Boa:
An End-to-end Repository Mining Platform

Dr. Robert Dyer
Bowling Green State University

The research and educational activities described in this talk was supported in part by the US National Science Foundation (NSF) under grants CNS-18-23294, CCF-15-18776, CNS-15-12947, CNS-15-13263, and CCF-13-49153.
Students

Che Shian Hung

Rebecca Brunner

Jingyi Su

Kaushik Nimmala

Mohd Arafat

Shubhendra Shrimal
What do I mean by *software repository*?
What do I mean by *ultra large scale*?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>7,830,023</td>
</tr>
<tr>
<td>Code Repositories</td>
<td>380,125</td>
</tr>
<tr>
<td>Revisions</td>
<td>23,229,406</td>
</tr>
<tr>
<td>Unique Files</td>
<td>146,398,339</td>
</tr>
<tr>
<td>File Snapshots</td>
<td>484,947,086</td>
</tr>
<tr>
<td>AST Nodes</td>
<td>71,810,106,868</td>
</tr>
</tbody>
</table>
Why do we want to mine software repositories?
Detecting/correcting API mis-uses

Training models

Analyzing language feature usage

Mining patterns

Clone detection
Consider a task to answer

"How was unit testing adopted over time?"
Is fork?

Access repository

mine revisions

Find all Java source files

mine sources

Find all Methods

Result: Unit Test Usages

Method has @Test?

Yes

Foreach project

mine project metadata
class UnitTestFinder {
    static void main(String[] args) {
        ... /* create and submit a Hadoop job */
    }

    static class UnitTestFinder.Mapper extends Mapper<Text, BytesWritable, Text, LongWritable> {
        static class DefaultVisitor {
            ... /* define default tree traversal */
        }

        void map(Text key, BytesWritable value, Context context) {
            final Project p = ... /* read from input */
            new DefaultVisitor() {
                Boolean preVisit(Modifier n) {
                    if (n.kind == ModifierKind.ANNOTATION && match("^(org\s\s).junit\s\s)?Test\$", n.annotation_name)) {
                        context.write(new Text("Tests\" + current(n).getCommitDate() + "\""), new LongWritable(1));
                    }
                }
            }.visit(p);
        }
    }

    static class UnitTestFinder.Reducer extends Reducer<Text, LongWritable, Text, LongWritable> {
        void reduce(Text key, Iterable<LongWritable> vals, Context context) {
            int sum = 0;
            for (LongWritable value : vals) {
                sum += value.get();
                context.write(key, new LongWritable(sum));
            }
        }
    }
}

Full program over 140 lines of code

Uses JSON, SVN, and Eclipse JDT libraries

Uses Hadoop framework

Explicit/manual parallelization

Too much code! Do not read!
A better solution

```java
Tests: output sum[timestamp] of int;

visit(input, visitor {
    before n: Modifier ->
        if (n.kind == ModifierKind.ANNOTATION &&
            match(`^(org\/.junit\/)Test$`, n.annotation_name))
            Tests[current(Revision).commit_date] << 1;
});
```

Full program **7 lines of code**!

**No external libraries** needed!

Automatically parallelized!

Analyzes **18 billion** AST nodes in minutes!
The Boa language and data-intensive infrastructure

http://boa.cs.iastate.edu/
How was unit testing adopted over time?

Tests: `output sum[timestamp] of int;`
How was unit testing adopted over time?

Tests: output sum[timestamp] of int;

visit(input, visitor {

})
How was unit testing adopted over time?

Tests: output sum[timestamp] of int;

visit(input, visitor {
    before n: Modifier ->
});
How was unit testing adopted over time?

Tests: `output sum[timestamp] of int;

visit(input, visitor {
  before n: Modifier ->
    if (n.kind == ModifierKind.ANNOTATION &&
How was unit testing adopted over time?

Tests: output sum[timestamp] of int;

visit(input, visitor {
    before n: Modifier -> 
    if (n.kind == ModifierKind.ANNOTATION &&
    match(`^(org\\.junit\.)?Test$`, n.annotation_name))
});
How was unit testing adopted over time?

Tests: output sum[timestamp] of int;

visit(input, visitor {
    before n: Modifier ->
        if (n.kind == ModifierKind.ANNOTATION &&
            match(`^(org\junit\)?Test$`, n.annotation_name))
            Tests[current(Revision).commit_date] <<= 1;
});
Tests: output sum[timestamp] of int;

visit(input, visitor {
    before n: Modifier ->
        if (n.kind == ModifierKind.ANNOTATION &&
            match(`^(org\junit\)?Test$`, n.annotation_name))
            Tests[current(Revision).commit_date] << 1;
});
Consider a more complex task

“Find the buggy files by examining past bug fixing behavior.”
“Find the buggy files by examining past bug fixing behavior.”

1a. Compute the fixing file counts, fixing revision counts (FRC), and average FRC across projects

1b. Download Boa output through Boa API

2a. Analysis Task

2b. Download Other Tools

3a. Analysis Task

3b. Output
“Find the buggy files by examining past bug fixing behavior.”

2a. Filter projects Based on MSR criteria

2b. Download Boa output through Boa API
“Find the buggy files by examining past bug fixing behavior.”

3a. Generate a list of potentially buggy files

3b. Output result
Boa does not enable sharing and reusability and consequently fails to become an end-to-end MSR platform.
Solution:

Materialized Views in Boa
“Find the buggy files by examining past bug fixing behavior.”
Language features

```plaintext
view Filter {
    view FixingRevision {
        FixFileCount: output sum[string][file: string]
        FixRevisionCount: output sum[string] of count:
        AverageFRC: output mean of int;
    }
    Retained: output collection[pid: string] of int;

    v := FixingRevision/AverageFRC;
    r: v._row;
    v >> r;
}
```
Language features

```java
1  view Filter {
2     Retained: output collection[pid: string] of int;
3     v := J12345/AverageFRC;
4     ...
5  }
6  o: output top(5)[pid: string] of fileName: string weight:
7
8  if (len(Filter/Retained[input.id]) > 0) {
9     v: output sum[file: string] of int =
10        @rdyer/FixingRevision/FixFileVersionCount[p.id];
11     r: v._row;
12     while (v >> r)
13    }
```
1. Submit Boa job

2. Request views with tag -views

3. Returning a list of external views

4. Retrieve job id for query roots with tag name

5. Compile query with resolved view paths and job ids

6. Compile finished
New HDFS Layout

```java
view Filter {
    view FixingRevision {
        FixFileCount: output sum[string][file: string] of int;
        FixRevisionCount: output sum[string] of count: int;
        AverageFRC: output mean of int;
        ...
    }
    Retained: output collection[pid: string] of int;

    v := FixingRevision/AverageFRC;
    r: v. row;
    v >> r;

    visit(input, visitor {
        before n: CodeRepository -> {
            v2 := FixingRevision/FixRevisionCount[input.id];
            r2: v2. row;
            if (v2 >> r2 && r2.count > r.1 && len(n.revisions) >= 100)
                Retained[input.id] << 1;
        }
    });

    o: output top(5)[pid: string] of fileName: string weight count: int;

    if (len(Filter/Retained[input.id]) > 0) {
        v := Filter/FixingRevision/FixFileCount[input.id];
        r: v. row;
        while (v >> r)
            o[input.id] << r.file weight r.2;
    }
}
```
New HDFS Layout

```java
view Filter {
  view FixingRevision {
    FixFileCount: output sum[string][file: string] of int;
    FixRevisionCount: output sum[string] of count: int;
    AverageFRC: output mean of int;
  }

  Retained: output collection[pid: string] of int;

  v := FixingRevision/AverageFRC;
  r: v._row;
  v >> r;

  visit(input, visitor {
    before n: CodeRepository -> {
      v2 := FixingRevision/FixRevisionCount[input.id];
      r2: v2._row;
      if (v2 >> r2 && r2.count > r._1
        && len(n.revisions) >= 100)
        Retained[input.id] << 1;
    }
  });

  o: output top(5)[pid: string] of file: string weight: count: int;

  if (len(Filter/Retained[input.id]) > 0) {
    v := Filter/FixingRevision/FixFileCount[input.id];
    r: v._row;
    while (v >> r)
      o[input.id] << r.file weight r._2;
  }
}
```
New HDFS Layout

```java
view Filter {
  view FixingRevision {
    FixFileCount: output sum[string][file: string] of int;
    FixRevisionCount: output sum[string] of count: int;
    AverageFRC: output mean of int;
    ...
  }
  Retained: output collection[pid: string] of int;
  v := FixingRevision/AverageFRC;
  r: v..row;
  v >> r;
  visit(input, visitor {
    before n: CodeRepository -> {
      v2 := FixingRevision/FixRevisionCount[input.id];
      r2: v2..row;
      if (v2 >> r2 && r2.count > r_.1 &&
        len(n.revisions) >= 100)
        Retained[input.id] <<= 1;
      }
    });
  output top(5)[pid: string] of fileName: string weight count: int;
}
if (len(Filter/Retained[input.id]) > 0) {
  v := Filter/FixingRevision/FixFileCount[input.id];
  r: v..row;
  while (v >> r)
    o[input.id] <<= r.file weight r_.2;
}
```
New HDFS Layout

```java
view FixingRevision {
  FixFileCount: output sum[string][file: string] of int;
  FixRevisionCount: output sum[string] of count: int;
  AverageFRC: output mean of int;
  ...
}
Retained: output collection[pid: string] of int;

v := FixingRevision/AverageFRC;
r: v.row;
v >> r;

visit(input, visitor { before n: CodeRepository -> {
  v2 := FixingRevision/FixRevisionCount[input.id];
  r2: v2.row;
  if (v2 >> r2 && r2.count > r_.l && len(n.revisions) >= 100)
    Retained[input.id] <<= 1;
}
});

if (len(Filter/Retained[input.id]) > 0) {
  v := Filter/FixingRevision/FixFileCount[input.id];
r: v.row;
  while (v >> r)
    o[input.id] <<= r.file weight r_.2;
}
```
Case Study 1: Splitting Identifiers

word10 \rightarrow \{\text{word, 10}\}

hardWord \rightarrow \{\text{hard, word}\}

hard_word \rightarrow \{\text{hard, word}\}

• *Binkley et al.* created a **gold set** containing 2,663 identifiers and the split form based on 8,522 human splitting judgements

• *Hill et al.* studied several splitting algorithms using gold set

• We applied **greedy splitting algorithm** to split the gold set
Case Study 1: Result

• Original study used three different dictionary sizes:
  • Small (50,276 entries) -> 56 %
  • Medium (98,569 entries) -> 51 %
  • Large (479,625 entries) -> 60 %

• Our dictionary contained **61,215** entries -> **53%**
Case Study 2: Developer Turnover Impact

- Foucault et al. analyzed turnover impact on 5 open source projects

- We computed turnover metrics and analyzed its impact on 1,676 open source projects
Developer Turnover – DAG

* Each query is also fed in the same Boa dataset.
Case Study 2: Results

Number of Projects for Each Level of External Ratio

Number of Projects for Each Conversion Rate Level

The Number of Projects for each Highest Activity Metric

Negative Correlation between Turnover and Bug-fixed Density

Positive Correlation between Turnover and Bug-fixed Density

Newcomer Activity vs. Leaver Activity
1,193 users
1,193 users, 35 countries
1,193 users, 35 countries, 6 continents
Future Research
Enabling Machine Learning in Boa

Key Challenges
1) What language abstractions can we provide to integrate ML techniques in Boa?
2) Can we scale the training phase?
Leveraging National Infrastructures

[1a] User submits query (on OSF.io)

[2a] User submits query (on boa.cs.iastate.edu)

[1b] Frontend transmits query to XSEDE backend

[1c/2b] Frontend transmits query to new XSEDE backend

Existing Frontend

New OSF.io Frontend (see D.2)

New XSEDE Backend (see D.3)

Boa compiler and runtime
(see D.1)

Machine Learning language features

Distributed Deep Learning runtime