****

软 件 体 系 结 构

开源系统的软件体系结构的建模与分析

OODT-FileManager的建模与分析

小 组 成 员 陈鸿超（SY1806214）

李铎坤（SY1806219）

刘 颖（SY1806418）

编制时间： 2019年5月

目录

[1 项目简介 3](#_Toc10040323)

[2 需求分析 5](#_Toc10040324)

[2.1 数据采集 5](#_Toc10040325)

[2.2 数据记录与管理 5](#_Toc10040326)

[2.3 数据存储 5](#_Toc10040327)

[2.4 数据查询 5](#_Toc10040328)

[2.5 数据传输 5](#_Toc10040329)

[2.6 配置管理 6](#_Toc10040330)

[2.7 异常处理 6](#_Toc10040331)

[2.8 质量属性 6](#_Toc10040332)

[3 软件体系结构设计 9](#_Toc10040333)

[3.1 数据采集模块 9](#_Toc10040334)

[3.2 数据记录与管理模块 10](#_Toc10040335)

[3.3 数据存储模块 11](#_Toc10040336)

[3.4 数据查询模块 12](#_Toc10040337)

[3.5 数据传输模块 13](#_Toc10040338)

[3.6 配置管理模块 15](#_Toc10040339)

[3.7 异常处理模块 16](#_Toc10040340)

[3.8 设计模式分析 16](#_Toc10040341)

[3.8.1 工厂模式 16](#_Toc10040342)

[3.8.2 策略模式 17](#_Toc10040343)

[3.8.3 迭代器模式 17](#_Toc10040344)

[3.8.4 享元模式 17](#_Toc10040345)

[4 设计特色分析 18](#_Toc10040346)

[4.1 目录属性的继承机制 18](#_Toc10040347)

[4.2 C/S模式 18](#_Toc10040348)

[4.3 不足之处 18](#_Toc10040349)

[5 组内分工情况 20](#_Toc10040350)

[6 参考文献 21](#_Toc10040351)

[7 附件：体系结构的分解过程 22](#_Toc10040352)

# 项目简介

OODT(Object Oriented Data Technology)是由Apache开发的一款面向元数据对象的智能数据管理与处理技术，支持分布式集群运行，可以帮助用户生成、管理、传输、处理与分析数据，是Apache软件基金会的一个顶级项目。

OODT主要由File Manager、Workflow Manager、Resource Manager三个主模块构成，如图1.1所示。

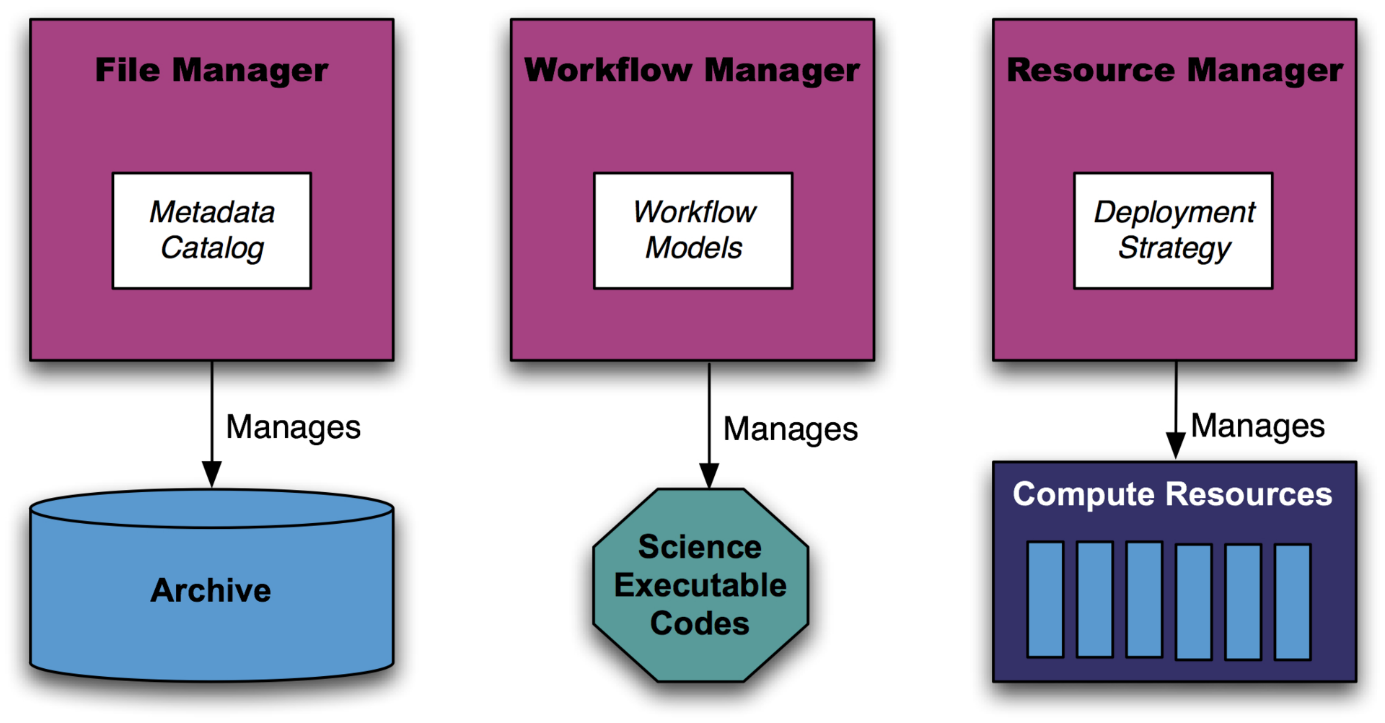


图 1.1OODT模块分解

其中，File Manager模块主要负责文件数据的管理，包括文件数据的采集、记录管理、数据存储、数存传输等；Workflow Manager模块主要负责管理与生成科学的可执行代码，用于数据的处理与分析；Resource Manager模块主要负责管理计算资源，分配计算任务，形成集群管理策略。

OODT的整体代码统计如表1.1所示。

表格 1.1OODT代码统计结果

|  |  |
| --- | --- |
| 统计项 | 结果 |
| 代码行数 | 149929 |
| 注释行数 | 62800 |
| 包的个数 | 254 |
| 类的个数 | 813 |
| 接口个数 | 116 |

接下来，本组将对OODT的File Manager模块进行体系结构的建模与分析。

# 需求分析

作为一个灵活的文件数据归档管理工具，File Manager模块主要有以下几方面的需求：数据采集、数据记录与管理、数据存储、数据查询、数据传输、配置管理、异常处理。下面将对这些方面进行需求分析并介绍该模块的主要质量属性。

## 数据采集

数据采集的主要任务是对用户指定的文件数据信息进行采集，然后交由后续的目录管理模块进行记录与转储。因此，数据采集模块需要满足以下需求：首先，该模块需要支持用户自定义采集项，即需要采集的文件属性，也叫做元数据，包括但不限于文件名、文件大小、标题、文件原始路径、文件最终路径等属性；其次，该模块要能够按照指定的采集项对用户指定的文件数据进行采集，然后返回采集结果。

## 数据记录与管理

数据记录的任务是对数据采集阶段获取的数据进行记录。为此，文件管理系统需要在一个目录中进行文件信息的记录和维护。记录的描述数据由用户指定，包括文件名、原始文件地址等。同时为了方便用户对目录进行管理，文件管理系统应该支持用户对记录进行数据获取、删除和查询等操作。

## 数据存储

当文件被记录在目录中后，文件管理系统需要将文件写入到磁盘中，并在目录中记录文件在磁盘中的存储地址。数据存储需要支持不同需求下的存储路径格式，例如用户需要记录产生文件的日期，或需要按照类型归纳整理文件。

## 数据查询

File Manager模块客户端采用的是命令行的形式，需要能够在命令行模式下调用各功能模块的代码进行相关的查询、文件获取、元数据修改和记录数据删除等操作，除了基本的功能性实现，同时也需要对命令行下输入的各种异常状况进行处理，并且需要对控制台的输出格式进行一定控制，提供用户命令交互的易用性。

## 数据传输

当使用File Manager模块对文件数据进行归档管理时，可能需要与本地文件系统、NFS网络文件系统、远端File Manager及某些云服务之间执行数据传输操作。而实际传输可能包含上下行的数据移动方向，即将新添加管理的文件数据传输至File Manager存储位置下，和将File Manager管理下的对象取回到当前文件管理系统下，所以该模块需要支持以上不同情况下的数据传输实现，并对数据传输过程中的状态信息进行追踪。另外，还需要对File manager模块所管理的对象提供删除功能。

## 配置管理

配置管理几乎是所有软件都要有的功能，OODT也不例外，其中File Manager模块需要进行配置管理的主要内容包括文件采集项、目录中记录条目的分组逻辑、元数据的组合逻辑等，都与文件数据的记录相关。因此，配置管理模块首先应该提供记录功能，能够显示的记录与展示配置配置信息；其次，该模块需要提供修改功能，允许用户使用多种方式修改配置信息；然后，该模块还需要提供加载功能，在软件运行前期就加载好所有必要的配置信息，以确保各功能的正常执行；最后，该模块需要提供监听功能，以获取运行阶段配置信息的改动情况。

## 异常处理

文件管理模块在进行数据采集、数据管理等操作时，难免会出现异常。因此文件管理系统需要异常处理模块，来提供程序运行时出现的任何意外或异常情况的方法。

## 质量属性

软件一般从功能和质量两部分进行评价。其中，一个软件的质量是指组件、系统或者过程满足指定需求或者用户、客户需求及期望的程度。而质量属性是印象质量的相关因素，是对质量的描述，根据上述需求分析，我们从性能、互操作性、可维护性和可重用性四个角度对系统进行了质量属性的刻画。

性能质量属性如表2.1所示。

表格 2.1性能质量属性

|  |  |
| --- | --- |
| 场景 | 大数据分析 |
| 源 | 用户 |
| 刺激 | 大量文件需要存储 |
| 制品 | 整个系统 |
| 环境 | 系统正常运行 |
| 响应 | 大量的文件被快速处理 |
| 响应度量 | 文件处理速度，用户等待时间 |

互操作性质量属性如表2.2所示。

表格 2.2互操作性质量属性

|  |  |
| --- | --- |
| 场景 | 用户需要较为复杂的功能 |
| 源 | 用户 |
| 刺激 | 组件之间需要进行数据交换 |
| 制品 | 高内聚、低耦合的系统 |
| 环境 | 系统正常运行 |
| 响应 | 各个组件以统一的标准协调完用户要求 |
| 响应度量 | 用户功能的完成质量 |

可维护性质量属性如表2.3所示。

表格 2.3可维护性质量属性

|  |  |
| --- | --- |
| 场景 | 系统需要升级 |
| 源 | 开发人员 |
| 刺激 | 系统有漏洞，需要升级 |
| 制品 | 升级后的系统 |
| 环境 | 系统正常运行 |
| 响应 | 修复了系统的漏洞，未产生副作用 |
| 响应度量 | 升级时间 |

可重用性属性如表2.4所示。

表格 2.4可重用性属性

|  |  |
| --- | --- |
| 场景 | 系统需要扩展 |
| 源 | 开发人员 |
| 刺激 | 系统需要增加新功能 |
| 制品 | 升级后的系统 |
| 环境 | 设计时，运行时 |
| 响应 | 系统增加了新功能，未产生副作用，代码复用性高 |
| 响应度量 | 重复代码增加量，模块重用数量 |

# 软件体系结构设计

由需要分析可知，OODT的File Manager模块在设计时可以划分为7个主要子模块，如图3.1所示。

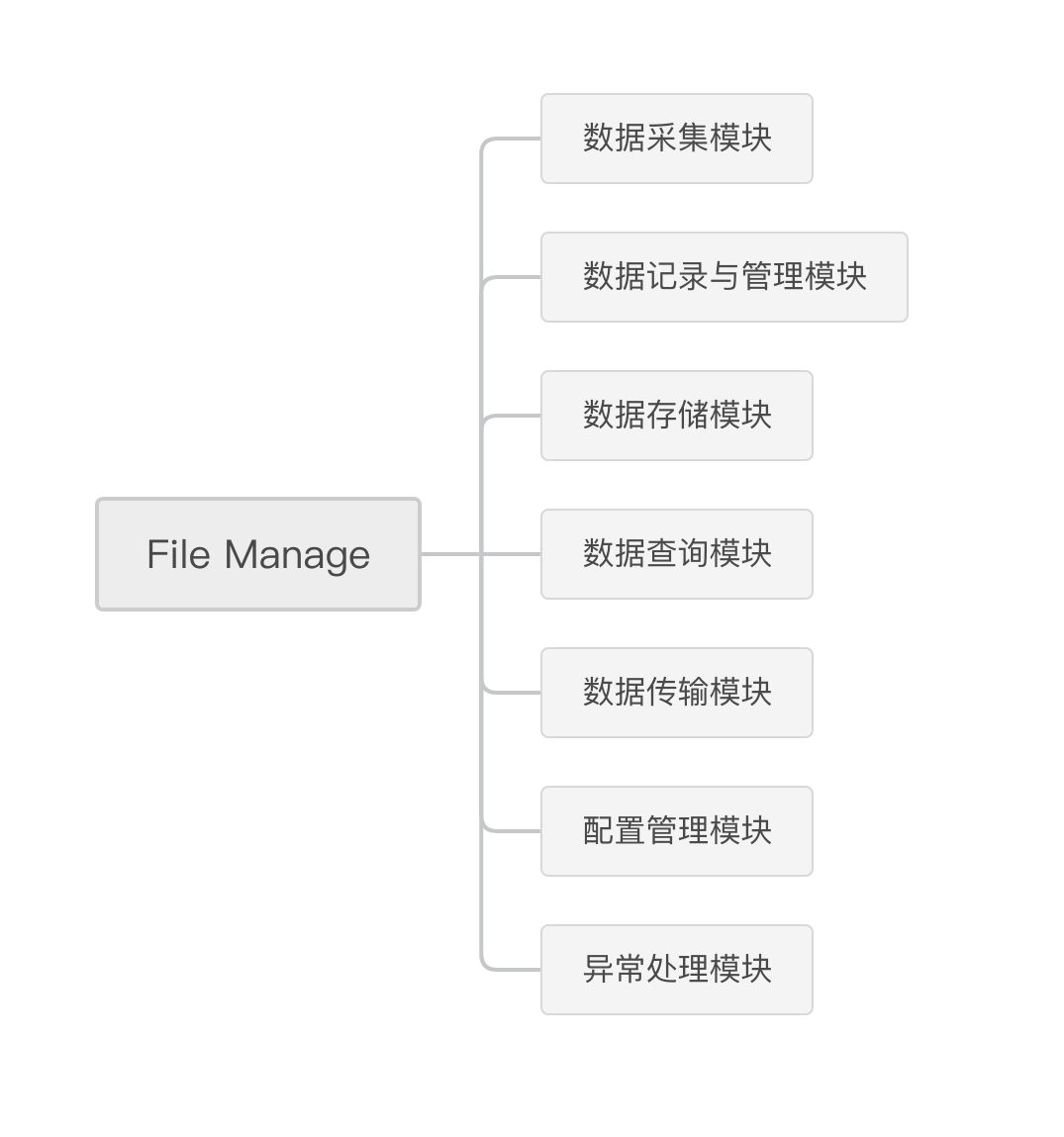


图 3.1File Manager的主要子模块

其中数据采集模块主要负责文件数据的信息采集；数据记录与管理模块主要负责采集信息的统一记录与管理，同时提供访问目录中记录数据的接口；数据存储模块主要负责文件数据的转储；数据查询模块主要封装了客户端对目录中记录数据的查询、获取、修改与删除功能；数据传输模块主要负责数据与请求在不同模块之间的传递，不同模块可以在同一台主机上也可以分别在不同的主机上；配置管理模块主要负责配置信息的记录、修改、加载和监听，以保证各功能的正确运行；异常处理模块则对整个File Manager模块运行时可能出现的主要异常进行了封装与响应，以提高软件的鲁棒性。

## 数据采集模块

由需求分析可知，数据采集模块的主要功能是根据用户指定的采集项对用户指定的文件数据进行信息采集，待采集的文件可以是存放于某个主机的硬盘中的文件，也可以是已经加载进缓存的文件。该模块的主要类图如图3.2所示。

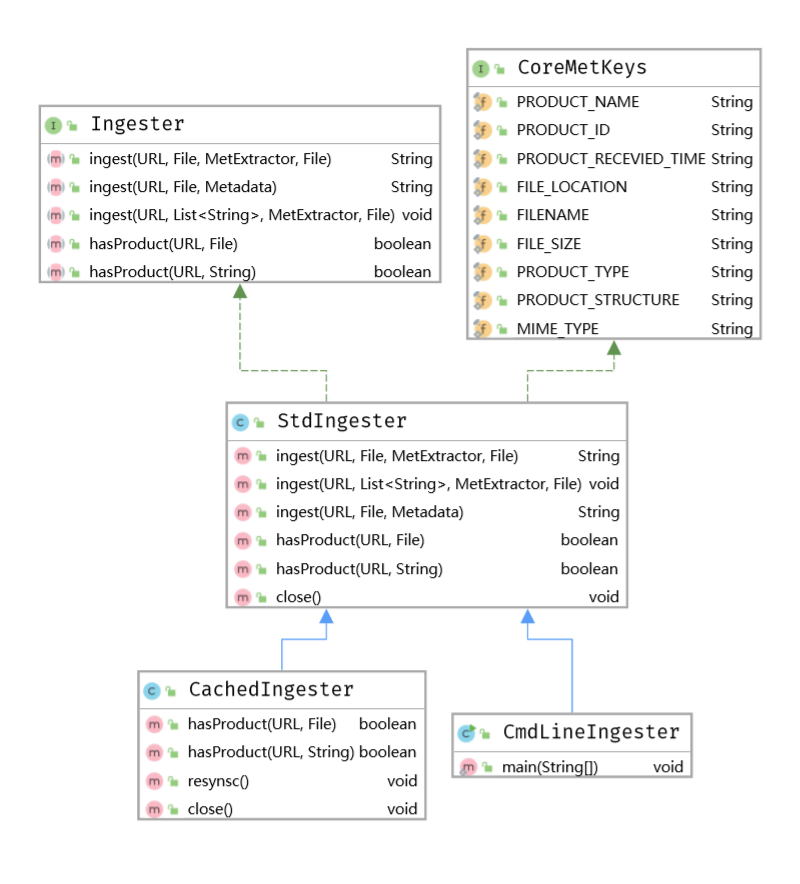


图 3.2数据采集模块主要类图

其中Ingester接口定义了一个文件采集器需要实现的多种采集方法；CoreMetKeys接口中定义了基本的文件采集项；StdIngester是一个文件采集器类，实现了上述两个接口中定义的采集方法与采集项，同时也定义了其他与数据采集相关的功能函数；CmdLineIngester文件采集器类继承于StdIngester类，主要用于以命令行的方式调用文件采集功能的情况；CachedIngester文件采集器类也继承于StdIngester类，主要用于从缓存中采集文件信息的情况。

## 数据记录与管理模块

数据记录与管理是通过Catalog来实现的，它类似于数据库中的表，记录着所有数据，在表格中存放了文件引用路径与元数据等。该部分通过设计接口Catalog来实现，该接口主要包括对Metadata和Product的增加、删除、检索查询等操作，以及查询结果分页显示等后处理方法。Catalog接口并未设计具体的从特定后端存储库检索和存储文件引用路径和元数据的检索算法，检索算法由SolrCatalog 和 LuceneCatalog实现。SolrCatalog 使用Solr引擎作为元数据存储和检索的后端，LuceneCatalog则使用Apache的Lucene文本搜索引擎作为后端。同时，该部分还实现了DataSourceCatalog与ScienceDataCatalog，它们适用于文件为SQL数据库与数据规模较大的情况。DataSourceCatalog也实现了两种不同的扩展功能MappedDataSourceCatalog与LenientDataSourceCatalog，可以对ProductType做映射，以及支持动态域添加。

该模块的主要类图如图3.3所示。

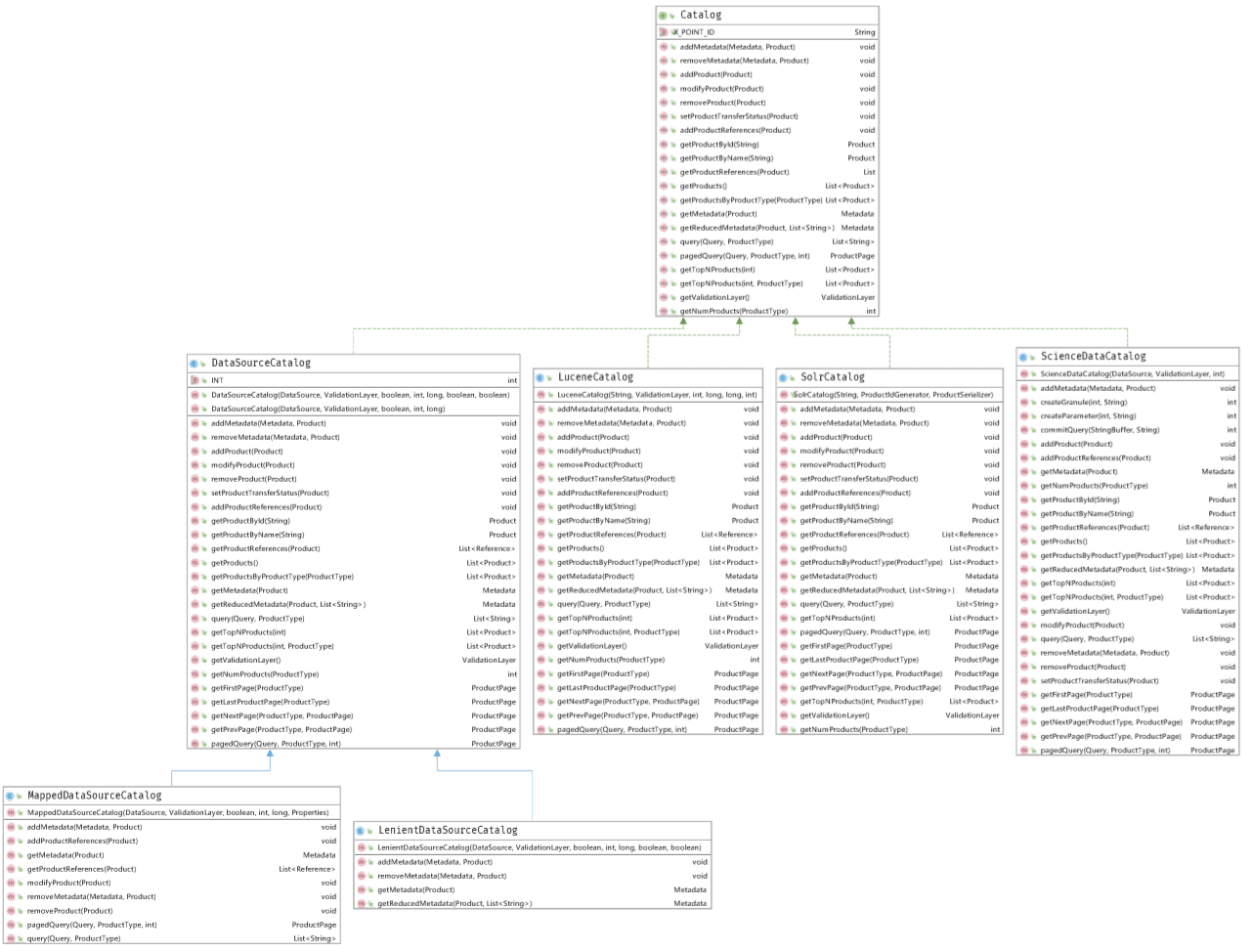


图 3.3数据记录与管理模块主要类图

## 数据存储模块

数据存储部分将目录中的信息存储在磁盘上，通过Versioner类来实现。它制定了目录中的信息存储在数据库中的模式，并将信息存储在磁盘上。存储功能通过接口 Versioner 中的createDataStoreReferences(Product product, Metadata metadata) 方法来实现。如无任何特殊需求，存储功能由 BasicVersioner完成，它只是使用Product名称和其对应的ProductType的引用路径来创建Product的实际存储路径，并将Product的文件内容写入对应的存储路径。该部分还实现了更加复杂的存储器，比如说DateTimeVersioner可以将Product产生的具体日期格式化在文件名中，SinglefileBasicVersioner 将ProductType和Product文件名合并作为路径，MetadataBasedFileVersioner则将更为复杂的元数据格式化后作为路径的一部分进行存储。更复杂的Versioner建立在MetadataBasedFileVersioner的基础上，比如说DirectoryVersioner在其基础上实现了层次化的目录存储，AcquisitionDateVersioner实现了对获取日期的自动记录。

该模块的主要类图如图3.4所示。

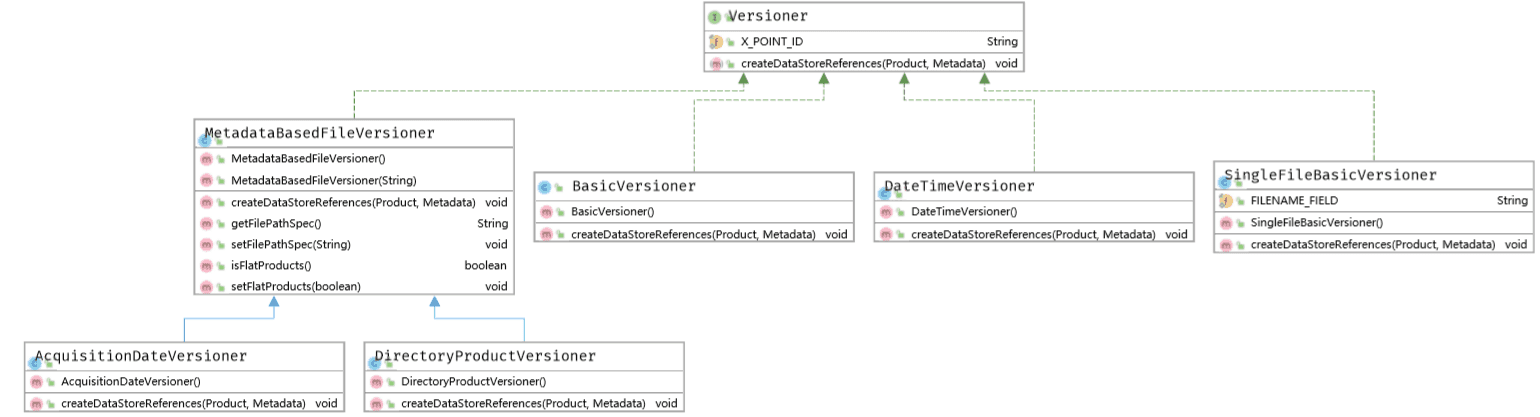


图 3.4数据存储模块主要类图

## 数据查询模块

数据查询模块的目的是为了设计一个便于用户交互的命令行交互客户端，首先需要对命令输入的格式进行解析，该部分设计了parse模块对标准输入的命令进行切分，并根据option模块定义的一系列命令选项进行归类处理。为了能够统一控制标准输出的内容与形式，此处首先设计了CmdLinePrinter/StdCmdLinePrinter类对命令行客户端的运行时提示与字符界面进行统一格式化管理，又设计了PrintMessageAction类对功能代码运行时产生的输出进行统一存放。

另一方面，为了调用系统已经设计好的其他功能性模块，首先，定义了XmlRpcFileManagerClient类对所有客户端可用的功能模块的调用进行了封装。然后建立一个CmdLineAction接口类，各个实际功能调用类需要实现该接口的execute方法。

在每个实际功能调用类的execute方法中，首先对之前命令解析的参数的实际意义进行判别，对异常情况进行处理，然后再实例化XmlRpcFileManagerClient类，并根据参数的不同情况，进行可能的预处理操作，再调用对应功能模块执行，并将输出结果交由PrintMessageAction类处理。

该部分中实现的实际功能调用类，主要包括以下操作实例：

1. 数据文件下行传输：包括执行传输（RetrieveFilesCliAction），传输实例的状态查询（GetCurrentTransfersCliAction/GetCurrentTransferCliAction），传输进度查询（GetFilePercentTransferredCliAction）
2. Product/ProductTypes相关查询：包括获取Product/ProductTypes获取（GetProductTypeByNameCliAction、AbstractGetProductCliAction及其实现类），元数据查询（AbstractQueryCliAction及其实现类，DumpMetadataCliAction），ProductTypes pages查询（GetPrevPageCliAction，GetNextPageCliAction，GetLastPageCliAction，GetFirstPageCliAction），productTypes数量查询（GetNumProductsCliAction）
3. 新增操作：新增ProductTypes（AddProductTypeCliAction）
4. 删除操作：删除Product记录（AbstractDeleteProductCliAction及其实现类）

其中部分实际功能调用类为抽象类，其实现类进一步细分了其功能，例如AbstractQueryCliAction的继承类分别实现了按照类SQL形式查询和类Lucene形式查询两种方式。

该部分功能的主要类图3.5所示。

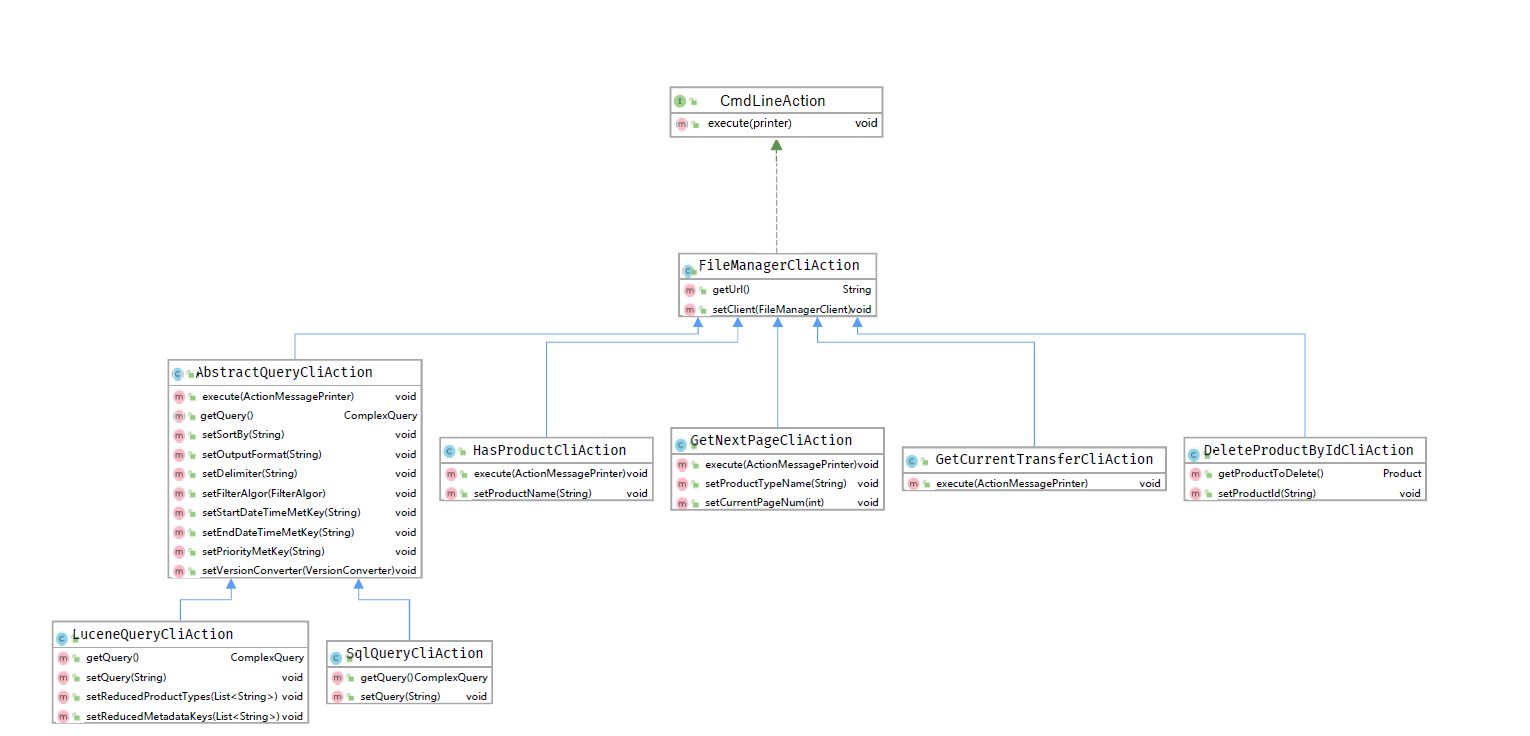


图 3.5数据查询模块主要类图

## 数据传输模块

由需求可知，数据传输模需要支持不同情况下的数据传输方法。因此，该模块首先定义一个DataTransfer的接口，并根据需求的定义分别规定了四个需要实现的方法：

1. 对数据传输实例设置传输目标路径(setFileManagerUrl())
2. 将指定的管理记录对应的数据文件上行传输至Flie Manager存储管理位置下(transferProduct())
3. 将File Manager某个/某些管理对象下行传输至某个文件系统路径下(retrieveProduct())
4. 删除File Manager指定的管理对象记录及文件数据(deleteProduct())

通过该接口分别实现了四个DataTransferer类，LocalDataTransferer负责处理与本地机器、挂载NFS的磁盘，以及挂载的本地文件仓库之间的传输操作；RemoteDataTransferer负责处理与能通过XML-RPC协议进行传输的远程文件管理器进行的传输操作，并且是以File Manager客户端的方式进行传输；S3DataTransferer是提供了与Amazon S3云服务之间进行上下行传输的功能；而InPlaceDataTransferer是一个特殊的实现，它不进行传输，而将管理对象仍然保留在原地。另外TransferStatusTracker则是一个使用内存中的ConcurrentHashMap来跟踪数据传输状态信息的辅助类。

该部分功能的主要类图3.6所示。

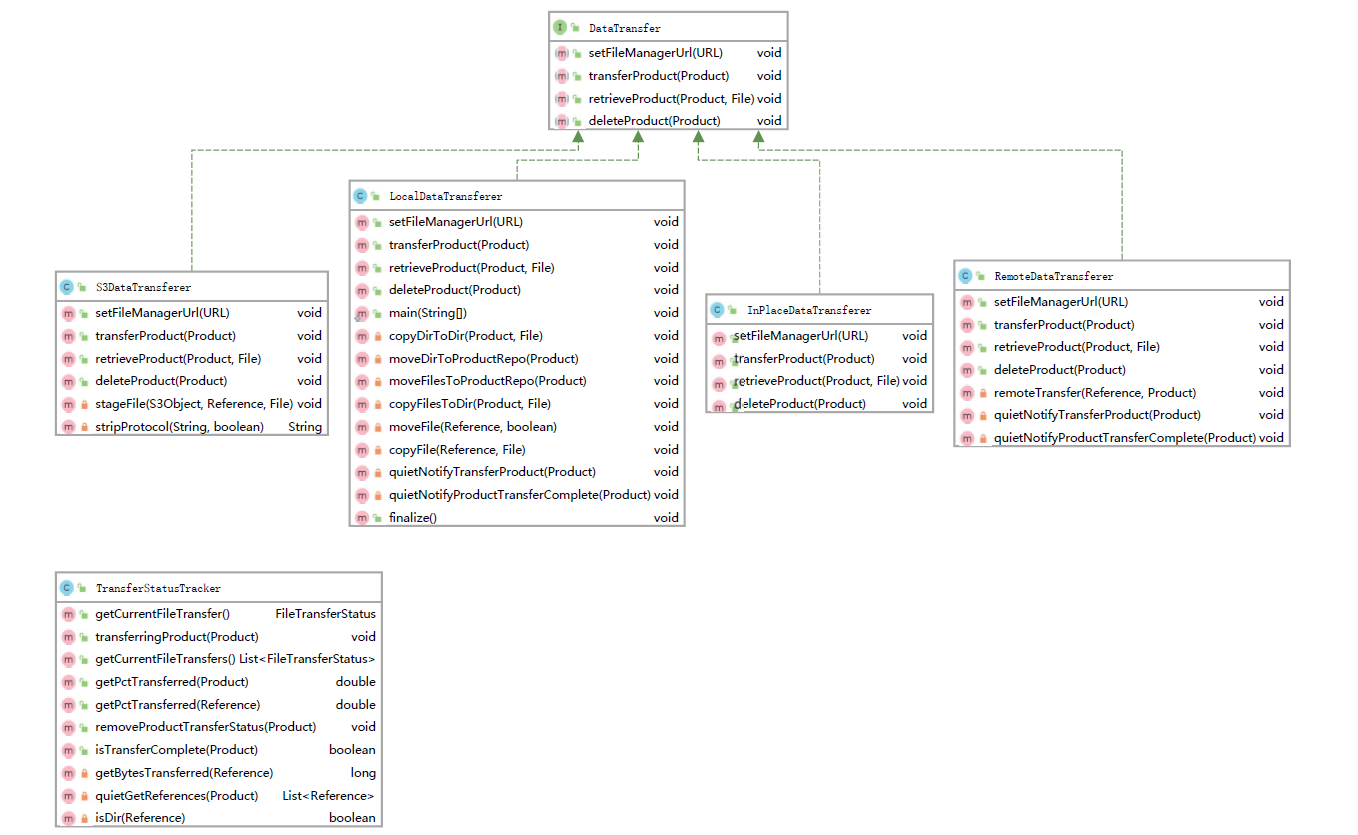


图 3.6数据传输模块的主要类图

## 配置管理模块

由需求分析可知，配置管理模块主要需要提供记录、修改、加载和监听功能。其中，记录功能借助若干指定的xml文件，包括product-types.xml、elements.xml、product-type-element-map.xml等，这些文件中记录了目录的分组逻辑、文件采集项、目录的各种属性以及目录中记录条目的分组逻辑等。用户可以直接修改相应的xml文件来指定各项配置信息，以实现修改的功能。

对于配置信息的加载与监听功能，则都是借助一个全局范围的类ConfigurationManager类实现的，该类不仅仅负责OODT中File Manager模块的配置加载与监控，同时也负责Workflow Manager模块和Resource Manager模块的配置管理。配置管理模块相关的类图如图3.7所示。

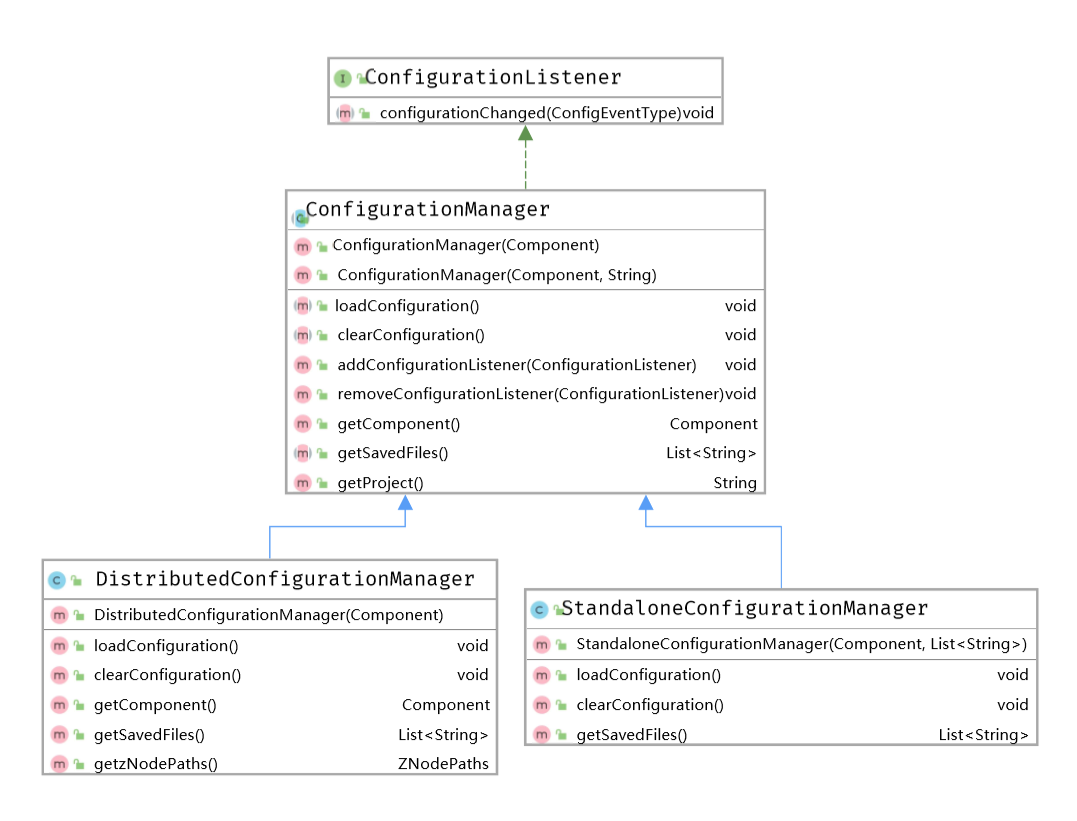


图 3.7配置管理模块的主要类图

上图中，ConfigurationListener接口主要定义了获取相关配置信息改动情况的方法，用于监听运行时指定配置信息的改动情况；ConfigurationManager类是配置管理器类，定义了配置加载、配置清楚、配置监听、配置信息获取等功能函数；对于分布式与单机两种运行环境，又有各自相应的子类DistributedConfigurationManager类和StandaloneConfigurationManager类，这两个子类继承了父类的主要功能函数，同时重新封装了必要的功能函数，以适应分布式与单机两种不同的运行环境。

## 异常处理模块

异常处理部分都继承自一个父类Exception，拥有抛出异常被上层捕获、打印异常信息等功能。每个异常类都重载了Exception的构造方法，包括如下参数：Exception(String message)、Exception(Throwable cause)和Exception(String message,Throwable cause)。常见的异常类包括：CatalogException，DataTransferException，IngestException，QueryFormulationException,RepositoryManagerException和FileManagerException。其中CatalogException用于处理与数据记录相关的异常，DataTransferException用于处理与数据传输相关的异常，IngestException用于处理与数据采集相关的异常，QueryFormulationException用于处理目录查询相关的异常（比如布尔表达式构造出错），RepositoryManagerException用于处理数据存储相关的异常，而FileManagerException没有明显的使用场景，在代码中仅调用三次。

该部分功能的主要类图3.8所示。

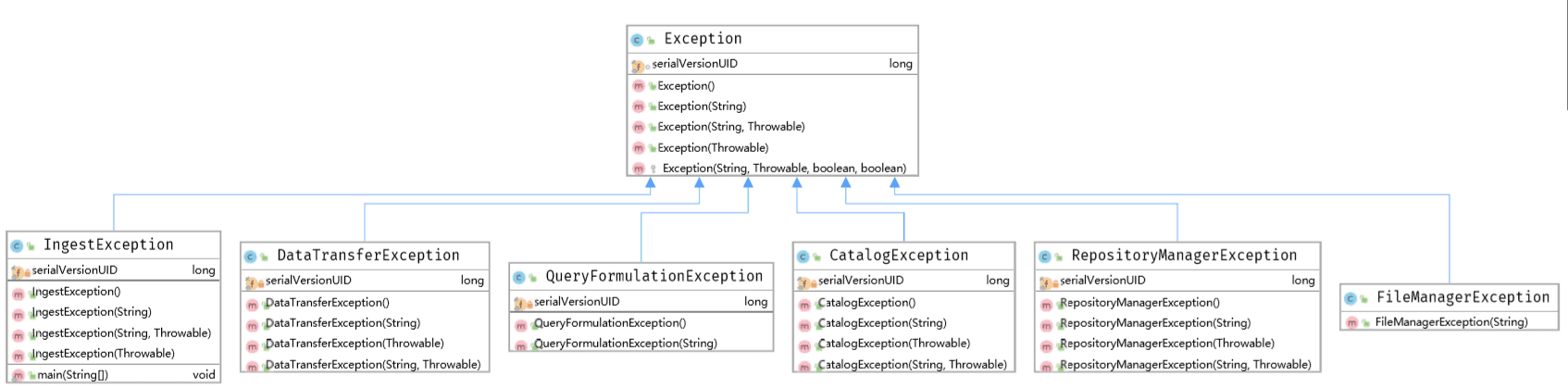


图 3.8异常处理模块的主要类图

## 设计模式分析

软件设计模式可以使人们更加简单方便的复用成功的设计和体系结构，对于一个大项目或框架设计，使用设计模式组织代码是很好的选择，因此，为了对本开源项目的代码中关联组合关系等进一步了解，以及对设计经验进行总结，我们对项目采用的设计模式进行了分析整理，总结出以下四个设计模式。

### 工厂模式

数据传输模块中，采用了工厂模式对DataTransfer的各个类的对象创建进行管理。创建了DataTransferFactory类作为抽象工厂，DataTransfer类作为抽象产品，xxxInDataTransferFactory作为具体工厂，xxxInDataTransferer作为具体产品，从而分离了数据传输对象的创建与使用过程。

同样的，数据记录与管理模块的Catalog及其相关类，数据采集模块的Cache相关类，数据存储模块的Versioner相关类等等也都采取了同样的工厂模式，其基本组织方法是类似的，这里不做赘述。

### 策略模式

在数据查询模块中，对于文件管理中不同功能的调用采用了策略模式，CmdLineAction作为抽象策略类，定义了一个公共接口 execute() ，而不同action实现不同的算法。比如GetCurrentTransferCliAction，GetNextPageCliAction等类都作为具体策略，而对命令行解析完成后，持有一个策略类的引用，然后调用相关的策略。

### 迭代器模式

由于Java语言集成了对Iterator的支持，所以迭代器模式的使用在该项目中是比较常见的，除此以外，客户端命令行解析部分代码还实现了一个自定义迭代器CmdLineIterable，它将命令行提供的参数都封装在此迭代器中，分离了聚合对象与其遍历行为。

### 享元模式

对于File Manager客户端中的XmlRpcFileManagerClient类，是通过url获取其客户端对象的，对于相同的url所获取到的对象应该是相同的，对于不同的功能执行模块，其client对象是共用的，这就是享元模式，其主要优点是相同对象只要保存一份，这降低了系统中对象的数量，从而降低了系统中细粒度对象给内存带来的压力。

# 设计特色分析

本章首先分析了File Manager模块比较突出的几点设计特色，包括目录属性的继承机制、C/S模式，然后分析了该模块的不足之处。

## 目录属性的继承机制

在File Manager模块中，目录类似于数据库中的表，记录着采集到的文件数据以及文件的各种附件信息，包括文件名、文件大小、标题、文件原始路径、文件最终路径等属性，同时，和数据库一样，File Manager模块也允许同时维护多个目录。而这些目录的很多属性大都是相似的，如果每个目录都需要由用户去重新定义全部的属性，这无疑是一件重复性很大并且很麻烦的工作。

因此，File Manager模块在设计目录时添加了继承机制，允许目录像类一样拥有继承关系。这时，对于多个相似的目录，只需要定义一个共同的父目录和共有属性，然后再定义各自独立的子目录和特有属性即可。

## C/S模式

File Manager模块采用的是C/S模式，即客户端/服务器模式。其中，服务器只负责数据目录的管理与维护，并提供必要的访问接口；客户端则负责具体的数据采集工作，同时可以向服务器发送各种请求以实现对目录中已记录数据的查询、获取、修改与删除，并且对查询到的数据进行各种复杂的逻辑处理。

这种C/S模式，完美的实现了对分布式集群的支持。用户可以指定集群中的若干硬盘资源充足的主机充当系统的存储核心，负责数据目录的管理与维护以及数据的存储；然后指定其余计算资源充足的主机充当计算核心，负责执行对数据进行的复杂处理逻辑。

## 不足之处

作为一个支持分布式的文件数据归档管理模块，File Manager模块仍有以下几点可改进的方面：

1. 冗余存储。File Manager模块对于采集到的文件数据，会统一写入到指定的磁盘位置中，但是并没有进行冗余存储，当磁盘损坏时很容易造成数据的丢失。
2. 集群资源的统一管理。File Manager模块的目录信息存储位置以及文件数据的存储位置都是由用户指定，无法根据整个集群的存储资源情况进行自动分配，这样很容易因为某台主机磁盘占尽导致系统无法正常运行。
3. 部分模块封装混乱。FileManagerException处于顶级目录下的 exceptions package中，但主要异常类却被放置在 structs.excepetions pacakage中，二者的不一致性造成代码异常类管理的相对混乱。

# 组内分工情况

本组各成员在本次分析作业中的任务分配情况如表5.1所示。

|  |  |
| --- | --- |
| 组员 | 分工 |
| 陈鸿超 | 1. 项目调研 2. 工具调研 3. 代码分析 4. 类图绘制 5. 项目简介撰写 6. 数据采集模块与配置管理模块的需求分析与结构设计撰写 7. 设计特色分析撰写 |
| 李铎坤 | 1. 项目调研 2. 工具调研 3. 代码分析 4. 类图绘制 5. 项目简介撰写 6. 数据查询模块与数据传输模块的需求分析与结构设计撰写 7. 设计模式分析与撰写 |
| 刘颖 | 1. 项目调研 2. 工具调研 3. 代码分析 4. 类图绘制 5. 项目简介撰写 6. 数据记录与管理模块、数据存储模块、异常处理模块的需求分析与结构设计撰写 7. 质量属性分析撰写 |

# 参考文献

1. <https://github.com/apache/oodt>
2. <https://cwiki.apache.org/confluence/display/OODT>
3. <http://c.biancheng.net/view/1317.html>
4. 程杰.大话设计模式[M].北京:清华大学出版社,2007.
5. 覃征.软件体系结构[M].第四版.北京:清华大学出版社,2018.23-65.
6. 六个常见质量属性场景分析[EB/OL].https://www.cnblogs.com/kangzhijia/p/8575298.html

# 附件：体系结构的分解过程

