**基于Lucene的分析与应用**

**项目计划**

Version 1.02

小组成员：

刘宏宇

滕延林

顾泽鹏

杨帆

**版本变更记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 变更时间 | 修改人 | 审核人 |
| 1.0 | 20160316 | 滕延林 | 刘宏宇 顾泽鹏 杨帆 |
| 1.01 | 20160320 | 滕延林 | 刘宏宇 顾泽鹏 杨帆 |
| 1.02 | 20160403 | 刘宏宇 | 滕延林 顾泽鹏 杨帆 |

**版本变更说明表**

|  |  |  |
| --- | --- | --- |
| 变更原因 | 变更位置 | 变更内容概述 |
| 外因（老师点评） | 第四章 | 将非功能性需求分离出来作为与系统功能需求对应的独立部分 |
| 外因（老师点评）：  需求文档中应该包含系统运行数据的需求 | 第五章 | 添加了系统运行的数据需求 |
| 外因（老师点评）：  需求文档中应该包含系统运行需求 | 第六章 | 添加了系统的运行需求，包括硬件、软件以及用户界面的需求 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[1前言 4](#_Toc447478993)

[1.1目的 4](#_Toc447478994)

[1.2系统概述 4](#_Toc447478995)

[1.3文档概述 5](#_Toc447478996)

[1.4术语和缩略语 5](#_Toc447478997)

[2引用文档 6](#_Toc447478998)

[3功能需求 6](#_Toc447478999)

[3.1软件功能分析 6](#_Toc447479000)

[3.2需求识别 7](#_Toc447479001)

[3.2.1索引生成 8](#_Toc447479002)

[3.2.2索引段的合并 8](#_Toc447479003)

[3.2.3检索过程 9](#_Toc447479004)

[3.2.4分词器 Analyzer 10](#_Toc447479005)

[3.3RUCM模型 11](#_Toc447479006)

[3.3.1生成索引 11](#_Toc447479007)

[3.3.2创建IndexWriter对象 12](#_Toc447479008)

[3.3.3创建Document对象 12](#_Toc447479009)

[3.3.4将文档写入IndexWriter 13](#_Toc447479010)

[3.3.5将文档加入DocumentWriter 13](#_Toc447479011)

[3.3.6关闭IndexWriter对象 14](#_Toc447479012)

[3.3.7缓存管理 14](#_Toc447479013)

[3.3.8索引段的合并模块 15](#_Toc447479014)

[3.3.9检索过程 16](#_Toc447479015)

[3.3.10分词器 16](#_Toc447479016)

[4非功能性需求分析 18](#_Toc447479017)

[4.1兼容性 18](#_Toc447479018)

[4.2可修改性 18](#_Toc447479019)

[4.3高效性 18](#_Toc447479020)

[5.数据需求 19](#_Toc447479021)

[5.1数据格式 19](#_Toc447479022)

[5.2数据来源 19](#_Toc447479023)

[5.3数据规模 19](#_Toc447479024)

[6运行需求 20](#_Toc447479025)

[6.1硬件要求 20](#_Toc447479026)

[6.2软件要求 20](#_Toc447479027)

[6.3用户界面需求 20](#_Toc447479028)

# 1前言

# 1.1目的

为了便于协调组内成员进行后期的工作，对项目进行跟踪和监控，对任务的进度进行安排与调控，故对后期工作进行计划。提出需求，指导后续工作。

# 1.2系统概述

Lucene 是一个基于 Java 的全文信息检索工具包，它不是一个完整的搜索应用程序，而是为你的应用程序提供索引和搜索功能。Lucene 目前是 Apache Jakarta 家族中的一个开源项目。也是目前最为流行的基于 Java 开源全文检索工具包。

目前已经有很多应用程序的搜索功能是基于 Lucene 的，比如 Eclipse 的帮助系统的搜索功能。Lucene 能够为文本类型的数据建立索引，所以你只要能把你要索引的数据格式转化的文本的，Lucene 就能对你的文档进行索引和搜索。比如你要对一些 HTML 文档，PDF 文档进行索引的话你就首先需要把 HTML 文档和 PDF 文档转化成文本格式的，然后将转化后的内容交给 Lucene 进行索引，然后把创建好的索引文件保存到磁盘或者内存中，最后根据用户输入的查询条件在索引文件上进行查询。不指定要索引的文档的格式也使 Lucene 能够几乎适用于所有的搜索应用程序。

图 1 表示了搜索应用程序和 Lucene 之间的关系，也反映了利用 Lucene 构建搜索应用程序的流程：

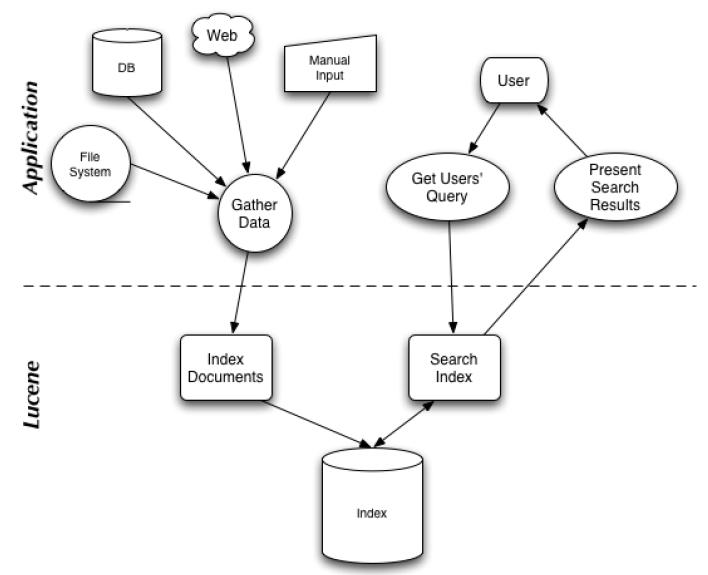


图 1搜索应用程序和 Lucene 之间的关系

# 1.3文档概述

文档用途：本文档主要是介绍Lucene系统需求及规格说明。 主要内容：

⮚以用例图、状态图的形式给出 Lucene系统功能需求的分解结构，并对用例模型中的参与者和用例进行详细的描述，其中主要包括软件系统的用 例模型、系统的核心流程等；

⮚使用 RUCM 模型对功能需求进行建模；

⮚描述了与此次系统实施相关的硬件环境的一些要求；

⮚描述了与此系统实施相关的软件环境的要求；

# 1.4术语和缩略语

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 术语 | 英文 | 说明 |
| 1 | UCM | UCM | 用例建模 |
| 2 | RUCM | RUCM | 限制性用例模型 |
| 3 | 索引 | Index | 在 Lucene 中一个索引是放在一个文件夹中 |
| 4 | 段 | Segment | 一个索引可以包含多个段，段与段之间是独立的，添加新文档可以生成新的段，不同的段可以合并。 |
| 5 | 文档 | Document | 文档是我们建索引的基本单位，不同的文档是保存在不同的段中的，一个段可以包含多篇文档。 |
| 6 | 域 | Field | 一篇文档包含不同类型的信息，可以分开索引，比如标题，时间，正文，作者等，都可以保存在不同的域里。 |
| 7 | 前缀后缀规则 | Prefix+Suffix | 所谓前缀后缀规则，即当某个词和前一个词有共同的前缀的时候，后面的词仅仅保存前缀在词中的偏移，以及除前缀以外的字符串(称为后缀)。 |
| 8 | 差值规则 | Delta | 所谓差值规则(Delta)就是先后保存两个整数的时候，后面的整数仅仅保存和前面整数的差即可。 |
| 9 | 词元 | Token | 将文档分词，并且去除标点符号和停词后，得到的一个个单独的单词。 |
| 10 | 词 | Term | 经Token经过过滤后，得到的小写、词根形式的单词。 |
| 11 | 停词 | Stop word | 一种语言中最普通的的一些单词，由于没有特殊的意义，因而大多数情况下不能成为搜索的关键词，例如“this”， “a”， “the”等。 |
| 12 | 分词组件 | Tokenizer | 将文档文本进行分词的组件。 |
| 13 | 语言处理组件 | TokenFilter | 将Token串进行过滤的组件。 |

# 2引用文档

《Lucene原理与代码分析完整版》

# 3功能需求

# 3.1软件功能分析

Lucene 软件包的发布形式是一个 JAR 文件，下面我们分析一下这个 JAR 文件里面的主要的 JAVA 包，使读者对之有个初步的了解。

Package: org.apache.lucene.document这个包提供了一些为封装要索引的文档所需要的类，比如 Document, Field。这样，每一个文档最终被封装成了一个 Document 对象。

Package: org.apache.lucene.analysis这个包主要功能是对文档进行分词，因为文档在建立索引之前必须要进行分词，所以这个包的作用可以看成是为建立索引做准备工作。

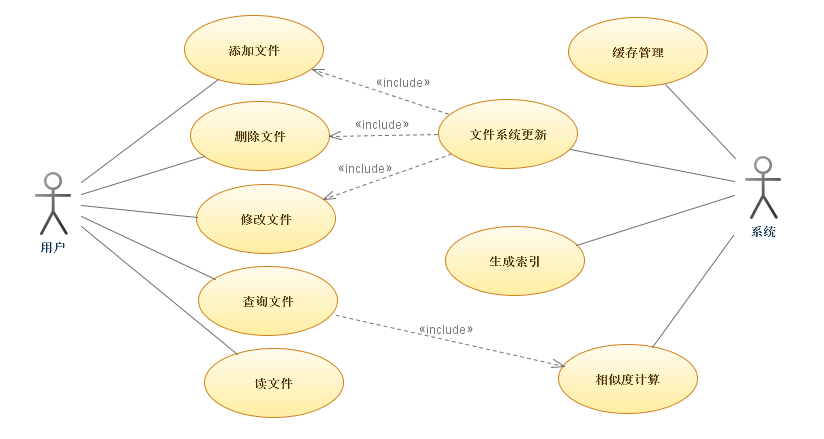
Package: org.apache.lucene.index这个包提供了一些类来协助创建索引以及对创建好的索引进行更新。这里面有两个基础的类：IndexWriter 和 IndexReader，其中 IndexWriter 是用来创建索引并添加文档到索引中的，IndexReader 是用来删除索引中的文档的。

Package: org.apache.lucene.search这个包提供了对在建立好的索引上进行搜索所需要的类。比如 IndexSearcher 和 Hits, IndexSearcher 定义了在指定的索引上进行搜索的方法，Hits 用来保存搜索得到的结果。

# 3.2需求识别

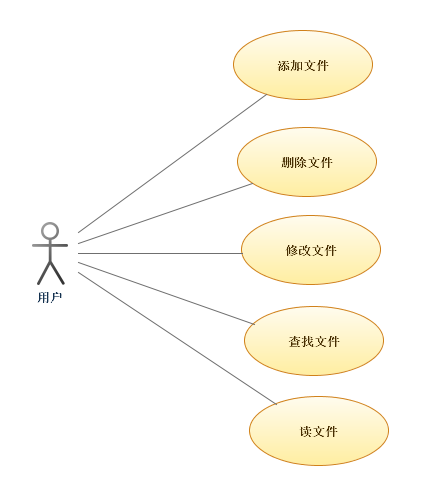
将业务需求分解为功能性需求，同时考虑到非功能性需求，最终得到的Lucene的用例图如下图所示。

# 3.2.1用户与系统交互



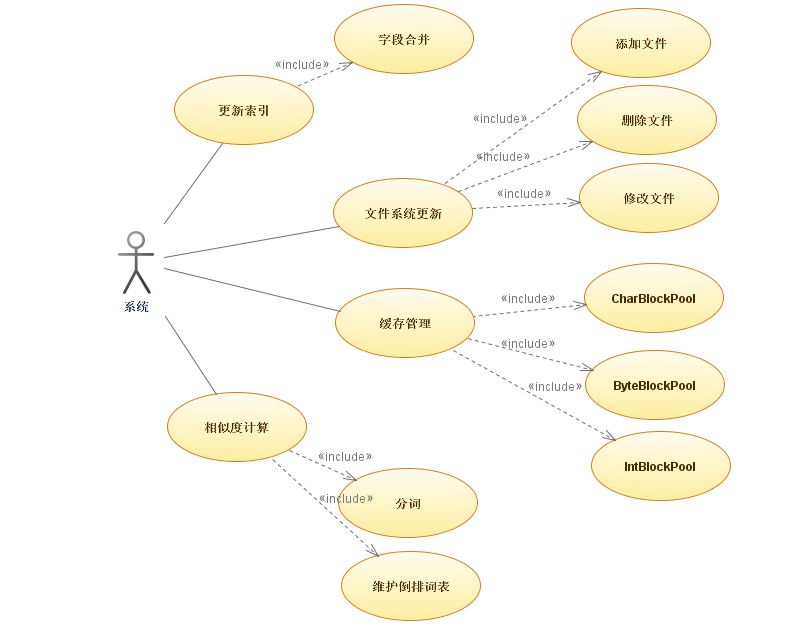
用户对于文件系统有读/写操作，读操作包括对于文件的读取与查询。读文件时需要知道文件的路径，查找时，可以输入关键字，时间，作者，标题等信息进行查找。写操作包括文件的添加，删除与修改。系统可以进行文件更新操作，当用户对于文件系统进行写操作时，系统会自动更新文件，同时修改相应的索引。但用户查询文件时，系统会进行相似度计算，从文件系统中检索，给出合理的排序，反馈给用户。缓存管理的任务贯穿系统运行的始终。

# 3.2.2用户操作文件系统



用户对于文件系统拥有增、删、改、查的权限。

# 3.2.3系统生成索引



系统的主要任务是索引更新，文件系统更新，缓存管理好相似度计算。在索引更新中，还有涉及到字段合并的操作。文件系统更新对于与用户对于文件系统的修改，包括文件的添加、修改和删除。缓存管理是为了提升系统的效率并且使系统的高负载情况下不会崩溃，增强系统鲁棒性。对于不同类型的数据有不同的策略，主要包括char,byte,int三种。相似度计算用于系统检索文件。当用户提出检索请求时，系统进行该操作。相似度计算首先要进行分词，并维护一个倒排词表来比较文件的相似度，给出一个合理的排序，越符合用户需求的文件会排在越前面。

# 3.2.4索引段的合并

索引段的合并模块(merge\_module)

从索引段队列中选择待合并的段，根据具体情况确定合并策略，并合理地调度任务。

合并过程最重要的是三部分：

1. 一是选择哪些段应该参与合并，这一步由Segment selection 来决定

2. 二是选择什么样的合并策略能够取得最优性能，这一步由MergePolicy 来决

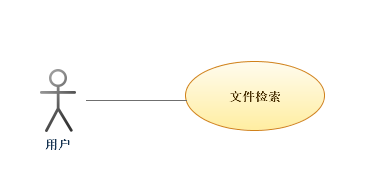
3. 三是调度合并段的过程，这一步由MergeScheduler 来执行

# 3.2.5检索过程

检索过程：

索引的过程主要分为以下几个步骤

首先用户输入要检索的词语，比如 “apple not iphone”，之后打开索引文件，并将索引文件的相关信息读入并对查询分析器进行初始化。然后根据用户输入的词语，分析其中的逻辑，该例子就是要检索出不包括iphone的apple。根据分析结果，进行检索。之后对检索的结果进行打分排序，返回最终的检索结果。



# 3.2.6分词器 Analyzer

被索引的文档在建立索引之前，首先要经过分词器的处理。其目的是，将文档分解为一系列的单词，从而更方便的建立索引。分词器主要分为两个部分，即分词组件和语言处理组件。其用例图如下所示。

分词组件的主要功能是将输入的文本分割为一个个词元组，即一系列Token。在这个过程中，同时还要去除标点符号和停词。停词是一种语言中最普通的的一些单词，由于没有特殊的意义，因而大多数情况下不能成为搜索的关键词，例如“this”， “a”， “the”等。在经过分词组件的处理后，文本中大部分的没有意义的信息被去除了。

# 3.3RUCM模型

RUCM 即限制性用例建模。 它的目标是：

1．使 UCMs 更加可理解并且更精确。

2．从 UCMs 自动生成分析模型。

RUCM 有以下两部分组成：

1．一个用于系统组织 UCSs 的用例模板。

2．限制用户写 UCSs 的一系列规则。

通过 RUCM 模型能够对用例进行规范的描述，接下来将使用 RUCM 模型描述图 1 中的用例。

# 3.3.1生成索引



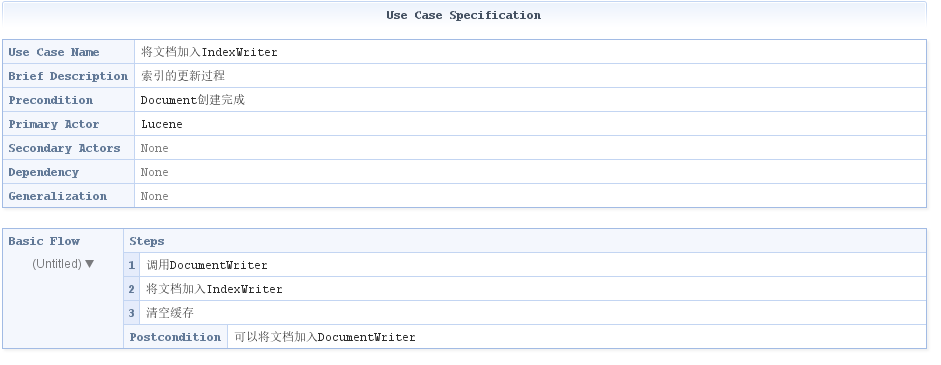
# 3.3.2创建IndexWriter对象



# 3.3.3创建Document对象



# 3.3.4将文档写入IndexWriter



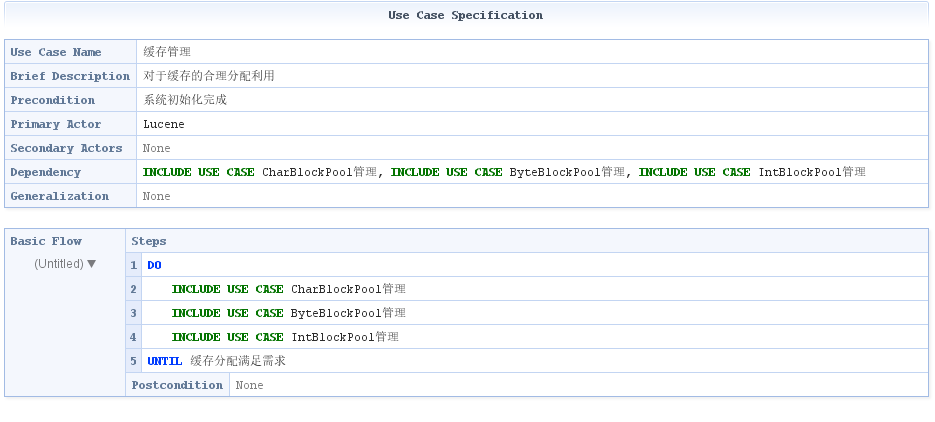
# 3.3.5将文档加入DocumentWriter



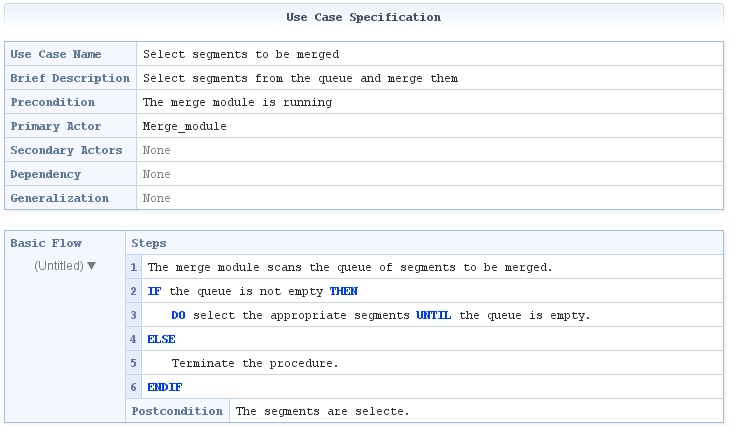
# 3.3.6关闭IndexWriter对象

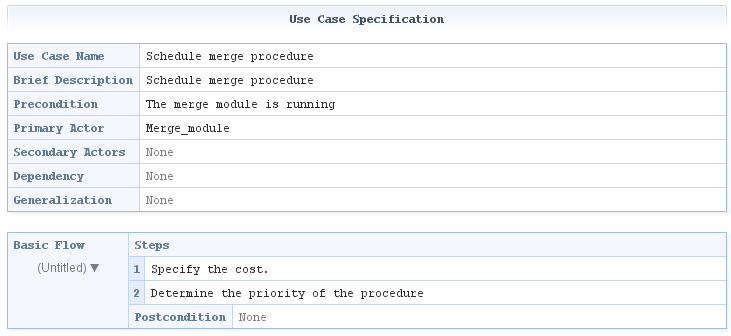


# 3.3.7缓存管理



# 3.3.8索引段的合并模块

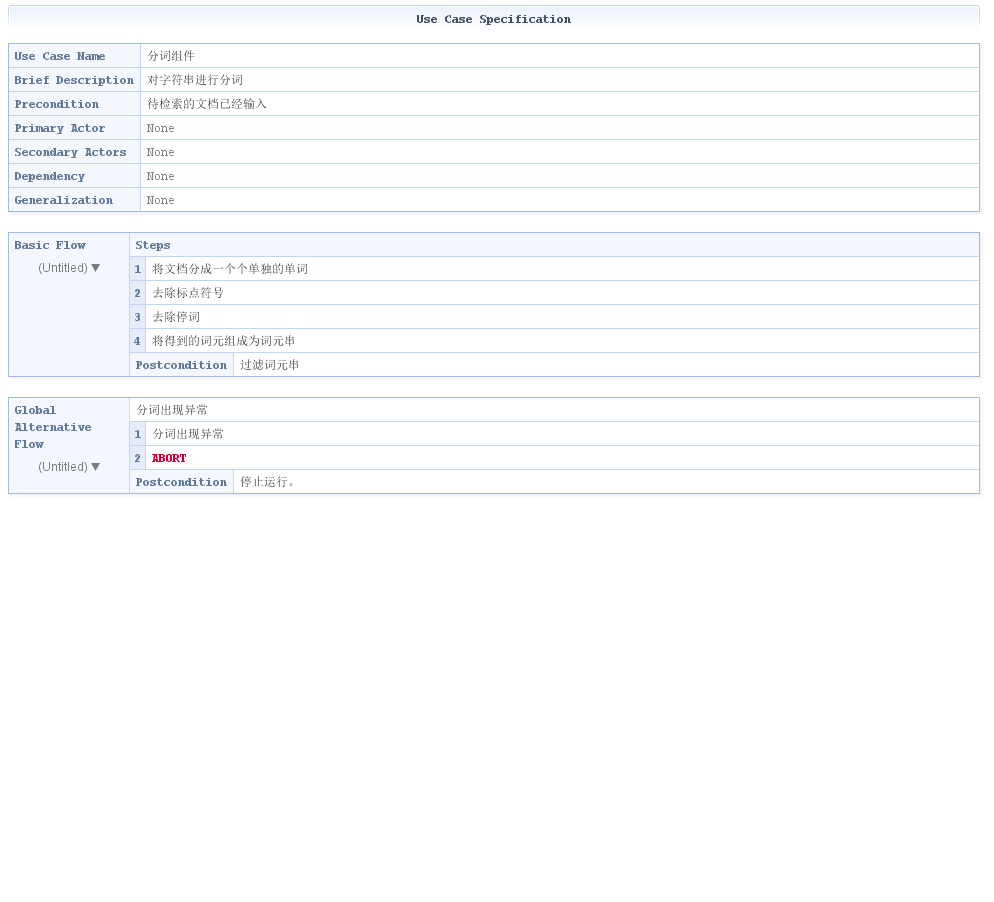




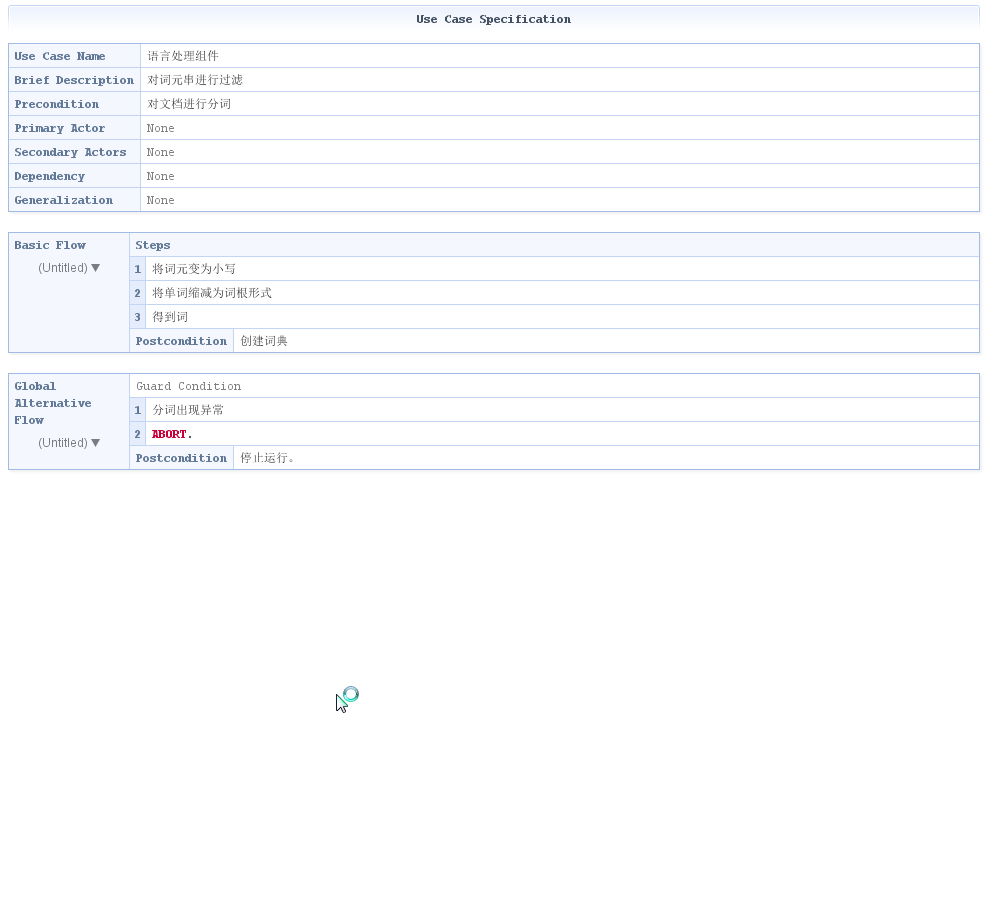
# 3.3.9检索过程



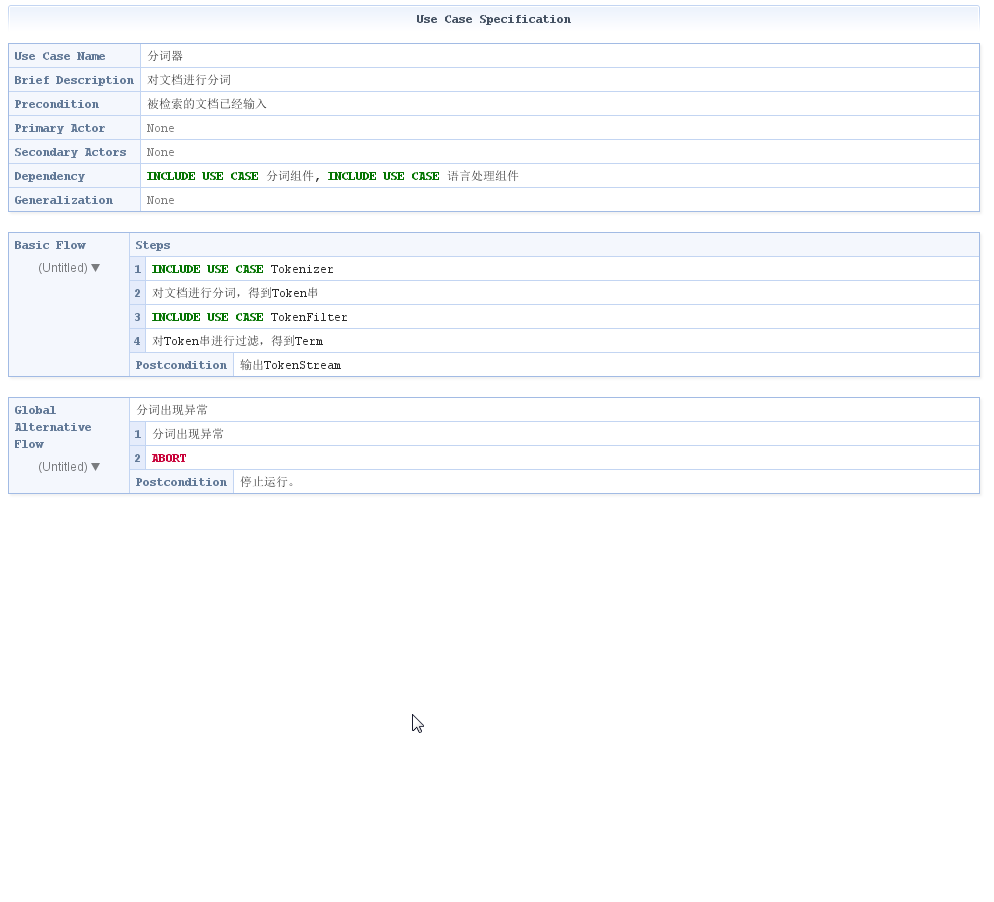
# 3.3.10分词器



经过分词处理后的词元串流入语言处理组件中。语言处理组件对得到的词元做一些同语言相关的处理。对于英语，语言处理组件一般进行两项处理：将词元变为小写；将单词缩减或转变为词根形式。例如，将”cars”缩减为”car”，将”drove”转化为”drive”。语言处理组件的RUCM模型如下图所示。



分词器整体的RUCM模型如下图所示。



以上分词器的工作过程为通用的对于英文文本的处理过程。对于中文则需要更复杂的处理方式，例如，需要一定的算法先进行句子和词组的划分，再对词组和词语进行缩减与转化，处理流程因算法而异。

# 4非功能性需求分析

# 4.1兼容性

作为一个跨平台的全文搜索引擎，系统应该具有强大的兼容性：

1. 操作系统兼容性

理想的软件应该具有与平台无关性，因此基于Lucene开发的搜索程序应该具有运行于不同操作系统的能力，需要定义独立于平台的索引格式，还要考虑前端和后端操作系统的可选择性。

1. 异构数据库兼容性

搜索引擎索引的数据需要数据库系统的支持，因此程序需要考虑其对不同数据库平台的支持能力，兼容异构数据库，使得不同数据库能够共享建立的索引文件。

1. 新旧数据转换

对于软件升级后可能定义的新的数据格式或者文件格式，提供新旧数据转换的功能。提供对原来格式的支持及更新，使得原来的用户记录能够被集成，在新的格式下依然可用。这还涉及了转换过程中数据的完整性与正确性的验证问题。

# 4.2可修改性

作为一种检索系统框架，Lucene并不直接提供系统的实现，而仅仅是系统框架而已。为了进行高效的开发。要求程序要具备一种简明、方便的构架与函数接口来方便用户的使用。此外，为了使运行更加高效，需要不断学习，引入新的技术、算法或辅助措施，因此程序必须具备很好的可修改性（ modifiability）。

# 4.3高效性

本程序作为检索系统，需要支持对大规模数据的索引及搜索，因此需要具有很高的效率。这主要体现在两个方面：

1. 高效实现大规模数据实时索引入库

作为一个全文检索系统，如果入库数据占据了大量时间，那么必然影响了检索系统的实时性。因此程序需要高效完成大规模数据的索引入库过程。

1. 迅速响应查询条件并返回结果

衡量搜索引擎性能的一个重要指标就是响应时间，因此程序需要能够在很短的时间内响应查询请求并返回检索结果

# 5.数据需求

# 5.1数据格式

多种类型文本文件，包括txt,doc,pdf,html等

# 5.2数据来源

* 互联网上下载
* 已有文本
* 自主构造

互联网下载的文本用于丰富语料库，作为检索源。越大的文本库越能够检验系统的性能，包括准确率，排序效果，时间等；已有的文本是检索的目标，根据已有的文本进行优化，可以构造更加适用于该文本的检索系统；自主构造指构造一些特殊的文本，用于检索一些系统的边缘情况，可以更好测试系统的稳定性，。

# 5.3数据规模

10M左右的测试样本，放在1G的文件系统内。

# 6运行需求

# 6.1硬件要求

* CPU:E7500以上
* 内存：4G内存及以上
* 硬盘：100G硬盘及以上

# 6.2软件要求

* 操作系统：Windows7版本及以上
* 编译环境：JRE6.0版本及以上。

# 6.3用户界面需求

简单实用，容易操作