**摘要**

软件开发的过程中产生了大量的软件开发过程数据，这些软件开发过程数据记录了软件从诞生开始的成长与发展轨迹。这些数据不仅能使项目的管理者与开发者了解到目前项目的具体开发情况，从而做出项目决策，还能帮助人们回答许多软件工程学的问题。因此，对于软件开发过程数据的研究与理解具有重要的意义。而要对这些软件开发过程数据进行理解与使用，则首先要建立起一套基础设施，帮助收集、存储与检索软件开发过程数据。但目前这方面的工作还存在很多不足。

本文以北京大学软件工程研究所数据分析小组的数据为基础，设计与实现一套软件开发过程数据的检索工具。本文提出的工具有以下几个方面：第一，从数据的处理脚本中提取出脚本的描述信息，包括脚本的处理对象，用途，使用方法等；第二，从软件开发过程数据中提取出数据的描述信息，描述数据的来源，基本信息与存储位置等；第三，通过数据与脚本的描述信息，建立检索工具。可以让用户方便快速的检索到所需的数据，并进一步对数据进行提取与处理，从而提高软件开发过程数据的使用效率。

**Abstract**

The software development data is an important data to show how software evolves, because it records all the events from the born of software to its end. This data can not only help project managers understand the project’s process and make decisions, but also help researchers solve lots of research questions about software engineering. Therefore,

# 引言

软件开发的过程中产生了大量的软件开发过程数据，包括项目的源代码，缺陷报告以及项目开发者们之间的通信邮件等，这些软件开发过程数据记录了软件从诞生开始的成长与发展轨迹。对这些数据进行研究，理解与使用可以帮助人们回答很多软件工程学的问题，例子如下。

开源社区的生存与发展离不开贡献者的参与，如何吸引与维持贡献者就成为开源社区非常关注的一个问题。Zhou和Mockus通过对Mozilla和Gnome两大社区的缺陷数据进行收集、分析与研究，建立模型解释了影响开源社区参与者成为长期贡献者的原因[英文 1]。Mockus针对Apache和Mozilla两个社区的代码变迁数据进行分析与研究，回答了他所提出的六个假设[英文 2]，这篇文章也得出了许多有意义的结论，他的这些发现也被大量的软件工程研究者引用与再研究。Mockus还发现，收集与分析整个开源代码库，能够帮助解决许多软件工程的基础问题，包括代码演化等[英文 3]。

上述例子都是通过收集、分析与使用软件开发过程数据，总结出项目开发的最佳实践，从而指导软件项目的开发。因此，为了帮助人们更好地理解与使用软件开发过程数据，目前有一些工作已经建立起统一的，普遍的软件开发过程数据资源库，为人们收集、存储与分析这些数据提供了便利。例如，OpenHub[[1]](#footnote-1)网站建立了一个开源社区的目录网站，收集、统计并展示了大量开源社区的基本信息，为人们了解开源社区提供了帮助。GHTorrent[[2]](#footnote-2)通过利用GitHub API，向用户提供了对GitHub[[3]](#footnote-3)上的项目数据的定制功能[英文 4]，以方便人们使用GitHub上的数据。北京大学软件工程研究所数据分析小组也建立了软件开发过程数据池,收集与存储了来自多个开源社区的软件开发过程数据[中文 1]，然而，为了有效地利用这些数据，需要提供一套检索软件开发过程数据的工具，帮助用户快速的检索软件开发过程数据，并进一步对它们进行研究，理解与使用，从而解决许多软件开发实践问题。

本文以北京大学软件工程研究所数据分析小组的数据为基础（包括从互联网收集的源代码，缺陷数据，邮件数据等软件开发过程数据以及相应的数据处理脚本），尝试设计与实现一套软件开发过程数据的检索工具，帮助用户检索软件开发过程数据。

本文结构安排如下：第一章是引言；第二章介绍相关技术；第三章给出工具的设计；第四章介绍工具的具体实现；第五章介绍本文的工作进展和对未来工作的展望。

# 相关技术

## 软件开发过程数据

本文所述的软件开发过程数据就是指在软件开发过程中出现的用于辅助软件开发的数据。本文针对的软件开发过程数据包括三种类型：代码变迁数据，缺陷数据和邮件数据。

### 代码变迁数据

代码是软件项目的重要组成部分，它也是最重要的软件开发过程数据。代码一般被存储于版本控制系统中，以方便管理和协同开发。常见的版本控制系统有SVN, GIT, Bazaar等，这些系统记录了项目中所有的代码历史以及每一次的代码提交信息。本文所面向的代码变迁数据就是指项目开发过程中产生的代码提交日志，标记着项目中的每一次代码提交。这些数据通常包含如下信息：代码提交者，提交时间，本次提交所变更的文件信息以及对本次提交的注释等。代码变迁数据的格式会由于所使用的版本控制系统的不同而不同，获取它们的方法也不尽相同

代码变迁数据可以让我们了解到项目中的代码是如何变化的，我们可以直接从这些数据中得到许多信息，包括整个项目的提交历史，项目的时间跨度，人员分布。同时，我们还可以通过分析这些数据，得到更细节的信息，包括项目代码是如何演化的等。因此，分析与理解这些信息，能让我们从宏观上和微观上了解项目开发的整个过程。

### 缺陷数据

缺陷数据记录了人们在软件开发过程中发现的软件项目存在的错误，或者对软件项目提出的新需求。缺陷数据一般被存储于缺陷追踪系统中，缺陷追踪系统为用户提供一套报告，追踪，讨论缺陷的方法。常见的缺陷追踪系统有Bugzilla，Jira等。本文所面向的缺陷数据就是存储在缺陷追踪系统中的缺陷报告，这些数据通常包括两个方面：缺陷报告的基本信息和活动信息。其中，缺陷报告的基本信息是描述这个报告的整体情况的一些信息，通常包含如下信息：报告时间(reported)，状态(status)，重要程度(Importance)，所属产品(product)，负责人(assigned to)，评论(comments)，附件(attachments)等。缺陷报告的活动信息则是描述这个报告的历史状态变动情况，通常包含如下信息：参与人(who)，时间(when)，变更内容(what added, what removed)等。缺陷数据的格式会由于使用的缺陷追踪系统的不同而不同，同时，即使使用相同的缺陷追踪系统，不同的项目之间的缺陷数据的格式也不尽相同。

缺陷数据可以帮助我们了解项目中的缺陷报告是如何产生，如何分发，以及如何被追踪的等等。我们可以通过观察用户在缺陷追踪系统中进行操作与交互时留下的数据，来了解项目的发展历程。

### 邮件数据

在软件项目开发过程中，项目的开发者往往会通过邮件进行交流，邮件数据则记录了这些开发者之间的日常交流活动。项目开发的过程中，开发者往往会建立多个邮件列表，来讨论项目开发过程中遇到的不同方面的问题。这些邮件数据一般存在邮件管理系统中，如mailman等。邮件系统中的邮件数据通常包括如下信息：发送者，接收者，发送时间，邮件内容等。与代码变迁数据和缺陷数据相同，邮件数据的格式也会因所使用的邮件管理系统的不同而发生改变。

邮件是项目的开发者之间交流的主要方式，邮件数据能够反映出开发者之间交流和协作的关系。邮件数据可以让我们了解开发者之间的交流模式，对于分析项目开发者如何协同对项目进行开发有重要的意义。

## MongoDB

MongoDB是一种基于文件存储的非关系型数据库，它与传统的关系型数据库相比，具有可移植，可扩展，高性能，存取方便等特点。

MongoDB中的数据的存储模型从上到下分别为：数据库，集合，文档。MongoDB是面向集合进行存储的，并且模式自由。面向集合是指数据被存储在集合中，这里的集合类似于关系型数据库中的表。而模式自由则意味着数据存储在MongoDB中时，不需要定义任何模式，我们可以把任意结构的数据存入其中。存储在集合中的是文档，存储的形式是键－值对，键用于标识一个文档，为字符串类型，而值可以是任意复杂的文件类型。

本文使用MongoDB对数据进行存储，有以下几点原因：

1. MongoDB是一种非关系型数据库，不要求所存储的数据具有特定的结构，存储模式非常灵活。相对而言，关系型数据库则要求数据有确定的结构，并可以支持如表连接等复杂的数据库操作。对于本文所需要存储的数据而言，软件开发过程数据的描述信息之间差异很大，例如需要描述项目邮件数据的时间分布，不同的项目时间跨度不同，有的跨越的时间长，有的跨越的时间短，不方便使用特定的结构进行存储。同时，除了软件开发过程数据本身，本文还要存储处理这些数据的脚本的描述信息。除此之外，本文所存储的数据之间不需要很复杂的表连接操作，所以MongoDB非常适合用于存储本文所需存储的数据的描述信息。
2. MongoDB很好的实现了面向对象的思想，在MongoDB中，每一条记录都是一个文档对象，这使得我们在使用编程时，可以很方便的与数据库进行交互，同时它还具有多种语言配套的API，为我们操作数据库提供了很好的帮助。
3. MongoDB中的数据是使用JSON格式进行存储的。而本文的数据展示部分，其中的图形化界面使用web UI的形式进行展示。在前端与后端之间使用RESTful API进行交互，所传输的数据类型就是JSON，所以使用MongoDB可以为数据存储与传输带来极大便利。

综上所述，本文使用MongoDB作为数据的存储系统。

同时，本文的工具使用python来进行实现。Python是一种面向对象的解释型语言，具有良好的跨平台性。它拥有强大的开源工具库，其中pymongo就是MongoDB的一套python API，封装了MongoDB的基本操作。pymongo提供在python中操作MongoDB的一系列方法，这些方法包括以下部分：

1、连接数据库，包括建立连接，选取数据库，选取集合等。

2、操作数据库，包括数据的插入、查询、修改、删除。

# 工具的设计

本章首先介绍工具的总体设计，然后再分别介绍工具各个模块的设计。

## 工具的总体设计

工具按照功能划分，包括三个部分：数据提取，数据存储，数据检索。工具的整体结构如图所示：

* 数据提取模块：

用于提取软件开发过程数据及相关处理脚本的描述信息。

* 数据存储模块：

用于将所提取的描述信息存储到数据库中。

* 数据检索模块：

用于通过所提取的软件开发过程数据的描述信息来对软件开发过程数据进行检索。

## 数据提取模块设计

数据提取模块是用于提取软件开发过程数据及相关处理脚本的描述信息，按照提取的需求可以分为三个小的模块：日志数据描述，缺陷数据描述，邮件数据描述，下文将针对上述几个小的模块详细描述数据提取模块的设计。

### 日志数据描述

本模块的输入是项目的日志数据，记录了项目代码的每一次提交信息。本模块的输出是日志的描述信息，用于描述每个项目的日志数据的基本信息，并作为查询日志数据的基本维度。日志数据描述模块具体的流程与方法设计如下：

1、在服务器上定位代码变迁数据。本文的定位已经在数据抓取的阶段做好，对代码变迁数据抓去时，已有一个文件专门记录了项目的名字，版本控制系统以及项目的代码变迁数据在服务器上存放的位置。

2、从版本控制系统中提取日志。这一步的通过遍历上一步的定位文件里的信息，得到每一个项目的版本控制系统，根据不同的版本控制系统，使用不同的命令来打印每一个项目的日志。

3、从项目日志中提取描述信息。这一步的输入是2中提取的日志信息，根据每个项目所属的版本控制系统的不同设计相对应的提取脚本，提取出描述信息。

### 缺陷数据描述

本模块的输入是项目的缺陷报告数据，记录了项目中缺陷的基本信息以及历史状态变更。本模块的输出是缺陷数据的描述信息，用于描述每个项目的缺陷报告的基本信息，并作为查询缺陷数据的基本维度。缺陷数据的格式随着不同的缺陷追踪系统而改变，同时，使用相同缺陷追踪系统的不同的社区的缺陷数据也会存在格式上的不同。因此，在提取描述信息时需要针对不同的数据格式来实现不同的提取方式。缺陷数据描述模块具体的流程与方法设计如下：

1、在服务器上定位缺陷追踪数据。本文使用后缀名来定位缺陷追踪数据，抓取缺陷数据时，原始数据的后缀名是.level0，使用linux的find命令查找\*.level0的文件可以找到缺陷数据。同时，还有一部分数据是抓取时未合并成\*.level0的数据，存放在bug目录底下，将在下一步中进行合并。

2、合并分散的缺陷数据。这一步将多个文件按照文件名的规则合成一个大文件。文件名是用数字进行编号的，从1到n，按照从小到大的顺序将文件顺序合成一个大文件，同样以\*.level0来命名，方便下一步处理。

3、从缺陷数据中提取描述信息。这一步的输入是1中的level0数据以及2中合并过后的数据。本文根据不同的数据格式，设计不同的提取脚本，提取出描述信息。

### 邮件数据描述

本模块的输入是项目的邮件数据，记录了项目中开发者之间的邮件交互。本模块的输出是邮件数据的描述信息，用于描述每个项目的邮件数据的基本信息，并作为查询邮件数据的基本维度。邮件数据的格式随着不同的邮件系统而改变，在提取邮件数据时需要针对不同的数据格式来实现不同的提取方式。邮件数据描述模块具体的流程与方法设计如下：

1、在服务器上定位邮件数据。同样是根据后缀名进行定位，这里不再赘述。

2、从邮件数据中提取描述信息。根据不同的数据格式设计不同的脚本，这里要注意的是存储邮件数据的文件有可能是压缩后的文件（\*.gz），需要先进行解压再进行分析，并提取出描述信息。

## 数据存储模块设计

数据存储模块用于将数据提取模块提取的描述信息存储到MongoDB中。这个模块起到了承上启下的作用，一方面将提取的数据插入到数据库，对数据进行了一层抽象，另一方面将数据整合，为之后的检索提供媒介。下文将分几个部分来介绍数据存储模块的设计，包括数据集合设计，数据批量插入和数据修改与删除。

### 数据集合设计

MongoDB中的数据以集合来存储，每个集合就类似于关系型数据库里的表。本文要存储数据提取模块提取的描述信息，首先需要对数据集合进行设计以安排合理的数据存储模式。数据集合设计包括以下部分：

* 集合分类。集合按照所存的数据类型分成四个大类，分别是代码变迁数据，缺陷数据，邮件数据和脚本；同时，每个大类又按照数据存储系统的不同分成若干个小类。集合分类设计情况如下表所示：
* 集合内文档结构设计。集合内的文档结构按照所提取的描述信息进行安排，键为描述信息的名字，值为描述信息的值，每一个项目或者社区的信息。

### 数据批量插入

本文在数据提取阶段，提取出的描述信息都以文件的形式存储，在存储阶段要做的就是将这些文件中的数据批量插入到MongoDB中。上一步以及设计好了数据集合，这一步就按照所设计的模式将数据导入数据库。

数据批量插入需要解决的问题同样是针对不同的数据格式设计不同的脚本来进行数据的插入。本文在数据提取阶段，已经将各个类型的数据所提取出的描述信息的格式进行了统一，在存储阶段只需要针对三类不同的软件开发过程数据以及脚本数据分别设计数据插入的脚本即可。数据批量插入的具体流程和方法设计如下：

1、连接数据库。利用python的pymongo包对MongoDB进行连接，程序进入到数据库中准备进行数据插入操作。

2、确认数据类型与数据格式。首先根据要插入的数据来确定数据的类型，从而选取相对应的集合。然后根据数据的存储系统来确定数据的格式，从而确定数据的描述信息所包含的域有哪些，为下一步做准备。

3、批量插入数据到数据库。文件里每一行数据都是一个项目或者一个社区的描述信息，一行里有多个域，每个域对应一项描述信息。当插入数据到数据库中时，读取文件每一行的数据，每个域都是一个值，而它的键就是该值所描述的信息，将数据组成键值对，并插入数据库中。

### 数据修改与删除

数据存储模块不仅需要支持批量数据的插入，还需要支持数据的修改与删除。例如，对脚本的用法与说明进行修改，对新增的数据进行添加，对重复的数据进行删除等等都需要数据库中的数据进行修改与删除。数据修改与删除的具体流程和方法设计如下：

1、连接数据库。利用python的pymongo包对MongoDB进行连接，程序进入到数据库中准备进行数据修改与删除操作。

2、确认数据所属集合。首先根据要修改的数据的类型（代码变迁数据，缺陷数据，邮件数据或脚本）来确认数据可能属于哪些集合；然后根据数据所属的系统来确定存储数据的集合。

3、定位数据。在将数据插入MongoDB时，它会自动为数据分配一个id值。修改或删除数据时，本文通过获取数据的id值来从集合中定位要修改或删除的数据。

4、修改或删除数据。利用pymongo包中的方法对数据进行修改或删除。

## 数据检索模块设计

数据检索模块需要提供给用户检索数据的接口，通过所提取的软件开发过程数据的描述信息来对软件开发过程数据进行检索。这个模块通过连接到数据存储系统，对数据库进行访问并进行查询来获取到要检索的数据。数据检索模块需要完成的功能有：1.对数据库的操作进行封装，本文需要对数据库进行操作来提供检索功能，首先要对数据库的这些操作封装成可以调用的函数以方便检索；2.提供简化的命令以方便操作，用户对数据的查询需要输入命令，这些命令的长度应该适当，需要设计合理的命令以方便用户进行数据检索；3.分割与组合，如上一章所述，本模块需要完成的一项功能就是将用户的查询请求进行分割，再对数据库进行查询，最后将查询结果合并。下文将分两个部分对数据检索模块进行介绍，包括操作封装与命令简化，请求分割与结果组合。

### 操作封装与命令简化

在数据检索时，主要使用到pymongo提供的数据的查询接口。在进行数据查询时，根据不同的数据类型，pymongo提供了不同的查询方法，本文也对这些方法进行了封装，具体如下：

* 字符串类型的数据。对这部分数据需要提供的查询方式就是正则匹配。在pymongo中，使用“$regex”这个参数代表正则匹配，本文将其封装为\_\_REGEX\_\_函数，调用这个函数并传入适当参数，即可返回查询结果。
* 数字类型的数据。数字类型数据即整型数据，在数据库中都是以整数形式进行存储的。对这部分数据需要提供包括大于、小于、大于等于、小于等于、不等于等这些比较数值的查询。例如，在pymongo中，使用“$lte”这个参数代表小于等于，本文将其封装为\_\_LTE\_\_函数。

对数据进行查询时，需要提供的参数包括三个：collection，即数据所属的集合；key，即数据的键，标记数据是表示什么信息的；val，即要查询的值。本文对这三个参数进行了封装，具体如下：

* collection，不同的集合中，包含的数据的描述信息不同，首先本文需要对collection这个参数进行封装，将用户对不同类型的集合的查询分发到不同的函数进行处理。本文设计了四个类：“logFind”封装了对代码变迁数据的查询语句，“bugFind”封装了对缺陷数据的查询语句，“mailFind”封装了对邮件数据的查询语句，“scriptFind”封装了对脚本的查询语句。
* key与val，上一步中按照集合对用户的查询操作进行了分配，同一类数据具有相同的key值。本文对key进行了封装，对不同key的查询，使用不同的函数。然后再根据不同的key，设定不同类型的val作为参数。

### 请求分割与结果组合

用户对数据的检索请求往往是横跨多个维度的。例如，用户需要检索时间跨度为100个月以上，缺陷报告数为10万以上的社区。本文封装的数据检索接口为对每项数据的键（key）提供对应的值（val）的检索，而例子中需要对维度一（时间）和维度三（项目统计数据）中的缺陷报告数两项键值对同时进行检索，本文设计的方法是：首先将用户的检索请求分割成多个单项的键值对检索，然后调用数据检索接口取得检索结果，最后将结果合并返回给用户，具体的设计如下：

* 请求分割：按照上文介绍的检索命令格式，用户对不同键值对的检索以空格进行隔开。在请求分割阶段，本文根据用户的输入，按照空格进行分割，并获取用户需要检索的所有键值对，最后调用相对应的检索接口对这些数据库进行检索，获得结果。
* 结果合并：上一步中，本文根据用户输入通过调用检索接口，从每个检索接口中获得了检索结果。而这些结果是以集合的形式存在的，集合中包含了一个或多个文档，在合并时，本文通过求取这些文档的交集来取得最终的结果并返回给用户。

# 工具的实现

上一章介绍了工具的总体设计以及各个模块的设计情况，本章将介绍各个模块的实现细节。

## 数据提取模块实现

为了实现数据存储模块，本文开发了一个单独的包，其中包含了数据提取模块中使用文件和函数。数据提取模块中的主要文件的具体信息如下表所示：

|  |  |
| --- | --- |
| 文件名 | 用途 |
| extr-hg.py | 提取vcs（版本控制系统）为mercurial的日志描述信息 |
| extr-git.py | 提取vcs为git的日志描述信息 |
| extr-baz.py | 提取vcs为bazaar的日志描述信息 |
| extr-svn.py | 提取vcs为svn的日志描述信息 |
| extr-bugzilla.py | 提取缺陷数据描述信息 |
| extr-bugzilla-bz2.py | 提取缺陷数据描述信息（原始文件以bz2打包） |
| extr-mailman.py | 提取系统为mailman的邮件描述信息 |
| extr-mbox.py | 提取系统为mbox的邮件描述信息 |

可以看出，提取模块按照日志数据描述、缺陷数据描述和邮件数据描述三个小模块进行实现。在提取日志描述信息和邮件信息时，根据不同的版本控制系统或邮件系统，利用不同的正则表达式，匹配得出统一的描述信息；在提取缺陷数据时，同样根据不同的数据格式，采用不同的正则表达式进行匹配提取信息，同时还需要根据输入文件格式的不同（普通文件或者压缩文件）编写不同的脚本来对数据进行处理。

## 数据存储模块实现

数据存储模块的主要功能是将提取模块中提取出的数据存入数据库中，同时提供包括对数据的添加，修改，删除等在内操作。数据存储模块中的主要文件的具体信息如下表所示：

|  |  |
| --- | --- |
| 文件名 | 用途 |
| insert-log.py | 将日志描述信息批量存入数据库 |
| insert-bug.py | 将缺陷数据描述信息批量存入数据库 |
| insert-mail.py | 将邮件数据描述信息批量存入数据库 |
| insert-script.py | 将脚本描述信息批量存入数据库 |
| pyupdate.py | 提供数据添加，修改，删除等功能 |

## 数据检索模块实现

数据检索模块需要根据用户的检索请求，通过检索数据库来返回检索结果给用户。数据检索模块中的主要文件的具体信息如下表所示：

|  |  |
| --- | --- |
| 文件名 | 用途 |
| pyfind.py | 对数据库的操作进行封装，提供抽象的数据检索接口 |
| bugFind.py | 具体实现缺陷数据的检索接口 |
| logFind.py | 具体实现日志数据的检索接口 |
| mailFind.py | 具体实现邮件数据的检索接口 |
| scriptFind.py | 具体实现脚本的检索接口 |

其中pyfind文件中定义的抽象的数据接口包括对字符串类型数据的检索以及数字类型数据的检索。例子如下：

|  |
| --- |
| //检索字符串：正则匹配  def \_\_REGEX\_\_(self, collection, key, val):  posts = self.db[str(collection)].find({str(key) : {'$regex' : str(val)}})  ret = []  for post in posts:  ret.append(post)  return ret  //检索数字：小于  def \_\_LT\_\_(self, collection, key, val):  posts = self.db[str(collection)].find({str(key) : {'$lt' : val}})  ret = []  for post in posts:  ret.append(post)  return ret |

## 配置文件

本文的工具除了实现以上几个模块，还需要为各个模块提供配置文件，在运行时通过配置文件提供必要的参数以帮助程序正常运行，这些配置文件包括以下几个部分：

|  |  |
| --- | --- |
| 文件名 | 内容 |
| collection.conf | 记录数据库中的集合的名字 |
| db.conf | 记录数据库的连接信息 |
| extr.conf | 记录数据提取阶段原始文件的位置 |
| header.conf | 记录每个集合中的键的名字 |

其中collection.conf用于记录集合的名字，包括数据库中存放日志数据、缺陷数据、邮件数据以及脚本的集合。db.conf记录程序初始化时，进行数据库连接所需的参数，包括数据库的ip，端口等。extr.conf记录数据提取模块所需的输入文件在服务器上的存储位置，在进行数据描述信息提取时，第一步我们通过linux的命令找寻到日志数据、缺陷数据、邮件数据以及脚本在服务器上存放的位置并将它们分别存在不同的文件中，这些文件就记录在extr.conf中，当作下一步的输入。header.conf中记录了集合中每个文档的键的名字，用于辅助数据检索，在数据检索时，本文根据用户的输入，通过对用户查找的键所对应的值进行匹配来进行检索并返回检索结果。

# 相关工作

# 参考文献

1. Minghui Zhou and Audris Mockus. What make long term contributors: Willingness and opportunity in OSS community. In ICSE 2012, pages 518-528, Zürich, Switzerland, 2012.
2. Audris Mockus, Roy T. Fielding, and James Herbsleb. Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11(3):1-38, July 2002
3. Audris Mockus. Amassing and indexing a large sample of version control systems: towards the census of public source code history. In 6th IEEE Working Conference on Mining Software Repositories*,* May 16-17 2009
4. Georgios Gousios , Bogdan Vasilescu , Alexander Serebrenik , Andy Zaidman, Learn GHTorrent: GitHub data on demand, Proceedings of the 11th Working Conference on Mining Software Repositories, May 31-June 01, 2014
5. 邓飞.软件开发过程数据池的设计与实现.北京大学硕士生学位论文.2011
6. 林洪武.**大数据源中代码演化的追踪工具的设计与实现.2012**
7. MongoDB. <https://www.mongodb.org/>
8. GHTorrent. <http://ghtorrent.org/>
9. Github. <https://github.com/>
10. OpenHub. <https://www.openhub.net/>

1. <https://www.openhub.net/> [↑](#footnote-ref-1)
2. <http://www.ghtorrent.org/> [↑](#footnote-ref-2)
3. <https://github.com/> [↑](#footnote-ref-3)