**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，提出一种面向软件开发过程源数据的处理和检索方法，并设计与实现一套软件开发过程数据的检索工具。

本文提出的方法与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。

关键词：软件开发过程数据，数据描述，数据存储，数据检索

目录

1 引言 4

2 相关技术介绍 7

2.1 软件开发过程数据 7

2.1.1 代码变迁数据 7

2.1.2 缺陷数据 8

2.1.3 邮件数据 8

2.2 MongoDB 9

2.3 相关工作 10

2.3.1 OpenHub 10

2.3.2 GHTorrent 10

2.4 数据来源 10

3 问题分析 11

3.1 软件开发过程数据的描述方法 11

3.2 软件开发过程数据的检索工具 13

4 工具的设计 15

4.1 工具的总体设计 15

4.2 数据提取模块设计 15

4.2.1 日志数据描述 15

4.2.2 缺陷数据描述 17

4.2.3 邮件数据描述 18

4.3 数据存储模块设计 19

4.3.1 数据集合设计 19

4.3.2 数据批量插入 20

4.3.3 数据修改与删除 21

4.4 数据检索模块设计 21

4.4.1 操作封装与命令简化 22

4.4.2 请求分割与结果组合 23

5 工具的实现 24

5.1 数据提取模块实现 24

5.2 数据存储模块实现 24

5.3 数据检索模块实现 24

6 工具展示 25

7 总结与展望 26

参考文献 27

附录一 图目录 28

附录二 表目录 29

附录三 工具控制台命令 30

致谢 31

# 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[Zhou 09]。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[Mockus 02]，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[Mockus 09]。

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[Gousios 14]，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[邓飞 11]，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。

上述软件开发过程数据具有海量、分布、异构等特点，这使得我们要整合与检索这些数据变得困难，原因如下：首先，分布是指数据分散在服务器上的各个角落，这使得寻找和整合这些数据变得困难。其次，异构则指存储软件开发过程数据的系统各异，例如存储源代码的版本控制系统就包括SVN, GIT, Bazaar等，而不同的系统的数据存储格式也不同，这就使得处理这些数据时需要根据不同的系统采用不同的方法。此外，要对大量的软件开发过程数据进行检索，就需要寻找合适的检索数据的方法，以帮助用户快速地查询到所需的数据。综上所述，我们要检索软件开发过程数据，需要面对与解决不少问题和挑战。

本文尝试应对上述问题和挑战，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。具体来讲，本文提出的工具有以下几个方面：第一，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，用于描述与检索数据；第二，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，用于描述数据是如何收集与处理的；第三，通过数据与脚本的描述信息，建立检索工具，用于根据用户的检索需求定位和获取所需数据。

本文主要解决的的问题在于：第一，为纷繁复杂，类型多变的软件开发过程数据建立格式统一，描述简单，易于理解的描述信息。软件开发过程数据的存储系统众多，不同的软件项目使用不同的系统，不同的系统的数据格式不尽相同。本文需要对这些数据进行统一整理，提取出格式统一，描述简单，易于理解的软件开发过程数据的描述信息。第二，为脚本进行分类与建立描述信息。脚本数据与软件开发过程数据不同，不能直接通过编写程序来提取出描述信息，本文需要人为的对处理数据的脚本进行分类，并建立包括脚本输入输出，用途等脚本的描述信息，以帮助我们了解软件开发过程数据的收集与处理方法。第三，从大量的软件开发过程数据中定位到用户要检索的数据。首先选取合适的存储系统对描述信息进行存储，再对上述信息建立一套检索接口，将用户的检索请求分割成可执行的检索语句，利用检索接口进行检索，并将结果合并后返回给用户。

本文的主要贡献在于设计与实现了一种软件开发过程数据的检索工具。本文通过建立软件开发过程数据的描述信息来对数据进行检索，帮助研究者与开发者快速地获取到想要查找的软件开发过程数据。

本文结构安排如下：第一章是引言；第二章介绍软件开发过程数据的特点以及存储与检索所用到的相关技术；第三章对本文面临的问题与挑战进行分析并提出解决方案；第四章给出工具的设计；第五章介绍工具的具体实现；第六章对工具的效果进行展示；第七章介绍本文的工作进展和对未来工作的展望。

# 相关技术介绍

本文面向包含代码变迁数据，缺陷数据，邮件数据等软件开发过程数据，实现对这些数据的收集，整理，存储与检索。本文需要从大量的日志数据，缺陷数据和邮件数据中抽取数据的描述信息并进行存储，在此基础上建立这些软件开发过程数据的索引，并提供检索这些数据的工具。因此，本文面临着与多种软件开发过程数据的存储系统进行交互，对数据进行处理并建立数据的描述信息，利用描述信息对数据进行检索等问题。针对上面的问题，以下介绍本文面向的软件开发过程数据，数据的存储，数据的检索等方面的相关技术以及本文的相关工作。

## 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。

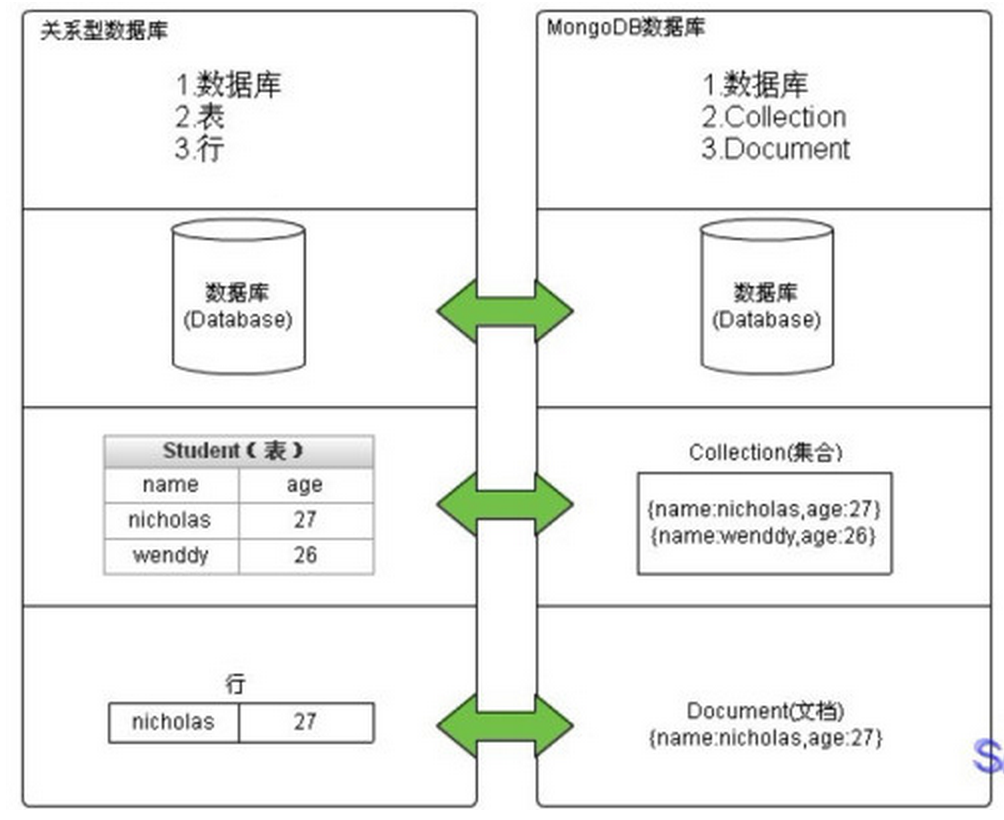
### 邮件数据

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。

本文所使用的具体方法将在后续章节中描述。

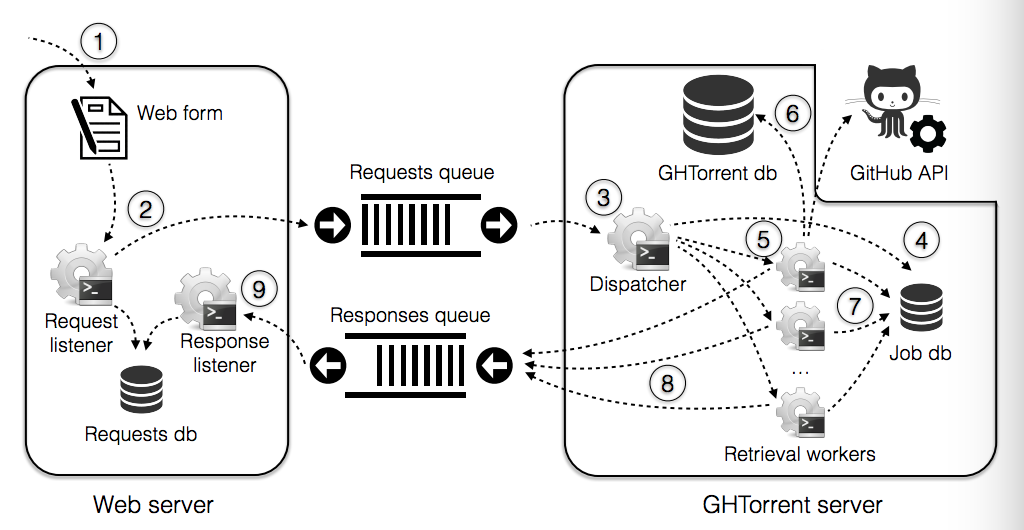
## 相关工作

### OpenHub

OpenHub是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。这些基本信息包括如下几个部分：

### GHTorrent

GHTorrent是一个向用户提供Github数据的工具，它利用Github API为用户提供数据定制的功能。该工具的具体流程如下图所示：



首先用户通过网页端输入需要获取的信息，输入可以是一个或者多个github上的项目的名字以及接受结果的email账户；然后，后台服务器根据用户的输入，调用github的API获得数据，这些数据包括项目的缺陷数据，代码提交数据，项目的fork和pull\_request[[4]](#footnote-4)信息，项目的watcher和follow信息[[5]](#footnote-5)等；接着，在获取了这些数据以后，将它们以MySQL数据表和MongoDB数据表的形式存在后台服务器；最后，后台服务器将MySQL数据表和MongoDB数据表传回给用户指定的email账户。通过GHTorrent可以获得github上的已格式化好的数据，这些数据可供研究者很方便地进行研究。

## 数据来源

# 问题分析

本文致力于回答的问题是：如何建立起一套软件开发过程数据的检索方法与工具。要回答这个问题，需要解决如下两个方面的困难：第一，描述软件开发过程数以及这些数据相关的处理脚本；第二，提供一套检索软件开发过程数据的工具。下面具体从这两个方面对本文遇到的问题进行分析。

## 软件开发过程数据的描述方法

本文的数据来自于北京大学软件工程研究所数据分析小组的实验数据（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）。本文要描述软件开发过程数据以及数据相关的处理脚本，就需要从数据与其处理脚本中获取信息，以帮助我们建立一套软件开发过程数据的描述方法。本文要建立一套描述方法，具体需要解决的问题如下：

* 定义合理的收集脚本的方法。本文要从软件开发过程数据的处理脚本中获取信息，首先需要收集所有的处理脚本。然而，处理脚本分散在小组的服务器的各个角落，收集脚本时要不重不漏，因此就需要定义合理的脚本收集方法来对脚本进行收集整理，记录脚本的存储位置以便后续分析与使用。
* 定义合理的脚本分类的方法。软件开发过程数据的处理脚本的用途不同，包括收集数据，存储数据，分析数据等。同时，软件开发过程数据的类型不同，系统不同，项目不同等也造成了处理脚本的不同，这就导致了处理脚本数量众多。定义合理的脚本分类方法可以帮助我们快速的查找到合适的脚本来处理数据。
* 建立数据的描述维度。根据用户的需求去找寻描述软件开发过程数据的维度。软件开发过程数据的格式纷繁复杂，系统类型多变，处理的同时要考虑到用户在查询时可能会关注的信息。最终选取维度建立格式统一，描述简单，易于理解的描述信息，帮助用户理解与查询数据。
* 建立脚本与软件开发过程数据之间的连接。每一种类型的软件开发过程数据都有与之对应的处理脚本，收集完脚本信息之后，需要将脚本与数据之间的联系建立起来，以建立一套完整的软件开发过程数据的描述方法。

上文分析了本文要描述软件开发过程数据及其脚本时需要面对的问题。下文将对上述问题提出本文的解决方案。解决方案包括数据收集，信息提取两个方面：

数据收集包括寻找脚本和软件开发过程数据在服务器上存储的位置，一方面可以帮助用户定位到这些数据，另一方面可以使后续工作更方便地对这些数据进行统一地处理与分析，提取描述信息。本文在服务器上定位数据的方法大部分是根据文件的后缀名来查询文件，具体的定位数据的方式包括以下两个方面：

* 脚本定位，脚本按照语言类型的不同其后缀名也不同。处理数据的脚本一般分为三种语言类型：python，perl和shell，其后缀名分别为.py，.perl和.sh。本文利用linux的find命令根据文件后缀名来寻找并定位脚本。
* 软件开发过程数据定位，本文面向的软件开发过程数据本身分为三类：代码变迁数据，这部分数据在克隆数据的时候记录了版本库和所存放的位置；缺陷数据，在抓取这部分数据时，我们对原始数据的后缀名进行了统一，都是.level0；邮件数据，根据邮件系统的不同，邮件数据的后缀名分为两种，.mbox和.mailman。最后可以同脚本定位类似，利用find来查找并定位数据。

信息提取需要建立维度来描述软件开发过程数据及其处理脚本。其中包括处理脚本的分类以及建立数据的描述维度。具体的信息提取的方式包括以下两个方面：

* 脚本分类，由于要建立软件开发过程数据与脚本之间的联系，所以最直观的脚本分类方法就是将脚本按照处理数据的类型横向分为三类：代码数据相关脚本，缺陷数据相关脚本，邮件数据相关脚本。同时，每一类又可以根据数据处理的流程来纵向划分：数据抓取脚本，数据存储脚本，数据分析脚本等。
* 数据描述维度，数据描述维度按照用户的查询需求来对数据进行描述。用户的查询需求比较抽象，例子如下：查找一个项目的代码数据，要求存在超过10年，代码提交人数大于100人；查找一个项目的缺陷数据，要求报告数大于5万个；找出服务器上缺陷报告数排前10的10个项目等等。根据例子可以看出，维度包括几个方面：1、时间跨度，时间长度往往代表一个项目的成熟程度，用户关心所选取的项目数据是否来自于成熟的项目，那么时间跨度会是一个很好的判断标准；2、参与人数，人是项目开发的主导者，人数越多代表这个项目越庞大，用户在选取项目数据时，往往也会关系一个项目的参与人数；3、各类数据的统计信息，包括代码提交数，报告提交数，邮件数等；4、项目标志，所谓的项目标志，就是项目的名字，所用的存储系统，项目所在的社区等等一些标志性的字符，用户可能有明确查询对某个项目的数据或者某种类型的版本控制系统数据的需求，项目标志可以帮助用户快速定位数据。总的来说，维度的设计就是根据拆分用户的查询需求来实现的，具体的维度设计情况将在后续章节中介绍。

## 软件开发过程数据的检索工具

软件开发过程数据包含代码变迁数据，缺陷数据，邮件数据等。我们要使用这些数据，就需要提供一套检索工具帮助我们查询到想要使用的数据。本文要设计一套检索工具，具体需要解决的问题如下：

* 数据与脚本信息的存储。本文需要利用数据的描述信息来对数据进行检索，首先需要选取合适的存储系统对描述信息进行存储。
* 数据的高效检索。由于用户的检索请求比较抽象，需要对本文收集的描述信息建立一套检索接口，将用户的检索请求分割成可执行的检索语句，再调用检索接口进行检索并得到结果。

面对上述难点，本文提出以下解决方案：数据检索根据用户的检索需求，设计合理的检索接口，提供数据检索的功能。本文在描述信息提取部分建立了数据的描述维度，每一个维度都是最小的检索可执行单元，多个执行单元的结果合并可以得到最终的查询结果。具体来说，数据检索的可以分成以下几个部分：

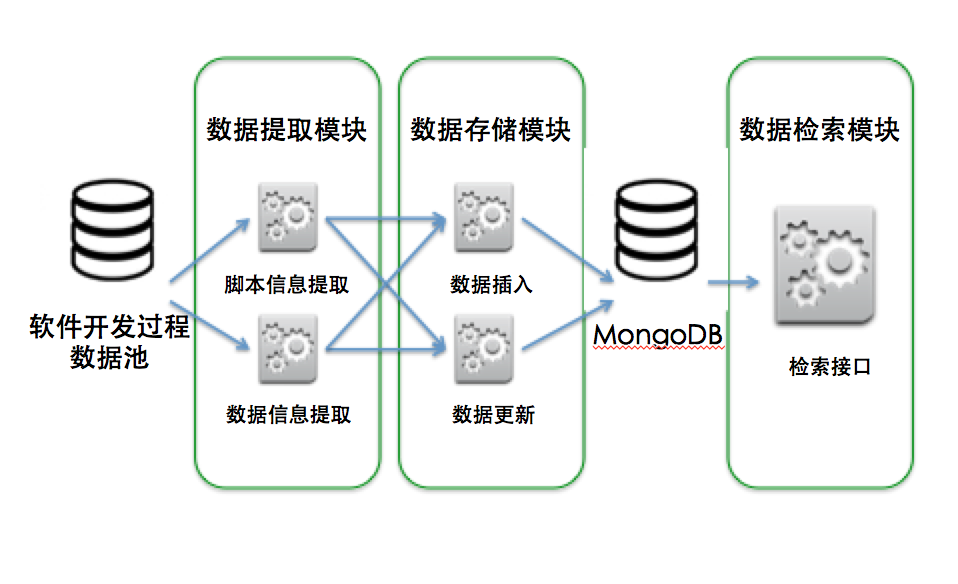
* 请求分割，把用户的查询请求按照所包含的维度进行分割。例如，查找一个项目的代码数据，要求存在超过10年，代码提交人数大于100人。这个请求包含时间跨度和参与人数两个维度，则分成两个查询语句，分别对时间和人数进行查询。
* 检索数据，按照上一步分割出来的查询语句，提供查询接口，到存储系统中进行查询，并将结果一一返回。
* 合并结果，将上一步中返回的多个结果进行合并，合并时遵循逻辑合并，即“与，或，非”等运算法则。

# 工具的设计

上一章我们对本文面临的主要问题及其解决方案进行了阐述。本章将介绍工具的设计，以及如何解决之前提出的问题。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。在之前章节也介绍了代码变迁数据的一些特点，在对日志数据进行描述信息提取时，要解决的问题包括以下几个方面：

* 在从版本控制系统中获取日志时，需要根据不同的版本控制使用不同的命令来取得日志信息。
* 不同的版本控制系统的数据格式不相同，需要针对每个版本控制系统设计和编写特定的提取脚本，提取出上文提出的四个维度的描述信息。（维度一：时间；维度二：人数；维度三：项目统计数据；维度四：项目标志）

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。具体命令如下表：

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，提取出的描述信息包括：

* vcs（版本控制系统）：版本控制系统的名字。
* repo（代码库）：项目的代码所属于的代码库或者社区的名字，例如：github、googlecode等。
* prj（项目）：项目的名字。
* n\_peo（人数）：项目的版本控制系统的参与人数，体现了一个项目中参与项目源代码开发的人数。
* b\_time（项目数据起始时间）：服务器上记录的该项目的最早的代码变迁数据的提交时间。
* e\_time（项目数据终止时间）：服务器上记录的该项目的最晚的代码变迁数据的提交时间。
* span（项目时间跨度）：服务器上记录的该项目的代码变迁数据的时间跨度。
* n\_cmt（代码提交次数）：项目的代码提交次数，体现一个项目的代码变更的活跃程度。
* log\_loc（日志存储位置）：项目的日志存放在服务器上的物理位置。
* src\_log（代码变迁数据存储位置）：项目的代码变迁数据存放服务器上的物理位置。

其中b\_time、e\_time、span属于维度一，n\_peo属于维度二，n\_cmt属于维度三，其余属于维度四。

### 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，提取出的描述信息包括：

* its（缺陷追踪系统）：缺陷追踪系统的名字。
* repo（社区）：缺陷数据所属于的社区的名字，例如：gnome、mozilla等。
* n\_peo（人数）：项目的缺陷追踪系统的参与人数，体现了一个项目中参与提出问题和新功能的人数。
* b\_time（缺陷数据起始时间）：服务器上记录的缺陷追踪系统中最早的缺陷报告的报告时间。
* e\_time（缺陷数据终止时间）：服务器上记录的缺陷追踪系统中最晚的缺陷报告的报告时间。
* span（缺陷数据时间跨度）：服务器上记录的缺陷追踪系统中的缺陷报告的时间跨度。
* n\_bug（缺陷报告数）：缺陷追踪系统中的缺陷报告数，体现一个社区的缺陷追踪系统的活跃程度。
* loc（缺陷数据存储位置）：项目的缺陷数据存放在服务器上的物理位置。

### 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，提取出的描述信息包括：

* mlist（邮件系统）：邮件系统的名字。
* repo（社区）：邮件数据所属于的社区的名字，例如：apache。
* prj（项目）：项目的名字。
* n\_mail（邮件数）：邮件系统中的邮件数，体现了一个项目中开发者的交互活跃程度。
* n\_peo（人数）：邮件系统的参与人数。
* loc（邮件数据存储位置）：项目的邮件数据存放在服务器上的物理位置。

## 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。集合分类要解决的问题是：数据要分成几个集合来存储，每个集合又是按照什么样的标准来分类的。首先，数据提取模块中提取了软件开发过程数据的描述信息，按照数据类型分成了三类：代码变迁数据，缺陷数据和邮件数据。其次，本文还需要对软件开发过程数据相关的处理脚本的信息进行存储，因此还需要一类集合来记录脚本信息。对于每一个大类，由于数据存储系统的不同，数据的格式也不同，还需要按照系统类别细分成几个小类，一方面可以方便用户检索特定系统的数据，另一方面可以更好的建立脚本与数据的连接。代码变迁数据的存储集合可分为svn，git，hg，bazaar；缺陷数据的存储集合可分为level0，level1；邮件数据的存储集合可分为mbox，mailman；脚本的存储集合可分为perl，python，shell。综上，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。MongoDB集合内的数据以文档形式存储。每个文档是一个键－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。在数据提取模块设计阶段，本文分别为每一类软件开发过程数据设计了要提取出的描述信息。集合内的文档结构就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。具体的文档结构设计展示如下：

### 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

MongoDB提供了强大的数据查询功能，可以快速的通过查询语句来检索数据。pymongo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。

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。本文中以字符串类型存入数据库的数据即上一章所说的项目标志这个维度的数据，例如社区名字，项目名字等。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“$regex”这个参数代表正则匹配，本文将其封装为\_\_REGEX\_\_函数，调用这个函数并传入适当参数，即可返回查询结果。
* 数字类型的数据。数字类型数据即整型数据，在数据库中都是以整数形式进行存储的。本文中以整型存储的数据包括上一章所说的时间跨度、参与人数与统计信息这几个维度的数据，例如项目时间跨度，项目参与人数等。对这部分数据需要提供包括大于、小于、大于等于、小于等于、不等于等这些比较数值的查询。例如，在pymongo中，使用“$lte”这个参数代表小于等于，本文将其封装为\_\_LTE\_\_函数。

从上表可以看出，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。如前文所说，数据集合共有三种大的类型，包括代码变迁数据，缺陷数据和邮件数据，同时还包括一个脚本类型的集合。每一个大类中的数据的描述信息相同，所以本文设计了四个类：“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。这就使得用户对数据的查询可以按照其所属大类分配到特定的函数进行查询。
* 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，例如bug数据都包含repo，n\_peo，span等数据。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。例如对项目名字的正则匹配查询，封装成函数名为find\_prj，参数为一个字符串类型的参数的函数；对时间跨度的范围查询，封装成函数名为find\_span，参数为两个整型的参数的函数，两个整型分别代表范围查询的上界和下界。

上式是pymongo对MongoDB中某项数据进行查询时的语句，每次查询时都输入长串的命令会导致查询的效率下降。本文通过对数据查询方式，集合和key与val的封装，实现了在查询时只需要用户提供少量的必要参数，通过这些参数自动分配函数对数据进行检索并返回结果，从而达到命令简化的目的。用户在进行数据检索操作时，需要在控制台输入的命令格式为：“collection key;op;val[s]”。具体解释如下：

* collection，表示数据所在集合，这里使用集合名字的缩写来唯一标识一个集合，如svn，git等；key，表示要检索的键的名字，如项目名字，项目时间跨度等；op，表示数据查询的方式，如正则匹配，小于，大于等；val，表示要查询的值，如要查询项目的名字为apache，项目跨度大于100月，这里所述的apache，100都是val。

### 请求分割与结果组合

# 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

## 数据存储模块实现

## 数据检索模块实现

# 工具展示

# 总结与展望

# 参考文献

# 附录一 图目录

# 附录二 表目录

# 附录三 工具控制台命令

# 致谢

1. <https://www.openhub.net/> [↑](#footnote-ref-1)
2. <http://www.ghtorrent.org/> [↑](#footnote-ref-2)
3. <https://github.com/> [↑](#footnote-ref-3)
4. github上用来进行项目协同开发的一套机制 [↑](#footnote-ref-4)
5. github上用来给用户关注项目的一套机制 [↑](#footnote-ref-5)