(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105320525 A
(43)申请公布日 2016.02.10

(21)申请号 201510916069.3

(22)申请日 2015.12.09

(71)申请人 扬州大学
地址 225009 江苏省扬州市大学南路88号

(72)发明人 孙小兵 唐莉 李斌 李云

(74)专利代理机构 南京中新达专利代理有限公司 32226
代理人 孙鸥 朱杰

(51)Int.Cl.
G06F 9/44(2006.01)

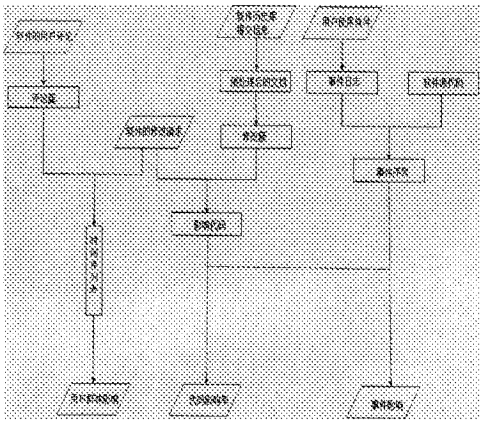
权利要求书2页 说明书5页 附图4页

(54)发明名称

一种面向移动应用软件的修改影响分析方法

(57)摘要

本发明提出了一种面向移动应用软件的修改影响分析方法。本发明将软件修改历史库中的提交信息进行聚类，找到与修改请求相关的修改代码，利用调用继承等依赖关系计算代码层次上的影响集，并根据修改影响的Activity找到对应的事件序列，预估对软件功能的影响，还挖掘与某次修改请求的相关评论，得到评论的时间序列表，来分析不同时间段受影响的用户群。本发明克服了软件修改影响分析技术对移动应用软件没有针对性，不能满足移动应用软件快速响应，快速发布，快速反馈，快速维护等缺陷。本发明从用户群体分析，可以挖掘与某次修改请求的相关评论，得到评论的时间序列表，来分析不同时间段受影响的用户群。



CN 105320525 A

1.一种面向移动应用软件的修改影响分析方法，给定某个移动应用 APP 和修改请求，其特征在于如下步骤：

1) 对该移动应用 APP 的软件历史库中的提交消息进行托肯化，去停用词，词干化三步操作，生成去除噪声的预处理后的文档；

2) 将预处理后的文档使用 K-Means 算法聚类，得到 k 个修改簇，根据修改请求匹配其修改簇，找到与其相关的修改文档，在文档里找到相关的修改代码；

3) 在修改代码中找到修改方法，修改类，利用调用和继承依赖关系计算代码层次的影响集；先构建一个包含影响集中所有方法的调用以及所有类的继承关系的传播图，然后通过数学公式

$$I(M)=\sum_{Mj\in S}\frac{1}{D(Mj)}$$

其中，Mj 是传播图中的方法，D(Mj) 是影响方法到修改方法的最短距离，计算传播图中被影响节点的可能性的总和，以量化整体的修改影响；

4) 从用户使用软件时生成的事件日志以及软件源代码中提取事件序列，最终以元组 e_i ＝＜ Activity，Window，GUI-Component，Action ＞的形式表示，一个元组表示一个事件；

5) 将修改代码在源代码中找到其所在的 Activity，与事件序列中的 Activity 相匹配，得到修改对应的事件序列，分析其可能影响软件的功能；

6) 从移动应用市场利用网页爬虫技术收集 APP 的原数据，提取用户评论，并进行预处理；

7) 将用户评论 K-Means 算法进行聚类，得到 k 个评论簇，然后与软件修改请求匹配，找到与修改相关评论；

8) 对修改相关评论按时间间隔 T 进行划分，T 可以为任意的时间间隔，将 T 设置为月或周；当 T 为月时，表示按月进行划分；当 T 为周时，表示按周进行划分；然后生成时间序列文档，记录一共有多少个时间段以及每个时间段内的提交消息数量，时间序列文档的格式为：第一行记录一共有多少个时间段，其数目记为 N；第二行到第 N+1 行分别为每个 时间段内的提交消息数量，根据时间序列表分析不同时间段受影响的用户群；

9) 绘制直方图，更清晰方便地分析不同时间段受影响的用户群体。

2.根据权利要求 1 所述的一种面向移动应用软件的修改影响分析方法，其特征在于：

步骤 1) 中对修改特征信息的预处理及步骤 6) 中对评论的预处理的具体步骤如下：

a) 托肯化：去除标点符号、去除数字；

b) 去除停用词、去除英文停用词、去除代词、去除冠词；

c) 词干化：将每个单词转化为它的原型。

3.根据权利要求 1 所述的一种面向移动应用软件的修改影响分析方法，其特征在于：

步骤 2) 中对预处理文档的聚类 K-Means 算法具体步骤如下：

a) 从 n 个文档任意选择 k 个对象作为初始聚类中心；

b) 根据每个聚类对象的均值，计算每个对象与这些中心对象的距离，并根据最小距离重新对相应对象进行划分；

c) 重新计算每个聚类的均值；

d) 计算标准测度函数，当满足一定条件，如函数收敛时，则算法终止；如果条件不满足则回到步骤 b)。

4. 根据权利要求 1 所述的一种面向移动应用软件的修改影响分析方法，其特征在于：

步骤 4) 中从事件日志以及软件源代码中提取事件序列的具体步骤如下：

A. 使用 APK 分析仪从程序源代码中静态挖掘事件序列：

a)APK-AnaIyzer 先使用 dex2jar 和 Procyon 工具反编译，然后通过 srcML 将源文件转换为基于 XML 的表现形式，再依赖 apktooI，从 XML 文件中提取 ID、类型和 GUI 组件的层次结构；

b) 将每个 GUI 组件链接到其各自的行为 / 手势：APK-Ana Iyzer 将固定手势分配给 GUI 组件的类型；对于标准的 Android 组件类型，与预期的手势做链接；对于自定义组件，APK-Ana Iyzer 解析源代码以确定手势处理器和事件侦听器；

c)APK-AnaIyzer 将组件链接到它们出现的活动；

B. 使用数据采集器从事件日志中挖掘事件序列：

a) 收集 Iow-IeyeI 事件日志，在 APP 正常使用过程中，通过 getevent 命令产生单击，长按，滑动以及一些复杂手势的事件日志，并保存在日志存储库中；

b) 使用 Data collector 将事件日志转换成 e_i 标志序列。

一种面向移动应用软件的修改影响分析方法

技术领域

[0001] 本发明提出了一种面向移动应用软件的修改影响分析方法，属于软件维护领域。

背景技术

[0002] 随着软件的大量使用，大部分软件已不是从头开始开发，而是通过维护来不断适应环境的变化，用户的需求，技术的更新，及市场的变化等。软件产品本身的固有属性就是演化性。在软件演化过程中，修改影响分析起着越来越重要的作用。近 20 年来，研究人员在这方面进行了大量的研究，在软件不同工件层次产生了各种修改影响分析技术，如需求和设计层次的静态影响分析，基于历史库的影响分析，动态影响分析等等。

[0003] 然而，如今所存在的软件修改影响分析技术大都是面向传统软件的，对日渐兴起的移动应用软件并没有针对性，不能满足移动应用软件快速响应，快速发布，快速反馈，快速维护等特点，也不能明确修改到底影响了软件的什么功能，给用户带来了什么影响。这些都给软件维护人员带来了很大程度上的修改困难，也造成了时间和资源的浪费，甚至影响用户正常使用软件。

[0004] 所以，针对上述问题，本发明为移动应用软件量身打造了一种修改影响分析方法。本发明从移动应用软件的代码，软件事件，以及用户群体三方面综合考虑修改的影响。对软件事件分析，我们可以清楚知道修改对软件功能的影响。对用户群体的影响分析，更能让我们知道用户的需求，以快速有效地做出修改。这两方面是之前修改影响分析所欠缺的，也是本发明的重要所在。而对代码的分析，前人已有了不少的修改影响技术，特别是从软件历史库中挖掘程序的依赖关系进行分析。例如 Jashki 等人从软件修改历史库中挖掘出各个源文件的修改频率，然后存储于一个矩阵，构造程序文件的聚类，再通过该聚类计算影响集。该影响分析方法有效降低了影响分析的复杂度，且计算影响集的过程也更高效。Canfora 等人也利用挖掘修改历史库信息的方法进行影响分析。他们的影响分析在两个层次进行 :代码层和源文件层。通过计算一个新的修改请求和源代码实体间的相似性来进行影响分析，其中那些相似度最大的程序实体将作为最易受影响的部分。另外，Hattori 等人提出了一种混合的方法计算影响集。他们首先利用普通的修改影响分析方法计算一个初始影响集 ；然后对初始影响集中每个元素计算其与修改集中元素在程序修改历史库中的相关性。但基于移动应用软件代码的开放性，我们对代码分析，可以更快地挖掘到软件的修改历史库，并找到与修改请求相关的代码，分析代码的影响率。

发明内容

[0005] 本发明的目的就在于克服上述缺陷，研制一种面向移动应用软件的修改影响分析方法。

[0006] 本发明的技术方案是 ：

[0007] 一种面向移动应用软件的修改影响分析方法，给定某个移动应用 APP 和修改请求，其主要技术特征在于如下步骤 ：

[0008] 1) 对该移动应用 APP 的软件历史库中的提交消息进行托肯化，去停用词，词干化三步操作，生成去除噪声的预处理后的文档；

[0009] 2) 将预处理后的文档使用 K-Means 算法聚类，得到 k 个修改簇，根据修改请求匹配其修改簇，找到与其相关的修改文档，在文档里找到相关的修改代码；

[0010] 3) 在修改代码中找到修改方法，修改类，利用调用和继承依赖关系计算代码层次的影响集；先构建一个包含影响集中所有方法的调用以及所有类的继承关系的传播图，然后通过数学公式

[0011] $$I(M)=\sum_{Mj\in S}\frac{1}{D(Mj)}$$

[0012] 其中，Mj 是传播图中的方法，D(Mj) 是影响方法到修改方法的最短距离，计算传播图中被影响节点的可能性的总和，以量化整体的修改影响；

[0013] 4) 从用户使用软件时生成的事件日志以及软件源代码中提取事件序列，最终以元组 e_i = <Activity,Window,GUI-Component,Action> 的形式表示，一个元组表示一个事件；

[0014] 5) 将修改代码在源代码中找到其所在的 Activity，与事件序列中的 Activity 相匹配，得到修改对应的事件序列，分析其可能影响软件的功能；

[0015] 6) 从移动应用市场利用网页爬虫技术收集 APP 的原数据，提取用户评论，并进行预处理；

[0016] 7) 将用户评论 K-Means 算法进行聚类，得到 k 个评论簇，然后与软件修改请求匹配，找到与修改相关评论；

[0017] 8) 对修改相关评论按时间间隔 T 进行划分，T 可以为任意的时间间隔，将 T 设置为月或周；当 T 为月时，表示按月进行划分；当 T 为周时，表示按周进行划分；然后生成时间序列文档，记录一共有多少个时间段以及每个时间段内的提交消息数量，时间序列文档的格式为：第一行记录一共有多少个时间段，其数目记为 N；第二行到第 N+1 行分别为每个时间段内的提交消息数量，根据时间序列表分析不同时间段受影响的用户群；

[0018] 9) 绘制直方图，更清晰方便地分析不同时间段受影响的用户群体。

[0019] 其特征在于：

[0020] 步骤 1) 中对修改特征信息的预处理及步骤 6) 中对评论的预处理的具体步骤如下：

[0021] a) 托肯化：去除标点符号、去除数字；

[0022] b) 去除停用词、去除英文停用词、去除代词、去除冠词；

[0023] c) 词干化：将每个单词转化为它的原型；

[0024] 其特征在于：

[0025] 步骤 2) 中对预处理文档的聚类 K-Means 算法具体步骤如下：

[0026] e) 从 n 个文档任意选择 k 个对象作为初始聚类中心；

[0027] f) 根据每个聚类对象的均值，计算每个对象与这些中心对象的距离，并根据最小距离重新对相应对象进行划分；

[0028] g) 重新计算每个聚类的均值；

[0029] h) 计算标准测度函数，当满足一定条件，如函数收敛时，则算法终止；如果条件不满足则回到步骤 b)；

[0030] 其特征在于：

[0031] 步骤 4) 中从事件日志以及软件源代码中提取事件序列的具体步骤如下：

[0032] A. 使用 APK 分析仪从程序源代码中静态挖掘事件序列：

[0033] a)APK-Analyzer 先使用 dex2jar 和 Procyon 工具反编译，然后通过 srcML 将源文件转换为基于 XML 的表现形式，再依赖 apktool，从 XML 文件中提取 ID、类型和 GUI 组件的层次结构；

[0034] b) 将每个 GUI 组件链接到其各自的行为 / 手势 :APK-Ana lyzer 将固定手势分配给 GUI 组件的类型；对于标准的 Android 组件类型，与预期的手势做链接；对于自定义组件，APK-Analyzer 解析源代码以确定手势处理器和事件侦听器；

[0035] c)APK-Analyzer 将组件链接到它们出现的活动；

[0036] B. 使用数据采集器从事件日志中挖掘事件序列：

[0037] a) 收集 low-level 事件日志，在 APP 正常使用过程中，通过 getevent 命令产生单击，长按，滑动以及一些复杂手势的事件日志，并保存在日志存储库中；

[0038] b) 使用 Data collector 将事件日志转换成 e_i 标志序列。

[0039] 本发明的优点和效果在于可以在软件修改历史库中挖掘到与某次修改相关的修改代码，然后利用调用和继承等依赖关系计算代码层次的影响集以及预估修改影响软件事件、修改影响用户群体的过程。具体主要有以下一些优点：

[0040] 本发明从事件方面进行影响分析，目前修改影响分析很多，但从事件方面的还没有。从事件分析，可以了解某次修改请求可能给软件功能带来的影响。

[0041] 本发明从用户群体方面进行影响分析，挖掘与修改请求相关的用户评论，这是传统的修改影响分析方法所欠缺的，也是移动应用软件与传统软件的不同之处。从用户群体分析，可以挖掘与某次修改请求的相关评论，得到评论的时间序列表，来分析不同时间段受影响的用户群。

附图说明

[0042] 图 1——本发明的总体流程示意图。

[0043] 图 2——本发明 HADOOP 软件修改历史库示意图。

[0044] 图 3——本发明某次修改请求涉及到的修改代码示意图。

[0045] 图 4——本发明中修改类的维承示意图。

[0046] 图 5——本发明修改方法的调用关系示意图。

[0047] 图 6——本发明中类和方法整体的影响传播示意图。

[0048] 图 7——本发明从源代码和事件日志中提取的事件序列示意图。

[0049] 图 8——本发明与修改 Activity 对应的事件序列示意图。

[0050] 图 9——本发明百度手机上“拉钩”软件预处理后的用户评论示意图。

[0051] 图 10——本发明时间序列文档 (SEQ) 的格式说明示意图。

[0052] 图 11——本发明时间序列文档 (SEQ) 文档的示例示意图。

[0053] 图 12——本发明中时间——评论数直方示意图。

具体实施方式

[0054] 本发明的技术思路是：

[0055] 将软件修改历史库中的提交信息进行聚类，找到与修改请求相关的修改代码，利用调用继承等依赖关系计算代码层次上的影响集，并根据修改影响的 Activity 找到对应的事件序列，预估对软件功能的影响，还挖掘与某次修改请求的相关评论，得到评论的时间序列表，来分析不同时间段受影响的用户群。

[0056] 下面结合附图对本发明的技术方案进行详细说明：

[0057] 1) 对软件历史库（版本控制库 CVS）中的提交消息 (commit messages) 进行预处理。首先进行托肯化 (tokenization)，比如去除标点符号（例如“@”，“？”），去除数字（例如“1”，“23”）等。其次去除停用词 (stopping)，比如介词“for”，“to”；代词“it”，“he”；冠词“a”，“an”“the”等。最后词干化 (stemming)，将每个单词转化为它的原型，比如“fixed”为过去时，转化为原型“fix”；“moving”为现在进行时，转化为原型“move”。词干化时选用的算法为 Porter stemming algorithm。三步操作之后，产生去除噪声的预处理后的文档。

[0058] 2) 将预处理后的文档使用 K-Means 算法进行聚类，得到 k 个修改簇。然后根据修改请求匹配其修改簇，找到与其相关的修改文档，在文档里找到相关的修改代码。如图 2，对 HADOOP 软件修改历史库的挖掘，并把修改分成了“INCOMPATIBLE CHANGES”、“NEW FEATURES”、“IMPROVEMENTS”、“BUG FIXES”“OPTIMIZATIONS”。针对修改请求“Race condition in JMX cache update”可知属于“BUG FIXES”，根据”BUG FIXES”中修改请求的预处理以及聚类，可看到与修改请求相关的修改文档以及其修改代码，如图 3。

[0059] 3) 将修改方法、修改类映射到代码层次，利用调用和继承等依赖关系计算代码层次的影响集。构建一个包含影响集中所有方法的调用以及所有类的继承关系的传播图，然后通过推导的数学公式计算传播图中被影响节点的可能性的总和，以量化整体的修改影响。例如，类继承图，如图 4 AirExpressCenters 类和 GroundShippingCenters 类继承 FullfillmentCenters 类。getEligibleCenters() 在 FullfillmentCenters 中申明，AirExpressCenters 中方法覆盖 FullfillmentCenters 中的方法。FullfillmentCenters 类中方法的改变可能影响 GroundShippingCenters 类但不会影响 AirExpressCenters 中的方法。调用关系图，如图 5，2，3 调用 1，4 调用 3，当 1 改变时，3 会被直接影响，而 4 可能会间接被影响。然后将类继承图和方法调用图结合起来，构造整体的传播图，并用公式计算影响值，值越小，影响越小，如图 6。

[0060] 4) 使用 APK 分析器组件从程序源代码中提取事件列表，使用数据采集器 (Data Collctor) 从事件日志中提取事件列表，如图 7，最终以元组 e_i = <Activity，Window，GUI-Component，Action> 的形式表示，一个元组表示一个事件。

[0061] 5) 将修改代码在源代码中找到其所在的 Activity，与事件序列中的 Activity 相匹配，得到修改对应的事件序列，分析其可能影响软件的功能。例如，根据修改代码可知修改影响 Activity 为 EditTaskActivity，则可找到关于 EditTaskActivity 对应的事件序列，如图 8，影响了内容，文本的单击事件以及键盘的显示。

[0062] 6) 从移动应用市场利用网页爬虫技术收集 APP 的原数据，提取用户评论，并进行预处理；如图 9，从百度手机助手上提取“拉钩”软件的用户评论，对其进行托肯化，去停用词，词干化三步操作。

[0063] 7) 对用户评论使用 K-Means 算法进行聚类，得到 k 个评论簇，然后与软件修改请求

匹配，找到与修改相关的评论。

[0064] 8) 将评论按时间间隔 T 进行划分，T 可以为任意的时间间隔。一般将 T 设置为月或周。当 T 为月时，表示按月进行划分；当 T 为周时，表示按周进行划分。划分的结果将记录在时间序列文档 (SEQ) 中。时间序列文档记录一共有多少个时间段以及每个时间段内的评论数量，其格式为：第一行记录一共有多少个时间段，其数目记为 N；第二行到第 N+1 行分别为每个时间段内的提交消息数量。时间序列文档的格式见图 10。时间序列文档的示例见图 11，第一行的 3 代表一共有 3 个时间段，第二行的 100 代表第一个时间段内有 100 篇评论，第三行的 90 代表第二个时间段内有 90 篇评论，第四行的 80 代表第三个时间段内有 80 篇评论。

[0065] 9) 绘制直方图，更清晰方便地分析不同时间段受影响的用户群体。例如，如图 12 可知，前 20 天的时候，该修改对用户的影响很大，有将近 43 个人发现了此类的问题；后来，该修改对用户的影响很小很小。

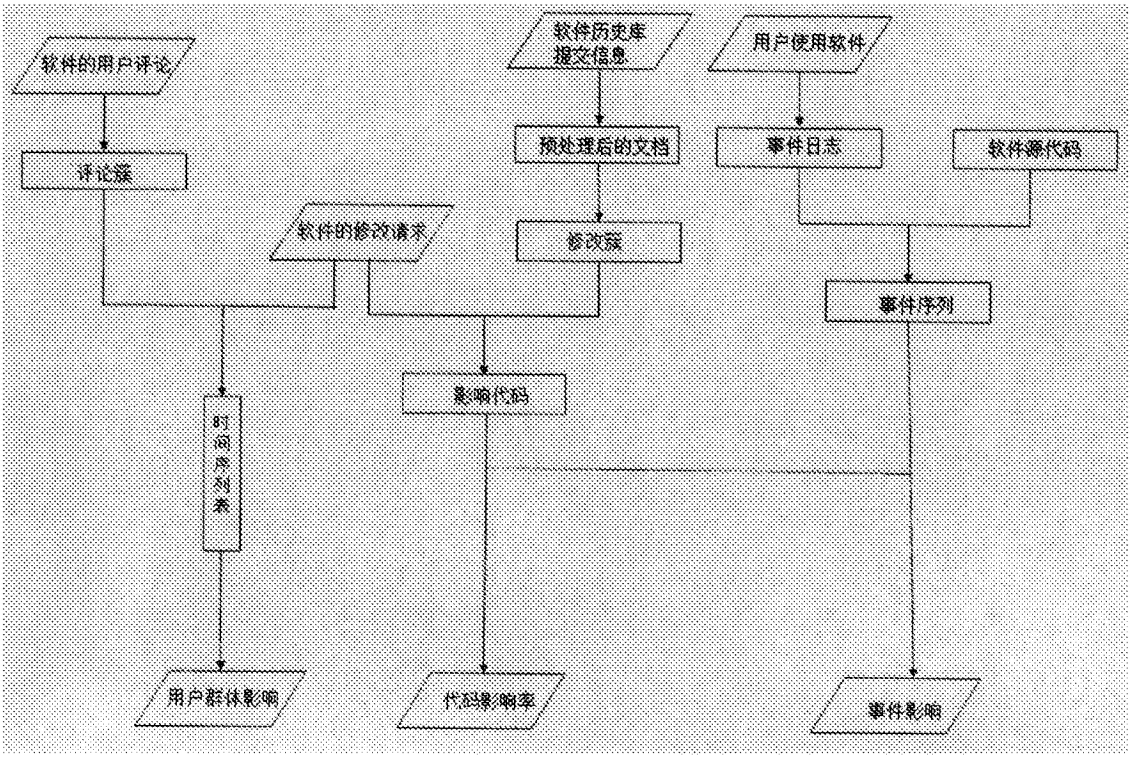
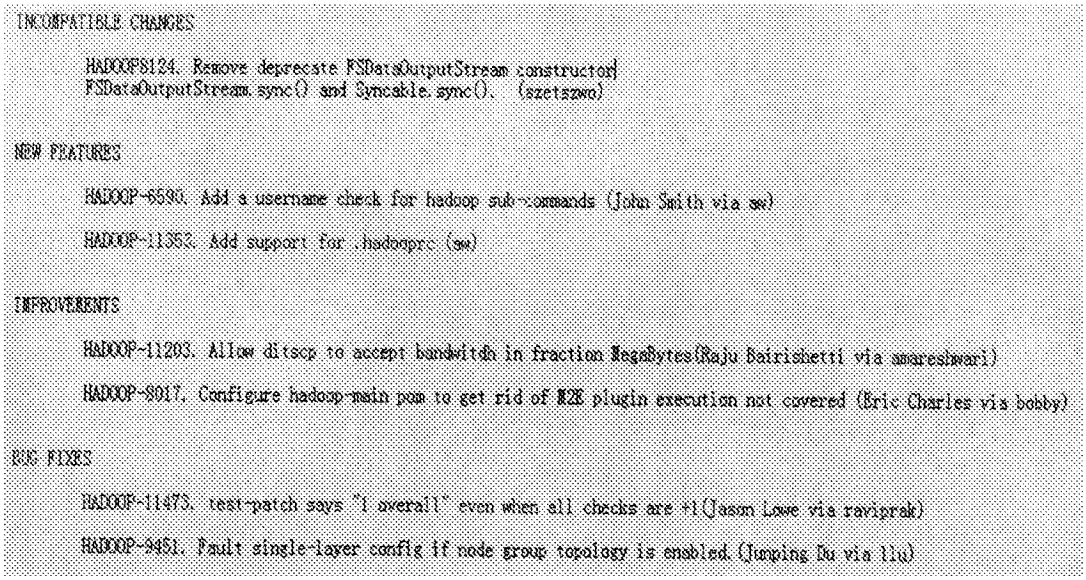


图 1



INCOMPATIBLE CHANGES

HADOOP-8124. Remove deprecate FSDataOutputStream constructor, FSDataOutputStream.sync() and Syncable.sync(). (szetszwo)

NEW FEATURES

HADOOP-6590. Add a username check for hadoop sub-commands (John Smith via aw)

HADOOP-11353. Add support for .hadooprc (aw)

IMPROVEMENTS

HADOOP-11203. Allow ditscp to accept bandwitdh in fraction MegaBytes(Raju Bairishetti via amareshwari)

HADOOP-8017. Configure hadoop-main pom to get rid of M2E plugin execution not covered (Eric Charles via bobby)

BUG FIXES

HADOOP-11473. test-patch says "-1 overall" even when all checks are +1(Jason Lowe via raviprak)

HADOOP-9451. Fault single-layer config if node group topology is enabled.(Junping Du via llu)

图 2

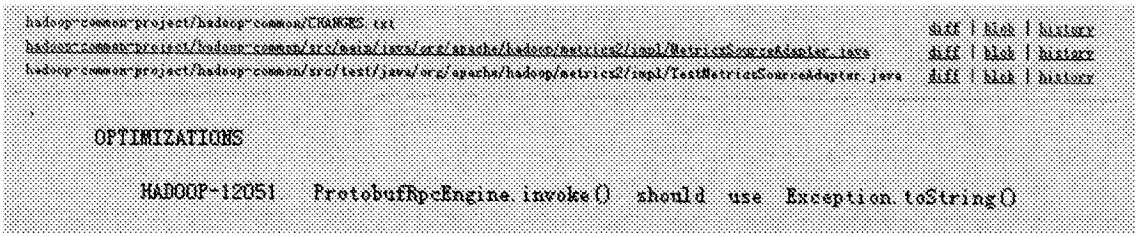

hadoop-common-project/hadoop-common/CHANGES.txt
diff | blob | history
hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/metrics2/impl/MetricsSourceAdapter.java
diff | blob | history
hadoop-common-project/hadoop-common/src/test/java/org/apache/hadoop/metrics2/impl/TestMetricsSourceAdapter.java
diff | blob | history
OPTIMIZATIONS
HADOOP-12051 ProtobufRpcEngine.invoke() should use Exception.toString()

图 3

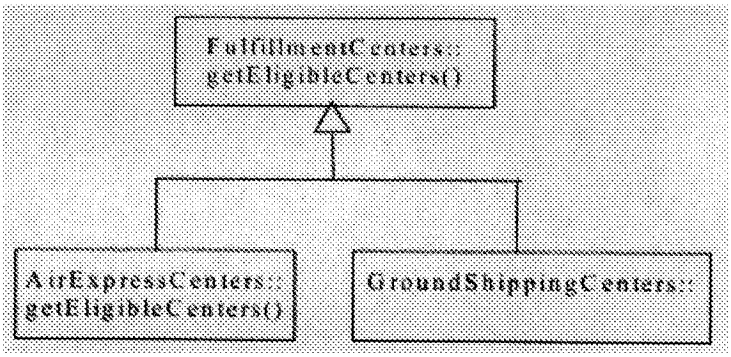

FulfillmentCenters::
getEligibleCenters()
AirExpressCenters::
getEligibleCenters()
GroundShippingCenters::

图 4

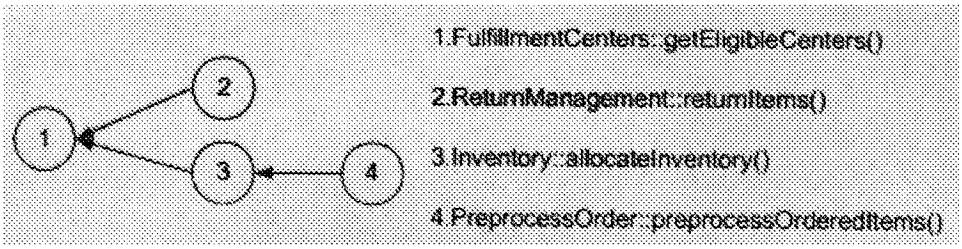

1
2
3
4
1.FulfillmentCenters::getEligibleCenters()
2.ReturnManagement::returnItems()
3.Inventory::allocateInventory()
4.PreprocessOrder::preprocessOrderedItems()

图 5

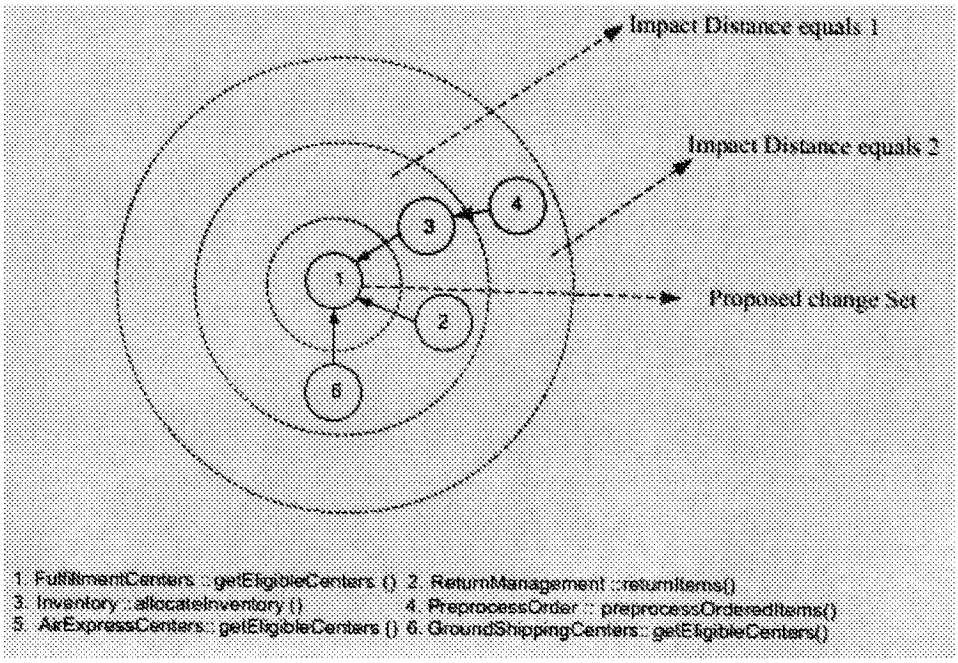

Impact Distance equals 1
Impact Distance equals 2
Proposed change Set
1
2
3
4
5
1. FulfillmentCenters :: getEligibleCenters () 2. ReturnManagement :: returnItems()
3. Inventory :: allocateInventory () 4. PreprocessOrder :: preprocessOrderedItems()
5. AirExpressCenters :: getEligibleCenters () 6. GroundShippingCenters :: getEligibleCenters()

图 6

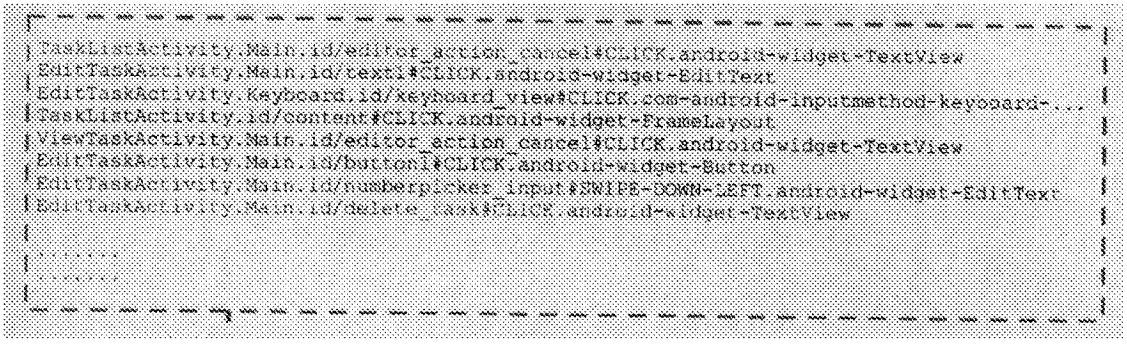

TaskListActivity.Main.id/editor_action_cancel#CLICK.android-widget-TextView
EditTaskActivity.Main.id/text1#CLICK.android-widget-EditText
EditTaskActivity.Keyboard.id/keyboard_view#CLICK.com-android-inputmethod-keyboard-...
TaskListActivity.id/content#CLICK.android-widget-FrameLayout
ViewTaskActivity.Main.id/editor_action_cancel#CLICK.android-widget-TextView
EditTaskActivity.Main.id/button1#CLICK.android-widget-Button
EditTaskActivity.Main.id/numberpicker_input#SWIPE-DOWN-LEFT.android-widget-EditText
EditTaskActivity.Main.id/delete_task#CLICK.android-widget-TextView
.....
.....

图 7

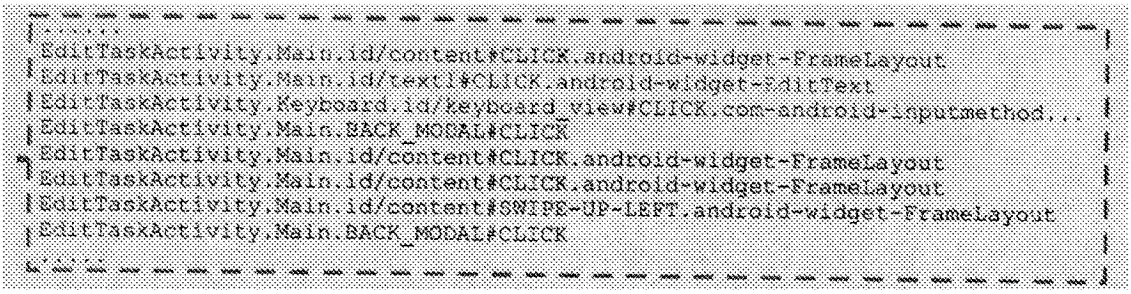

.....
EditTaskActivity.Main.id/content#CLICK.android-widget-FrameLayout
EditTaskActivity.Main.id/text1#CLICK.android-widget-EditText
EditTaskActivity.Keyboard.id/keyboard_view#CLICK.com-android-inputmethod...
EditTaskActivity.Main.BACK_MODAL#CLICK
EditTaskActivity.Main.id/content#CLICK.android-widget-FrameLayout
EditTaskActivity.Main.id/content#CLICK.android-widget-FrameLayout
EditTaskActivity.Main.id/content#SWIPE-UP-LEFT.android-widget-FrameLayout
EditTaskActivity.Main.BACK_MODAL#CLICK

图 8

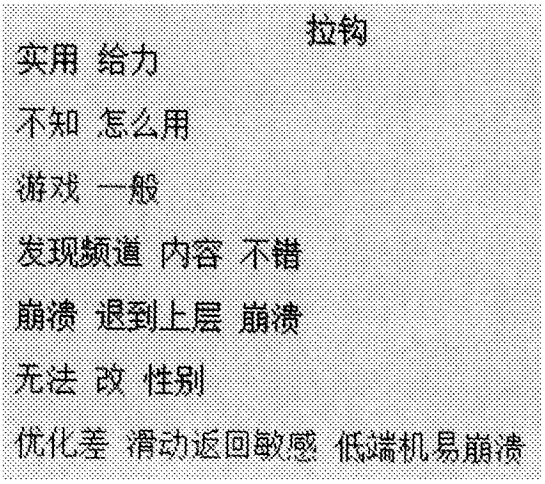


图 9

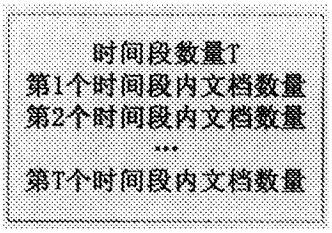


图 10

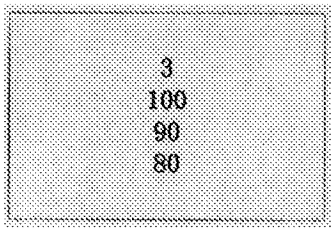


图 11

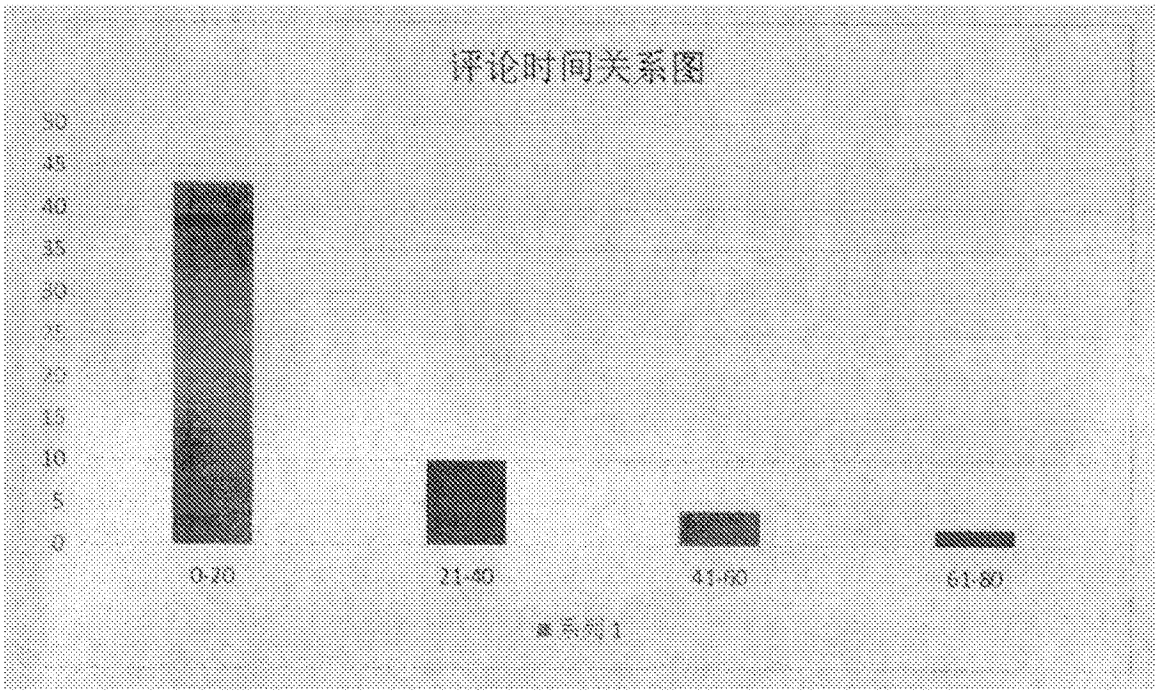


图 12