐标，就是将原来的线

性坐标轴都取自然对数，可以看到开发者活跃度与活跃仓库数量的分布符合幂律分布。经统计，活跃度超过 2,000 的开发者数量为 5,445 个，占全域开发者数量不足万分之六。而大部分开发者活跃度都在 [0, 500] 区间内，占全域开发者数量的 99.45%，说明大多数开发者还是处于低活跃度的一个状态。观察曲线尾部，我们发现开发者活跃仓库数量在最后有一个回升，其实是由于部分未被过滤掉的自动化协作类账号的活跃仓库数量巨大，远超正常人类开发者，因此尾部形成 V 形曲线。

由于开源开发者在地理位置上遍布全球，他们具有不同的工作时间分布情况。通过对 Github 事件日志的详细时间戳数据可视化，可以看到在 UTC 标准时间下，全球的开源开发者工作时间分布具有明显的规律，如图 2-2 所示, 横轴为一天 24 个小时（UTC 标准时间），纵轴为一周 7 天，圆点大小表示该时段项目内产生的日志量的总和。

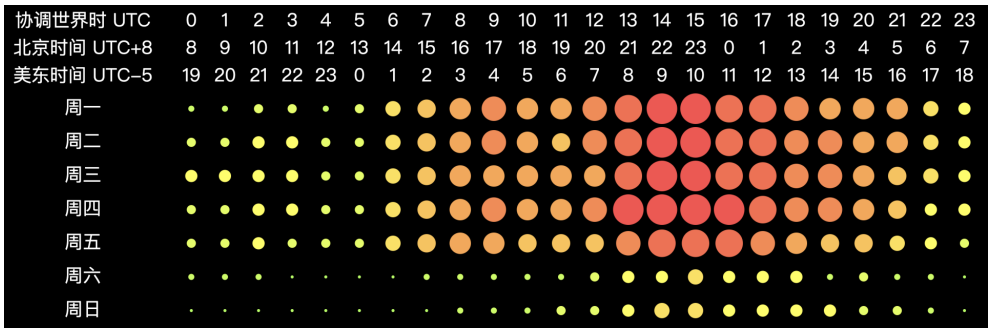

图 3-2: GitHub 2020 年全球日志时间分布情况
Figure 3-2: The global log time distribution of Github in 2020

如果从日志中抽取每个开发者独立的日志在每日不同时间段的分布情况，并通过对开发者进行时区估计后将其移动合并到同一时区，就能够得到开源开发者的典型工作时间分布情况，从而推断出开发者所属时区、地区，或是根据项目成员的协作时间分布，来判断一个项目的国际化程度。如图 2-3 所示，对项目内部所有开发者进行时区估计，随后以 UTC 标准时间下的 24 个时区为横轴，以每个时区开发者比例为纵轴，作分布直方图。

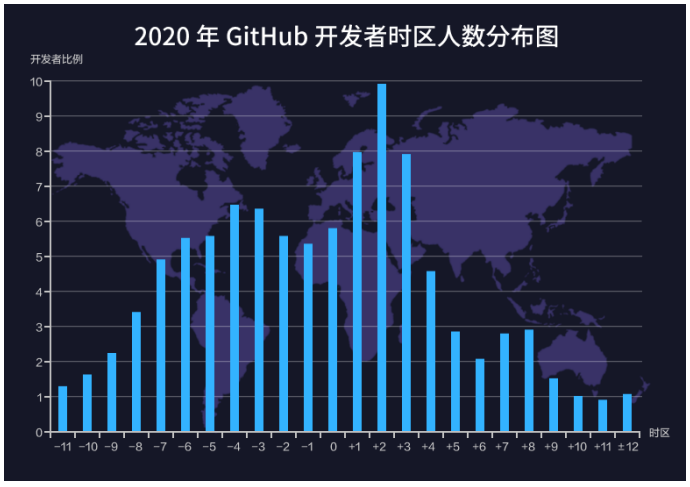

图 3-3: GitHub 2020 年全球开发者时区人数分布图
Figure 3-3: GitHub global developer time zone distribution map in 2020

开发者的地理分布情况一直是开源项目全球化指标的一个重要方面，然而一直以来没有较好的方式来估计开发者所在时区，但在全域日志数据下，开发者的个人行为为我们提供了有效的估计手段。可以在全域范围或特定开发者群体内估计开发者在不同时区的占比情况，从而有效判断 GitHub 全域或特定项目内的全球化覆盖情况。

另一方面，在不同的项目中，开源开发者们会使用不同的语言，也具有着一定的技术学习路线[14]，即主要使用的语言事件数量产生明显的增、减变化。在开源社区，由个体语言选择产生的变化组成了社区开发语言迭代的趋势，反之，开发语言的流行趋势也影响着开发者的学习路径。根据开源社区中活动频繁、贡献最大的一批用户的学习路径，就可以形成全域开发语言的流行趋势预测，从而为开发者的发展提供参考。

表 3-1: GitHub 2020 年全域活跃开发者和 Top10 万活跃开发者语言分布对比
Table 3-1: Comparison of the language distribution of GitHub's global active developers and the top 100,000 active developers in 2020

排名	全域开发者语言榜	全域范围 开发者账号数	Top 10 万开发者语言榜	Top 10 万范围 开发者账号数
1	JavaScript	305,814	JavaScript	15,858
2	Python	175,610	Python	10,866
3	HTML	159,303	TypeScript	7,419
4	Java	139,673	Java	6,665
5	Ruby	87,780	Go	5,094
6	TypeScript	85,116	C++	4,204
7	C#	54,343	Ruby	3,802
8	PHP	52,915	HTML	3,490
9	C++	47,799	PHP	3,000
10	CSS	46,428	C#	2,892

将 2020 年全域所有活跃开发者使用的语言分布和活跃度 Top 10 万的开发者使用的语言分布情况进行对比，可以看到存在着一定的区别，如表 3-1 所示。

图 3-4: 某项目的开发者协作网络图
Figure 3-4: The developer collaboration network diagram on a repository

除了个体之间的差异，不同的开发者之间也存在着协作关系[15]。在某一开源项目

中，来自不同开发者的贡献使他们构成协作网络，如图 3-4 所示，对于在该项目中作出贡献的开发者，取 top N 位（N 由用户定义配置）生成网络图，图中节点描述各开发者，节点大小描述开发者对该项目的贡献量，节点之间边的粗细描述两个开发者之间的协作活跃度[16]。

同一位开发者又有可能参与了多个项目，于是项目与项目之间也具有了关联性。综上两个方面考虑，优秀的开发者对项目具有着影响力，同时优秀项目更容易聚集优秀的开发者，这两者之间相互具有作用力。这样的关系可以用协作网络来进行可视化描述，通过项目协作关联度和社区发现算法对项目进行聚类，从而得到基于协作行为的聚类效果，并用于项目的大致分类。

根据连通性分析，在全域约 105 万个开源项目中，共分成了 44,990 个连通子图，其中 935,231 个项目在协作关系下构成一个巨大连通子图，其他 44,989 个连通子图中最大的子图包含 200 个项目，最小的包含 1 个项目，包含了 100 个项目以上的连通子图仅 9 个。这意味着全域活跃项目中 89% 的项目构成了一个巨大的协作网络，也就是 GitHub 开源世界的核心。而另外还有将近 4.5 万个小的协作孤岛，这些小的协作网络与开源世界不连通，意味着这些项目上的所有开发者都仅在自己的项目群中协作，从未在其他项目上有过活跃行为，同时在开源核心中的任意开发者也都没有在这个项目群中产生过协作行为[17]。

4. 面向开放组织的迷你看板设计

4.1 需求设计

当前，由于代码托管平台仅在项目或个人主页上呈现低阶的指标数据和有限的高阶指标数据，用户无法查看自定义的高阶指标数据，从而无法在较短时间内深入了解该项目或用户。面向开放组织的迷你看板最主要的目的与动机是直观清晰地展示项目高阶指标数据，并让这种可视化展现与项目本身无缝结合，充分发挥其助力能力，让用户能够轻松地把握各项项目“体征”，降低心智成本。在这一方面，对市面上已有的同类型看板应用进行调查，发现它们大都以独立的平台、网站的形式呈现，用户跨平台操作具有一定的门槛，且这样呈现的效果的直观程度大打折扣。

从另一个方面，作为 Hypertrons 的前端拓展，迷你看板可以改善 Hypertrons 的使用体验，补全 Hypertrons 在浏览器端的交互能力，为用户带来更加快速的开发、协作体验。比如，在 Hypertrons 项目中，存在一些场景下的交互指令（如 self-assign 等命令）需要手输，容易致使用户忘记或输入错误，但如果看板应用将手输代码指令这一个动作包装成为一个前端页面的按钮点击动作，那么用户就可以更快更好地达成操作。

4.2 总体设计

根据以上需求分析，面向开放组织的迷你看板其最主要的目的是辅助开源社区，并不希望深度改变用户的使用习惯，故在产品定位上，选择了浏览器插件的形式。该浏览

器插件将运行在网页端，通过检测当前浏览页面 url 的方式，为带有特定域名（目标开源平台域名）的网页插入迷你看板，并提供可以由用户自定义的图表设置选项，达到向开放组织提供有效的监控和运营手段的最终目标。

4.3 功能设计

4.3.1 高可配的数字化看板

数字化看板即在前端页面插入的可视化图表部分，也是该插件的最主要功能。它面向不同身份的用户，根据用户设置来展示定制的图表内容。当浏览器加载开源平台的页面时，插件将根据所加载页面对应的数据分析方法，提供用于分析的几个维度所对应的所有图表；而用户则在插件设置选项中选择性地开启或关闭特定图表，以达到定制化的目的。

特别地，在项目主页中，根据用户角色模型将用户分为 User、Contributor、Committer。在针对不同用户展示个性化的项目情况数字看板时[15]：a. 如果当前用户为 User，则展示该项目的基本信息，如 Star 数、Contributor 数、软件下载数等，帮助用户建立对整个项目的初步的整体认识。同时可通过一定的手段吸引并将该用户转化为贡献者，例如，列出标签为“good-first-issue”的 Issue 列表，引导用户为该项目做贡献。b. 如果当前用户为项目的 Contributor，则展示该名贡献者过去一段时间的贡献列表 (Issue 或 PR 等)，并显示该用户所贡献的 Issue 或 PR 的最新动态，同时鼓励用户持续贡献。c. 如果当前用户为项目的 Committer，则监控与项目健康度相关的一些指标，如 Issue 或 PR 的相应时间、项目贡献者及隶属组织多样性等，帮助 Committer 建立对项目现状及演化趋势的整体把握。d. 如果当前用户为项目的运营人员，则可以展示与项目运营活动相关的指标。

4.3.2 跨平台交互

跨平台交互在本项目中指: 来自不同系统、平台的用户，都可以在前端页面中实时地输入特定的指令来获取相应的支持，而不需要配置对应的本地环境，如：在浏览器页面键入/help 指令查看帮助，或使用受支持的/sendMsg 指令向其他平台发送自定义消息[18]。

4.3.3 跨平台实时消息通知

通过浏览器的消息通知机制，可以提供跨平台消息实时通知能力。例如，当项目发布新版本或者有其他新闻通知时，hypertrons 后端将消息推送给所有已安装的拓展程序，浏览器的消息服务将会唤醒用户安装的拓展程序，由拓展程序分析得到的消息数据，并创建通知，这样，不同身份的用户即可收到个性化的消息通知。

4.3.4 配置管理

用户可以在扩展程序的菜单栏中对配置进行修改，例如是否启用数字化看板、跨平台交互以及跨平台实时消息通知等功能。另外，可以考虑支持用户对本地页面的数字化看板进行配置，将这些配置存放于本地，覆盖默认的看板配置。

4.3.5 身份认证与权限管理

身份认证与权限管理是 Hypertrons 的核心功能之一。迷你项目看板可在浏览器前端页面中为用户提供身份认证入口，引导用户前往相关平台进行 Oauth 认证，并将用户的身份认证信息保存至 Hypertrons 后端，为用户行为数据分析与精细化运营提供基础。同时，为了支持用户进行跨平台交互操作，可支持用户在扩展程序中配置不同平台的 Token。

4.3.6 内容分级发放

迷你看板插件的安装与使用是开源并免费的，但对看板的可视化内容实行会员等级付费制度，即根据对数据的分析程度与内容信息量，将图表分为 Based、Detailed、Advanced 三个级别。其中 Basic 做引流，给用户展示所有组件的样例，含有最基本的数据信息；Detailed 则展示项目/开发者细节和基础数据聚合；Advanced 展示高级指标和深度分析结果。组件根据所属的等级区间标价，由用户定制化选择搭配，根据最终的组件组合收费。

5. 面向开放组织的迷你看板实现

本项目实现的是一个基于 Chromium 内核的浏览器插件,可支持一系列带有 Chromium 内核的单核或多核浏览器，如 Edge、Chrome、360 安全浏览器、360 极速浏览器等。

5.1 开发框架与语言

浏览器扩展程序基于 HTML、CSS 与 JavaScript 等前端技术构建。为了提升开发效率与代码可维护性，选择基于 React 框架进行开发。React 是一个开源的、声明式、高效且灵活的用于构建用户界面的 JavaScript 库，使用 React 可以方便的基于组件逻辑创建复杂的交互式 UI。

5.2 技术架构

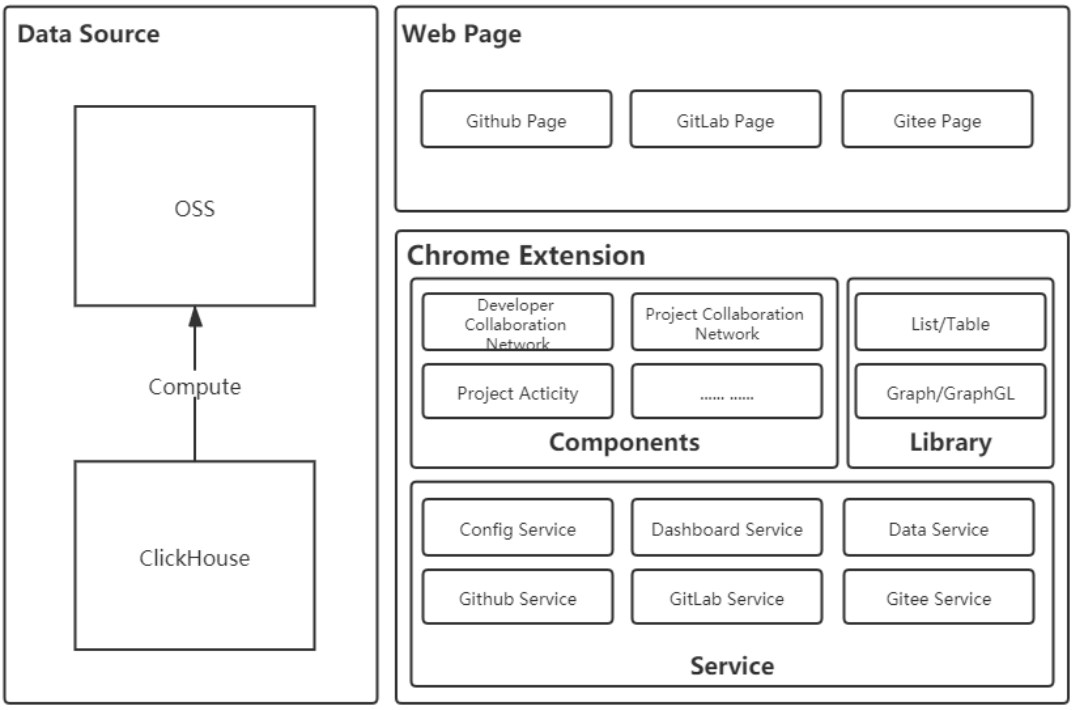

图 5-1: 技术架构图
Technical architecture diagram

面向开放社区的迷你看板的搭建过程本质上是一个浏览器插件的开发，从后端到前端，分为数据源、插件主体、前端页面三个部分。

在数据源部分，后端由 Hypertrons 收集的开放平台全域数据存储在 ClickHouse 数据库中，按照上文所述的定义与计算方法，计算得到具体的字段数值，作为静态对象存储到 OSS（Object Storage Service，阿里云对象存储）服务中，供前端直接调用。

在插件主体部分，使用 Typescript 类定义各可视化图表组件，作为可以调用的模板，提供接口以嵌入真实的数据。同时，在该部分构建了配置服务、仪表盘服务，并联通后端的数据服务和 3 个开放平台（Github、GitLab 和 Gitee）的服务接口。其中，配置服务负责维护一个存储用户配置信息的 json 文件，每一次向后端发起请求、渲染图表之前，都去读取；仪表盘服务则负责取得数据、嵌入可视化组件模板，生成图表以插入到前端页面。

在前端页面部分，则根据 Github、GitLab、Gitee 三个开放平台的 UI 风格，选择了与原生样式相近的组件库 FrontUI，在原网页布局的基础上，将仪表盘服务生成的图表插入到页面的特定位置，如边栏、文本间等。

在这个基础上，考虑到开发便利性与项目的国际化程度，在功能实现、架构搭建的基础上，还使用了 chrome 的 i18n 架构以增添插件对不同语言的支持，使用了静态代码分析工具 ESLint、代码格式化工具 Prettier 和同为开源项目的工具 Husky 来使代码规范化。这些工作是为了保障本项目作为开源项目的开放性。

5.3 高可配能力的实现

项目所有的配置都从代码仓库中的配置文件.github/hypertrons.json 中读取，即：插件所提供的功能由配置文件决定，配置文件配置在项目中，基于 Git 进行管理和迭代，公开透明。同一项目下的所有安装该插件的用户，Perceptor 提供的是相同的能力。同一用户，在不同项目页面下，看到的是不同的看板。

5.4 数字化看板模块的实现

项目基于 Echarts 提供数据可视化图表渲染能力。Echarts 是一个使用 JavaScript 实现的开源可视化库，提供直观、交互丰富、可高度个性化定制的数据可视化图表。通过对 Echarts 进行封装，并基于 Hypertrons 后端获取数字化看板的配置信息以及各个平台的开放数据，就可以根据每个社区/组织的自身业务场景，实现高度可配的数字化看板。基于此，能够实现支持每一位用户基于自身权限地配置浏览器页面的数字化看板[19]。

以 GitHub 项目数字化看板展示为例，可从 hypertrons 后端获取得到数字化看板的配置信息，以及项目的 Repository、Issue、Pull request 和从其他平台采集得到的开放数据，然后根据用户自定义的配置信息生成相应的可视化组件，再经由浏览器插件，以 HTML 的格式注入到用户浏览的当前页面，完成整个呈现[20]。

5.5 跨平台交互模块的实现

在浏览器页面中输入的交互式命令行会由浏览器插件传递到 hypertrons 后端，由 hypertrons 后端运行、完成指定操作后后，再通过浏览器插件，反馈结果给来自不同平台、使用不同语言的用户，从而提供跨平台交互的能力。

5.6 跨平台实时消息通知与配置管理模块的实现

跨平台实时消息通知与配置管理模块是基于浏览器插件的特性，在 Chromium 提供的用于浏览器插件开发的官方接口上进行实现。其本质是前后端之间信息的简单传递。

5.7 身份认证与权限管理模块的实现

用户能够在开源平台的设置项中管理自己的 Token，而当用户为迷你看板插件配置了个人账户的 Token，即迷你看板插件获得了该用户账号的部分权限。获得用户 Token 之后，插件将读取的用户名和头像自动显示，并且该 Token 会在后续发生的一系列请求访问中自动使用。

5.8 功能测试

项目使用 Jest，一个简洁的测试框架，它能够很好地与 Typescript 以及 React 框架共用。Jest 在大部分基于 JavaScript 的项目上可以实现开箱即用、无需配置，且具有优秀

的 api，易于书写与维护。

6. 总结与展望

6.1 总结

本课题针对开放组织，结合机器人流程自动化，实现一个基于浏览器插件的迷你看板，采用了浏览器插件的形式，通过在特定网址的网页中插入可视化组件并提供交互手段，协助开源社区中不同项目、不同视角的决策与运营，进一步完善了开源社区中的数字化治理框架，可以帮助开放组织或开源社区更高效的运作和管理。

论文首先阐述了项目的研究背景与研究动机，主要介绍了开源社区与开放组织的的运作模式与发展方向，随之提出了本文主要面向的问题与需求。接着，提出明确的方法定义，去对开放平台的数据进行探索分析，挖掘其隐藏并有价值的数据信息，并形成具体的的可视化分析的方向。明确了研究方向与目的以后，再从产品与业务的角度出发，剖析需求、流程、业务逻辑，完成迷你看板的总体定位与细节功能设计。最后，针对每一个开发环节与功能模块，调研技术方法、确定技术方法，并逐个实现。

6.2 展望

开放式组织中包含了人员招聘、项目管理、人力绩效、工资分红、退出机制、宣传营销等各个环节，下一步可考虑在本文实现的迷你看板的基础上，添加人员的进入与退出机制、工资分红机制等，从而完善开放组织或开源社区的治理框架，帮助开放组织完成宣传、活动、用户、开发、管理、布道等各个方面进行数字化管理和常态化。

经由这个项目，在开源社区中可视化深层信息、辅助决策的目的已经基本能够达到。这个迷你看板以浏览器插件的形式实现，在完善开源社区机制、推动生产力关系变革的进程中迈进了有意义的一步，在这个项目的基础上，已然可以展望开源供应链的迭代、区块链等新技术的加入，但这些也绝不会是终点。开源的进程在当前的基础上将不断被推动，并将最终演化出新的组织关系和技术产物，相互促进，往复循环。

参考文献

[1] 王璐. 开放式创新的协同演化机制研究: 创始人社会资本, 创新团队与创新绩效的交互影响[D]. 武汉大学, 2017.

[2] FUCHS C. Wikinomics: How mass collaboration changes everything[J]. International Journal of Communication, 2008, 2: 1–11.

[3] 李其锋. 面向开源社区的开发者群体行为分析方法[D]. 武汉大学, 2014.

[4] [1 万, 李德杰. 2020 自由开源软件发展蓝图综述[J]. 计算机应用研究, 2009, 26(11).

[5] MARTÍNEZ-TORRES M R, DÍAZ-FERNÁNDEZ M C. Current issues and research trends on open-source software communities[J]. Technology Analysis & Strategic Management, 2014, 26(1): 55–68.

[6] 王怀民, 尹刚, 谢冰, 等. 基于网络的可信软件大规模协同开发与演化[J]. 中国科学: 信息科学, 2014, 44(1): 1–19.

[7] 李存燕, 洪玫. Github 中开发人员的行为特征分析[J]. 计算机科学, 2019, 46(2): 152–158.

[8] 杨波, 于茜, 张伟, 等. GitHub 开源软件开发过程中影响因素的相关性分析[J]. 软件学报, 2017, 6.

[9] KALLIAMVAKOU E, GOUSIOS G, BLINCOE K, et al. The promises and perils of mining github[C]// Proceedings of the 11th working conference on mining software repositories. [S.l.]: [s.n.], 2014: 92–101.

[10] 王伟, 周添一, 赵生宇, 等. 全球开源生态发展现状研究[J]. 信息通信技术与政策, 2020(5): 38–44.

[11] TSAY J, DABBISH L, HERBSLEB J. Influence of social and technical factors for evaluating contribution in GitHub[C]// Proceedings of the 36th international conference on Software engineering. [S.l.]: [s.n.], 2014: 356–366.

[12] VASILESCU B, YU Y, WANG H, et al. Quality and productivity outcomes relating to continuous integration in GitHub[C]// Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. [S.l.]: [s.n.], 2015: 805–816.

[13] MCDONALD N, GOGGINS S. Performance and participation in open source software on github[G]// CHI'13 Extended Abstracts on Human Factors in Computing Systems. [S.l.]: [s.n.], 2013: 139–144.

[14] DABBISH L, STUART C, TSAY J, et al. Social coding in GitHub: transparency and collaboration in an open software repository[C]// Proceedings of the ACM 2012 conference on computer supported cooperative work. [S.l.]: [s.n.], 2012: 1277–1286.

[15] 廖志芳, 李斯江, 贺大禹, 等. GitHub 开源软件开发过程中关键用户行为分析[J]. 小型微型计算机系统, 2019, 40(1): 164–168.

[16] THUNG F, BISSYANDE T F, LO D, et al. Network structure of social coding in github[C]//2013 17th European conference on software maintenance and reengineering. IEEE. [S.l.]: [s.n.], 2013: 323–326.

[17] 刘玉辉. GitHub 开源软件项目团队协作过程监控与评价[D]. 哈尔滨工业大学, 2018.

[18] 朱亚琼. 可视化驱动的交互式数据挖掘方法研究[J]. 电脑知识与技术, 2016, 36.

[19] DEQINGLI H, YISHEN S, WENLIZHANG J, et al. ECharts: 是一款开源的, 基于 web 的, 跨平台的支持快速创建交互式可视化的框架[J]. Visual Informatics, 2018, 2(2): 136–146.

[20] 陈挺, 王小梅, 吕伟民. Ng-info-chart: 基于自定义 HTML 标签的交互式可视化组件[J]. 现代图书情报技术, 2016, 6: 88–95.

附录

1 部分代码

- 定义 PercepterBase 类做一些简单的工作如 logger、场景判断。

```
export default abstract class PerceptorBase {
    public logger: any;

    constructor() {
    this.logger = {
        info: (message?: any, ...optionalParams: any[]):
        void => {
        if (process.env.NODE_ENV !== 'production') {
            console.log('i Perceptor : ',
             message, ...optionalParams);
        }
        },
        error: (message?: any, ...optionalParams: any[]):
        void => {
        console.error('□ Error Message From Perceptor :',
         message, ...optionalParams);
        // TODO: pass error message to ErrorMessageBar component
        render(<ErrorMessageBar />,
        document.getElementById('perceptor'))
        }
    };
    }

    public abstract run(): Promise<void>;
}
```

- 定义 Perceptor 类继承 PerceptorBase，来检查 settings、创建 div、取配置文件、渲染每一个组件。

```
export class Perceptor extends PerceptorBase {
    public static Features: Map<string, any> = new Map();
    public settings: any;

    public async run(): Promise<void> {
    // wait until <body> element is ready
    await elementReady('body', { waitForChildren: false });
    this.logger.info('body element is ready.');

    this.logger.info('creating perceptor div ...');
    const perceptorDiv = document.createElement('div');
    perceptorDiv.id = 'perceptor';
    $('#js-repo-pjax-container').prepend(perceptorDiv);

    await this.checkSettings();

    // run every features
    Perceptor.Features.forEach(async (Feature, name) => {
        const featureId = name.replace(name[0],
        name[0].toLowerCase());
        this.logger.info('trying to load ', featureId)
        if (this.settings.toJson()[featureId] === false) {
        this.logger.info(featureId, 'is disabled');
        return;
        }
        if (Feature.prototype.include
        .every((c: () => any) => !c())) {
        this.logger.info(featureId,
         'does NOT run on this page')
        return;
```

```
        }
        try {
        this.logger.info('running ', featureId)
        const feature = new Feature();
        await feature.run();
        } catch (error: unknown) {
        this.logger.error(featureId, error)
        }
    }, this)

    private async checkSettings(): Promise<void> {
        this.logger.info('loading settings ...');
        if (isRepo()) {
            this.logger.info('Detected that this is a repo page,
             trying to load configuration file from the repo ...');

            const owner = utils.getRepositoryInfo(window.location)!
            .owner;
            const repo = utils.getRepositoryInfo(window.location)!
            .name;
            const configFromGithub = await getConfigFromGithub(owner, r
            this.logger.info('The configurations are: ', configFromGith
            this.settings = await mergeSettings(configFromGithub);
        } else {
            this.settings = await loadSettings();
        }
    }
}
```

- 渲染组件，以开发者协作网络为例：

```
public async run(): Promise<void> {
```

```
const pinnedReposDiv = $('.js-pinned-items-reorder-container')
.parent();
const DeveloperNetworkDiv = document.createElement('div');
DeveloperNetworkDiv.id = 'developer-network';
DeveloperNetworkDiv.style.width = "100%";
this._currentDeveloper = $('.p-nickname.vcard-username.d-block')
.text().trim();
const settings=await loadSettings();
try {
const forceGraphDataRaw = await getGraphData(`/actor/${this._curren
await this.generateForceGraphData(forceGraphDataRaw);

const circularGraphDataRaw = await getGraphData(
    `/actor/${this._currentDeveloper}_top.json`);
await this.generateCircularGraphData(circularGraphDataRaw);

const developerColumns = [
    {
    key: 'column1',
    name: getMessageI18n('global_developer'),
    fieldName: 'name',
    minWidth: 100,
    maxWidth: 200,
    isResizable: true,
    onRender: (item: any) => (
        <Link href={'https://github.com/' + item.name} >
        {item.name}
        </Link>
    ),
    },
```

```
    { key: 'column2', name: getMessageI18n('global_correlation'), f
    { key: 'column3', name: getMessageI18n('global_activity'), field
];
const repoColumns = [
    {
    key: 'column1',
    name: getMessageI18n('global_repo'),
    fieldName: 'name',
    minWidth: 100,
    maxWidth: 200,
    isResizable: true,
    onRender: (item: any) => (
        <Link href={'https://github.com/' + item.name} >
        {item.name}
        </Link>
    ),
    },
    { key: 'column2', name:
     getMessageI18n('global_contribution'), fieldName: 'value', min
];
render(
    <div>
    < GraphWithList
        layout='force'
        graphType={settings.graphType}
        title={getMessageI18n('component_developerCollabrationNetw
        graphData={this._forceGraphData}
        graphDataGraphin={this._forceGraphDataGraphin}
        columns={developerColumns}
        listData={this._developerListData}
```

```
        />
        < GraphWithList
            layout='circular'
            graphType={settings.graphType}
            title={getMessageI18n('component_mostParticipatedProjects_t
            graphData={this._circularGraphData}
            graphDataGraphin={this._circularGraphDataGraphin}
            columns={repoColumns}
            listData={this._repoListData}
        />
        </div>,
        DeveloperNetworkDiv,
    );
    pinnedReposDiv.before(DeveloperNetworkDiv);
    } catch (error) {
    this.logger.error('DeveloperNetwork', error);
    return;
    }
}
```

致谢

在华东师范大学数据科学与工程学院学习的这三年时间，给我留下了深刻的印象。我得到了许多老师和同学们的帮助和支持，值此论文完成之际，特向他们表示由衷的感谢。

我还想感谢我的论文指导老师王伟老师和 X-lab 的各位学长，他们在我撰写论文期间给予了我许多指导与帮助，拓宽我的思维与认知，最终能够形成完整的思路。同时，与他们的交流不只是带给了我学术上的收获，更多的是对开源、对社会生产关系的思索。正如这 3 年间每一次与王伟老师交谈时的所获所感，我相信，正是这些技术背后的东西，才能够启迪我、使我在未来走得更远。