Mining of Android SCM

Pavel Senin

February 21, 2012

Abstract

MSR (mining software repositories) challenge is an annual competition for researchers working in the area of software engineering to demonstrate their tools and approaches for discovering actionable information within software artifacts trails. Android OS was selected as the research subject for 2012 MSR Challenge. Here I report an application of Software Trajectory toolkit to the challenge data along with discussing my findings. (I will put a summary of findings here later)

1 Introduction

The research field of Mining Software Repositories (MSR) is focused on analyzes of the rich data available in software repositories. The immediate goal of such research is to infer interesting and actionable information about software systems and projects. While there are multiple scientific events where researchers working in the field meet, the Mining Software Repositories conference considered to be the major one. The current 2012 MSR conference is a 9th such event and is collocated with ICSE -the International Conference on Software Engineering. Along with research track, since 2006 MSR Conference agenda includes the Mining Challenge where researchers demonstrate application of their tools to the selected repository mining problem. This year Android platform was selected for the challenge.

2 Android OS

"Android is an open-source software stack for mobile phones and other devices" as it said at its home website, `http://source.android.com/`:

The development of Android OS was started by Android Inc. - a small startup company - which was purchased by Google in 2005. Later, in November 2007, the Open Handset Alliance, a consortium of 84 companies, announced the availability of the Android Software Development Kit (SDK). This open Open Handset Alliance was formed of hardware, software, and telecommunication companies (including Intel, HTC, ARM, Samsung and Motorola) and its creation was devoted to advancing of open standards for mobile devices.

As with any other open-source system, the Android OS code is open and released under the Apache License. As promised, it is a complete mobile platform built on the monolithic Linux 2.6 kernel. The Android platform provides developers with an SDK consisting of a set of development tools, debugger and a true device emulator. To speed-up development there are an Eclipse plugin, a set of libraries, a multimedia user interface, and a core set of phone and mobile applications. The Android application model alleviates the cost of software development by allowing extending, replacing, and reusing

of existing software components. The Dalvik virtual machine, which is the part of Android distribution, provides a way to maximize application portability, performance and security. Android OS allows true application multitasking, provides rich user notifications and customizable home screens with resizable widgets. Latest, 4.0 version of Android System provides streaming voice recognition.

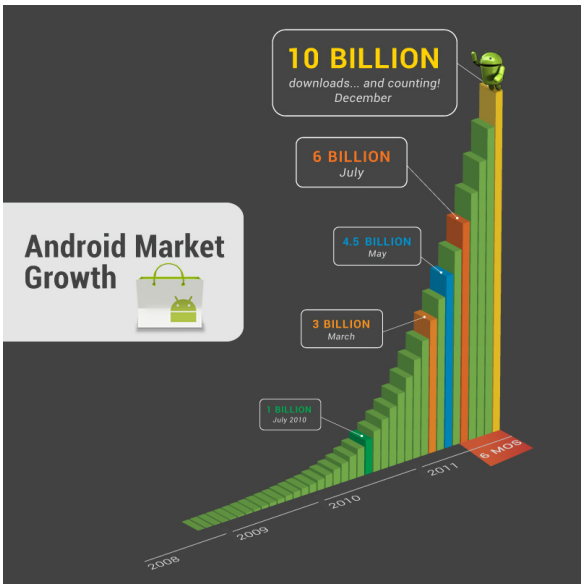


Figure 1: The Android store downloads timeline.

Currently, the Android Open Source Project (AOSP) is led by Google and includes not only the original members of OHA but many other companies. The role of AOSP is to maintain and develop Android.

Including first beta, there were 10 major releases of Android as well as a number of intermediate releases. All releases prior to 2.0 were for mobile phones exclusively. Since 2.0 release Android OS is a tablet-oriented operating system. Most of the mobile devices at market use 2.x version of Android. The 3.0 release, Honeycomb, does not officially run on phones; the latest 4.0 release, Ice Cream Sandwich, runs on all mobile devices.

In Q3, 2011 (November 15, 2011) Android officially become the most popular OS for newly sold mobile devices. In December 2011 there were registered 10 billions of downloads from Android store called Android Marketplace.

Two XML files containing change and bug report data were offered to participants to uncover interesting facts related to the Android platform.

Listing 1: List of metadata provided by change trail XML (fragment)

```
<xs:element name="change">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="project"/>
       <xs:element ref="commit_hash"/>
       <xs:element ref="tree_hash"/>
       <xs:element ref="parent_hashes"/>
       <xs:element ref="author_name"/>
       <xs:element ref="author_e-mail"/>
       <xs:element ref="author_date"/>
       <xs:element ref="commiter_name"/>
       <xs:element ref="commiter_email"/>
       <xs:element ref="committer_date"/>
       <xs:element ref="subject"/>
       <xs:element ref="message"/>
       <xs:element ref="target"/>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
```

3 Research question.

In this exploratory study I am trying to address an important question - is the SAX-based temporal data mining methods [1] [3] [2] are capable of discovering recurrent behaviors and if the discovered behaviors reflect any interesting and actionable information.

4 Challenge data.

Two XML files were offered for the MSR challenge. These files contain the most of the information obtainable from Google-hosted source code repository (Git) as well as Google-hosted bug and issue tracking system (Google project hosting). The full XSD for both XML files can be found in the Appendix section of this report. While the issues and comments trail 2 provided nearly complete information, the change trail 3 provided for a challenge contained only the high-level change information. The thirteen data fields of the change trail XSD file shown at the fragment Listing 1. The provided data contains information about revision tree, author and a commiter identification, change message and affected targets. Since I am focusing on mining of temporal patterns for inferring recurrent behaviors, I consider this provided dataset rather poor since it lacks many of auxiliary change and target information. In fact, among relevant to Trajectory toolkit data fields, every timestamped entity of the offered trail contain only author id and an indication of affected targets (the committer information is rather useless for trajectory since it could not be trusted due to the nature of git merged commits).

4.1 Android SCM facts

There are 1771667 changes registered in the repository. First change commit to Android repository dates back to Monday, January 12th, 1970, 14:46:40, while the last change within the analyzed data set belongs Tuesday, June 21st 2011, 12:41:41. The detailed data about changes per project is available in the Appendix Table 4. I was not able to recover the state of repository as it was parsed into XML, but per January, 1, 2012 Android repository contains a total of 26851267 lines of source code in 152225 files.

5 Auxiliary information collection and data organization.

In order to enrich the provided data for recurrent pattern analysis I decided to collect auxiliary information for changes. I have created a local mirror of Android OS and by iterating over existing commits hashes was able to recover auxiliary data for 68% of existing commits. The rest of changes which is about 32% of total change information belonging to legacy projects is unrecoverable due to the recent changes in Android repository.

For every recoverable change record I collected a summary of added, modified or deleted files as well as a summary about LOC changes: added, modified or deleted lines.

All this information was stored in the Trajectory database backend (since the TrajectoryDB schema was designed to accommodate change artifact trails from CVS and SVN repositories I have extended it to accommodate Git change data such as tree hash, commit hash and a commiter identification). The main tables of TrajectoryDB correspond to entities of change log and issue log; these tables accompanied with separate tables for change targets and issue comments as well as tables for authors, contributors etc. Overall, the TrajectoryDB is normalized and optimized for the fast retrieval of change and issue information using SQL language.

6 Threats to Validity

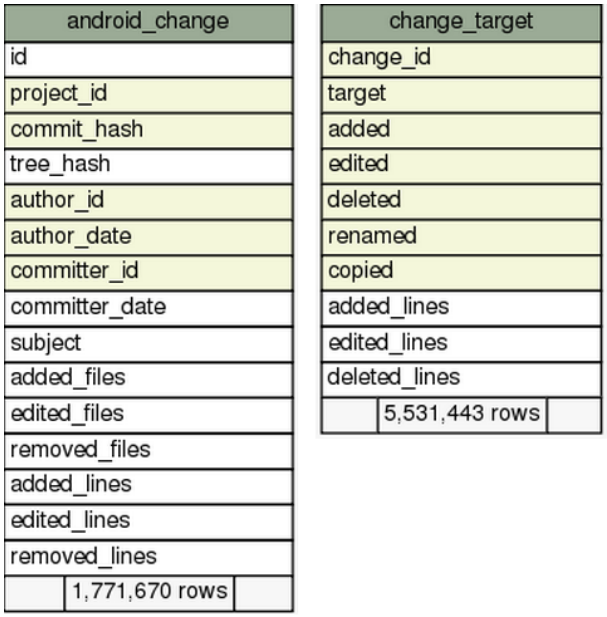


Figure 2: The Trajectory DB schema fragment. Change and target tables shown.

Before discussing the mining methodology and my findings I discuss several threats to validity of this research and how I have addressed them. These include general threats to construct validity, external validity, and specific threats to my particular methodology including repository threats, recall and precision.

6.1 General threats.

Construct validity requires that Trajectory must correctly identify authors and their activity. The threat to the construct validity comes from the shortcomings of both: the repository data and the Trajectory toolkit. First of all, it is hard to access how much of the contributor's activity devoted to the project is actually captured within the artifact trails. Certainly, it is insufficient to look only on the this data for drawing any general conclusion about contributors behavior or about all or part of Android operating system. Moreover the approximation and reduction of information within Trajectory workflow creates further difficulties for any assertion. Thus, I do not claim external validity and as in any other exploratory and characteristic in nature study this work should be considered together with evaluation methodology.

6.2 Repository data threats.

The Android OS developers using Git as the SCM system. Git is a distributed revision control system with an emphasis on speed. Git keeps all the history on the local machine allowing very fast branching and easy repeated merging of branches thus creating multiple trees which deviate from mainline (trunk). In my analyzes I ignore Git merge commits which only record metadata about integration between different development trees.

6.3 Treats due to incompleteness of data.

The information enclosed for the challenge appears to be partially not reproducible due to the changes happened within the Android OS hosting scheme and deletion of some legacy projects. Based on my own retrieval of the auxiliary data, it is impossible to confirm about 32% of all change records found in XML file. This fact implies that in my analyzes I missed some of the quantitative change information, however in my opinion there is no reason to believe that the lack of this data affect behavioral patterns recognition and classification. The reason for this statement is that the general development behavior doesn't change much between projects. For example ...

Table 1: Sliding window, PAA and alphabet size choices.

Sliding window size	PAA size	Alphabet size
one week (7)	3	3
two weeks (14)	5	5
month (30)	5	5

6.4 Threats to commit characteristics.

Some of the contributors of Android OS commit using several email addresses and some of the contributors have a variation in their names thus possessing a threat to author classification. In order to address this threat I have merged together authors which share the same full name reducing the authors id set from 11311 names to 8706.

7 Data curation.

In order to proceed with patterns mining apply Trajectory framework After the collection of auxiliary data The data collected at the previous step was filtered into the separate database table for further analyzes. The reason to perform this step was the overall amount of merged commits (35K) as well as the amount of commits which lacked the auxiliary data (581K).

8 Data reduction and indexing

SAX indexing depends on the three parameters which are required input. First parameter is the sliding window, second is the PAA approximation size and third is the SAX Alphabet size.

In this work I have selected three sizes for sliding window: 7 days, 14 days and 30 days. These represent an intuitive and logical intervals of a week, two weeks and a month.

For the PAA reduction I choose 3 steps for 7 days window, 5 steps for a weekly interval, and 7 steps for a monthly window.

Finally the alphabet, I choose 3 letters for weekly window, and 5 letters for bi-weekly and monthly windows. All these are summarized in the Table 1.

There will be a detailed indexing procedure description

9 Results

For application of the Trajectory toolkit I need to select particular projects from the Android OS project tree `https://sites.google.com/a/android.com/opensource/projects`. I have decided to select three sub-projects: the one of OS kernels, the Core Android App Framework libraries (frameworks_base) and a bluetooth subsystem components (platform_external_bluetooth_bluez and platform_external_bluetooth_glib).

As Figure 3 shows that there is almost no difference in the SCM trails of pre-2010 kernels, thus I have arbitrary selected the OMAP-kernel which is a customized Linux 2.6 kernel for OMAP-based devices. OMAP is a proprietary system on chips (SoCs) for portable and mobile multimedia applications and based on general-purpose ARM architecture processor provided by Texas Instruments `http://en.wikipedia.org/wiki/OMAP`.

While picking a single project from the kernel sub-projects was a trivial task. The choice of other projects was not that easy. If we look at the number of changes and their distribution in time, picking

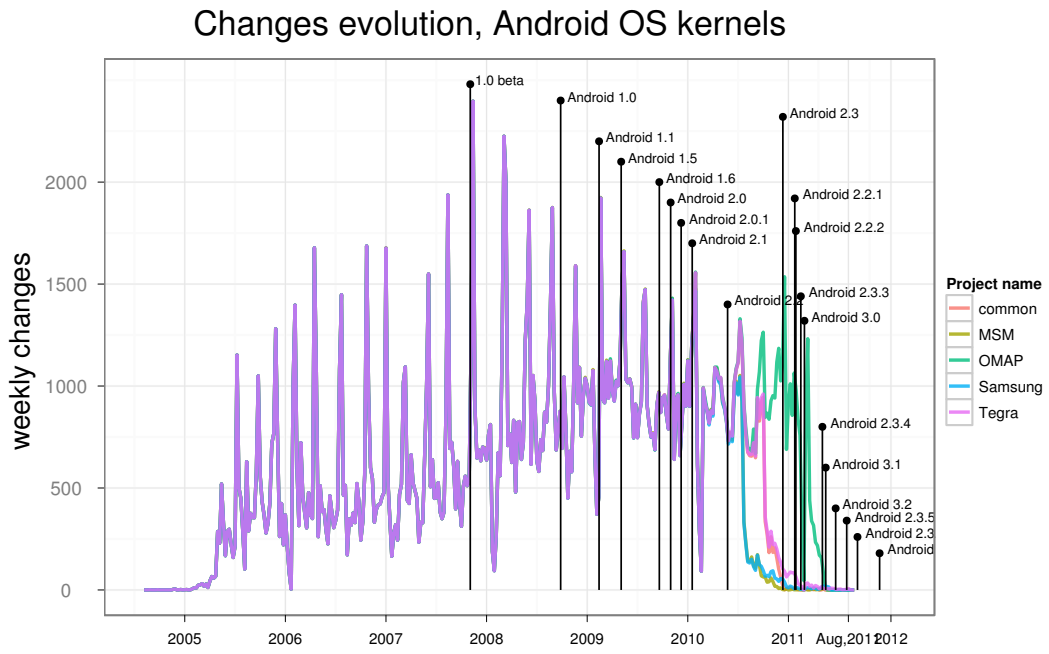


Figure 3: Weekly changes in Android OS kernel sub-projects

project randomly could impose the treat of insufficient sample size. By looking on the the number of changes and the activity interval I have picked the Core Android App Framework 4 and a bluetooth stack 5.

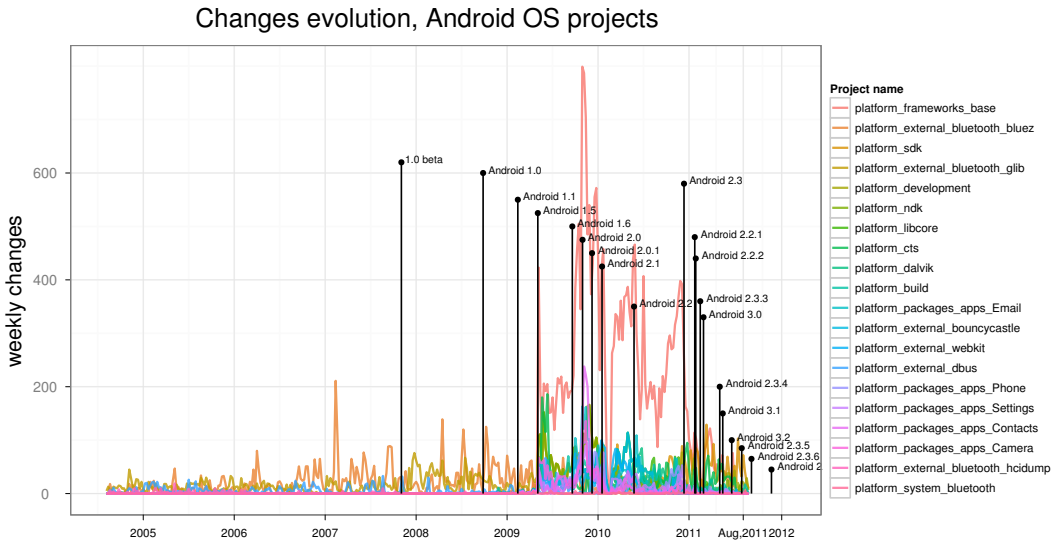


Figure 4: Weekly changes in Android OS sub-projects

By indexing these three trails with Trajectory SAX workflow and successive indexing I obtained dictionaries of patterns and their occurence frequencies. By using SQL it is fairly easy to extract the index of patterns for the project, inteval and an author.

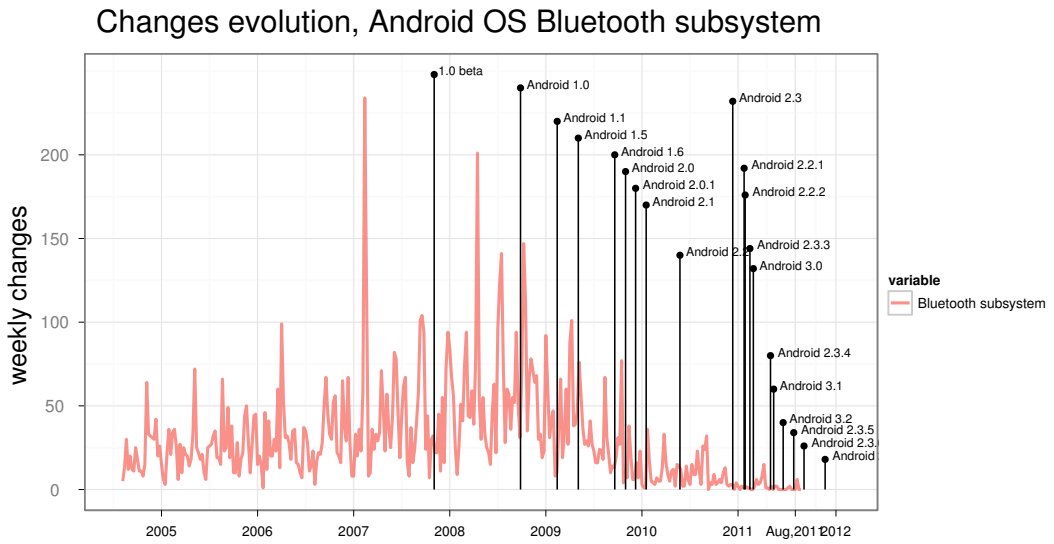


Figure 5: Weekly changes in Android OS Bluetooth sub-system.

Table 2: OMAP kernel weekly patterns sorted by occurence.

Release label	Sorted patterns
Android 1.0-2M	bbb,cbb,bbc,bcb,ccc,cba,acb,bca,abc,bac,caa,bba,abb,cab,aac,bab,aca,cac,cca,acc
Android 1.0-1M	bbb,cbb,bbc,bcb,ccc,bca,cba,abc,acb,bba,caa,aac,abb,bac,cab,bab,aca,acc,cca
Android 1.0+1M	bbb,bbc,cbb,bcb,ccc,cba,abc,acb,bca,caa,aac,abb,bba,cab,bac,bab,aca,cac
Android 1.1-2M	bbb,bbc,cbb,bcb,ccc,abc,cba,acb,bca,abb,bba,caa,aac,cab,bac,bab,aca,cca
Android 1.1-1M	bbb,bbc,cbb,bcb,ccc,cba,bca,abc,acb,caa,cab,bba,aac,abb,bac,bab,aca
Android 1.1+1M	bbb,cbb,bbc,bcb,ccc,cba,acb,bca,abc,cab,bac,abb,bba,caa,aac,bab,aca,acc,cca
Android 1.5-2M	bbb,bbc,cbb,bcb,ccc,bca,acb,cba,abc,abb,cab,bac,bba,aac,bab,caa,aca,acc,cca
Android 1.5-1M	bbb,cbb,bbc,bcb,ccc,cba,bca,acb,abc,caa,bba,abb,aac,bac,cab,bab,aca
Android 1.5+1M	bbb,bbc,cbb,bcb,ccc,acb,cba,bca,abc,cab,abb,aac,bac,caa,bba,bab,aca
Android 1.6-2M	bbb,bbc,cbb,bcb,ccc,acb,bca,cba,abc,abb,aac,caa,bba,cab,bac,bab,aca
Android 1.6-1M	bbb,bbc,cbb,bcb,ccc,bca,acb,cba,abc,bba,abb,aac,caa,cab,bac,bab,aca
Android 1.6+1M	bbb,bbc,cbb,bcb,ccc,cba,bca,acb,abc,caa,abb,bac,cab,bba,aac,bab,aca

9.1 OMAP kernel pre-release and post-release patterns

First I looked on the patterns seen within the Android OS release proximity in OMAP kernel project. The table 2 shows sorted by frequency weekly patterns observed by indexing a stream of changed targets.

By applying SAX minimal bounding distance to these symbolic vectors I have obtained the distance matrix 3. Which I used for clustering 9.1.

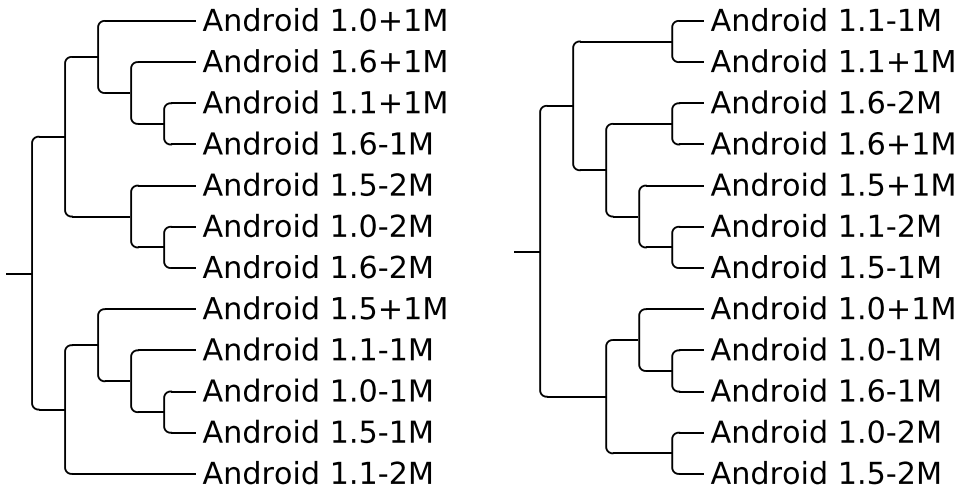
10 Future work

Ultimately by application of Trajectory I want to discover recurrent behaviors from software process artifacts trails. This data can be further applied to improve the productivity of development.

11 Appendix

Table 3: OMAP kernel weekly patterns distance matrix.

Android 1.0-2M	0.0	6.02	3.44	5.16	2.58	2.58	3.44	1.72	4.3	2.58	3.44	3.44
Android 1.0-1M	6.02	0.0	5.16	4.3	0.86	4.3	2.58	0.0	2.58	5.16	4.3	3.44
Android 1.0+1M	3.44	5.16	0.0	8.6	3.44	2.58	6.02	3.44	5.16	5.16	1.72	1.72
Android 1.1-2M	5.16	4.3	8.6	0.0	3.44	8.6	7.74	4.3	4.3	3.44	6.88	5.16
Android 1.1-1M	2.58	0.86	3.44	3.44	0.0	2.58	4.3	0.0	5.16	6.0	6.02	2.58
Android 1.1+1M	2.58	4.3	2.58	8.6	2.58	0.0	2.58	3.44	3.44	2.58	0.0	0.0
Android 1.5-2M	3.44	2.58	6.02	7.74	4.3	2.58	0.0	2.58	5.16	2.58	1.72	3.44
Android 1.5-1M	1.72	0.0	3.44	4.3	0.0	3.44	2.58	0.0	1.72	5.16	2.58	2.58
Android 1.5+1M	4.3	2.58	5.16	4.3	5.16	3.44	5.16	1.72	0.0	4.3	2.58	1.72
Android 1.6-2M	2.58	5.16	5.16	3.44	6.02	2.58	2.58	5.16	4.3	0.0	1.72	3.44
Android 1.6-1M	3.44	4.3	1.72	6.88	6.02	0.0	1.72	2.58	2.58	1.72	0.0	0.86
Android 1.6+1M	3.44	3.44	1.72	5.16	2.58	0.0	3.44	2.58	1.72	3.44	0.86	0.0



(a) Clustering of all changed targets

(b) Clustering of nightly edited targets

Figure 6: Clustering of SCM trails based on SAX approximation

Table 4: Android repository projects and associated change records statistics for projects with more than 500 changes.

DB project id	Project name	Total change records
15	kernel_linux-2.6	254073
18	kernel_omap	234235
21	kernel_tegra	213250
13	kernel_common	211941
19	kernel_qemu	211915
20	kernel_samsung	202860
17	kernel_msm	202773
14	kernel_experimental	110251
143	platform_frameworks_base	30775
40	platform_external_bluetooth_bluez	7249
226	platform_sdk	6638
41	platform_external_bluetooth_glib	6620
30	platform_development	5064
163	platform_ndk	4555
161	platform_libcore	3631
28	platform_cts	3631
29	platform_dalvik	3492
27	platform_build	3406
265	tools_gerrit	3085
174	platform_packages_apps_Email	2973
45	platform_external_bouncycastle	2971
135	platform_external_webkit	2780
50	platform_external_dbus	2519
188	platform_packages_apps_Phone	2315
192	platform_packages_apps_Settings	2227
172	platform_packages_apps_Contacts	2185
170	platform_packages_apps_Camera	2104
228	platform_system_core	1770
183	platform_packages_apps_Mms	1620
167	platform_packages_apps_Browser	1599
181	platform_packages_apps_Launcher2	1560
207	platform_packages_providers_ContactsProvider	1141
107	platform_external_opencore	1129
22	platform_bionic	1061
116	platform_external_qemu	954
12	device_samsung_crespo	950
202	platform_packages_inputmethods_LatinIME	854
169	platform_packages_apps_Calendar	732
173	platform_packages_apps_DeskClock	667
155	platform_hardware_msm7k	663
184	platform_packages_apps_Music	629
175	platform_packages_apps_Gallery3D	609
149	platform_frameworks_policies_base	549
206	platform_packages_providers_CalendarProvider	522
119	platform_external_skia	500

Table 5: Android repository snapshot source code metrics.

Nb. files	Language	Total lines	Source	Blanks	Comments
Assembly	3391	565314	426104	60673	98215
IDL	78	7926	7174	752	640
Perl	348	105024	75877	13999	18975
CSS	224	38417	30687	5899	2033
XML	10090	5963665	5762297	59093	143243
Matlab	889	62911	51323	10715	3911
Text	7563	1669710	0	93271	1576439
FlashParameter	3	364	265	40	59
C	42151	9979284	6676807	1278259	2363003
shell	3930	2476537	1986936	207861	288125
JavaScript	5347	866236	511330	131581	225745
Java	251129	5265723	3152054	639535	1509917
HTML	6209	1208415	1062506	100306	61760
make	3506	512065	377850	62696	71518
Awk	21	2762	1778	226	871
SQL	3	454	454	0	0
Pascal	160	15681	2555	372	13334
Python	912	190278	132202	30889	27888
PHP	180	50137	39060	2268	9218
C++	31390	9228878	6262708	1318921	1835150
Jess	2	246	192	54	0
CSharp	66	6513	5282	643	596
Lisp	10633	493777	285826	86471	176378
TOTAL	152225	38710317	4104524	8427018	26851267

Listing 2: Bugs XML file schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="Android_Bugs">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="bug"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="bug">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="bugid"/>
        <xs:element ref="title"/>
        <xs:element ref="status"/>
        <xs:element ref="owner"/>
        <xs:element ref="closedOn"/>
        <xs:element ref="type"/>
        <xs:element ref="priority"/>
        <xs:element ref="component"/>
        <xs:element ref="stars"/>
        <xs:element ref="reportedBy"/>
        <xs:element ref="openedDate"/>
        <xs:element ref="description"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="comment"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="bugid" type="xs:integer"/>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="status" type="xs:NCName"/>
  <xs:element name="owner" type="xs:string"/>
  <xs:element name="closedOn" type="xs:NMTOKEN"/>
  <xs:element name="type" type="xs:string"/>
  <xs:element name="priority" type="xs:string"/>
  <xs:element name="component" type="xs:string"/>
  <xs:element name="stars" type="xs:integer"/>
  <xs:element name="reportedBy" type="xs:string"/>
  <xs:element name="openedDate" type="xs:string"/>
  <xs:element name="description" type="xs:string"/>
  <xs:element name="comment">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="author"/>
        <xs:element ref="when"/>
        <xs:element ref="what"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="author" type="xs:string"/>
  <xs:element name="when" type="xs:string"/>
  <xs:element name="what" type="xs:string"/>
</xs:schema>
```

Listing 3: Change XML file schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="changes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="change"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="change">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="project"/>
        <xs:element ref="commit_hash"/>
        <xs:element ref="tree_hash"/>
        <xs:element ref="parent_hashes"/>
        <xs:element ref="author_name"/>
        <xs:element ref="author_e-mail"/>
        <xs:element ref="author_date"/>
        <xs:element ref="commiter_name"/>
        <xs:element ref="commiter_email"/>
        <xs:element ref="committer_date"/>
        <xs:element ref="subject"/>
        <xs:element ref="message"/>
        <xs:element ref="target"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="project" type="xs:NCName"/>
  <xs:element name="commit_hash" type="xs:string"/>
  <xs:element name="tree_hash" type="xs:string"/>
  <xs:element name="parent_hashes" type="xs:base64Binary"/>
  <xs:element name="author_name" type="xs:string"/>
  <xs:element name="author_e-mail" type="xs:string"/>
  <xs:element name="author_date" type="xs:string"/>
  <xs:element name="commiter_name" type="xs:string"/>
  <xs:element name="commiter_email" type="xs:string"/>
  <xs:element name="committer_date" type="xs:string"/>
  <xs:element name="subject" type="xs:string"/>
  <xs:element name="message">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="line"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="target">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="line"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="line" type="xs:string"/>
</xs:schema>
```

References

[1] *HOT SAX: efficiently finding the most unusual time series subsequence*, 2005.

[2] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286, Aug. 2001.

[3] E. Keogh, S. Lonardi, and B. Y. chi' Chiu. Finding surprising patterns in a time series database in linear time and space. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 550–556, New York, NY, USA, 2002. ACM.