Bringing Ultra-Large-Scale Software Repository Mining to the Masses with Boa

Robert Dyer

November 8, 2013
Department of Computer Science
Iowa State University

The research and educational activities described in this talk were supported in part by the US National Science Foundation (NSF) under grants CCF-13-49153, CCF-13-20578, TWC-12-23828, CCF-11-17937, CCF-10-17334, and CCF-10-18600.
Research Overview

Dynamic Aspect Virtual Machine Support - Nu
[AOSD’08] [TOSEM]

Language Evaluation - Ptolemy
[AOSD’12] [TAOSD]

Easing Ultra-large-scale Software Mining - Boa
[ICSE’13] [GPCE’13] [SPLASH’13 SRC]
In submission: [ICSE’14] Planned: [PLDI’14]
Why mine software repositories?

Learn from the past
Keep doing what works
Empirical validation
To find better designs
Spot (anti-)patterns
Inform the future
1,000,000+ projects

1,000,000,000+ lines of code

10,000,000+ revisions

3,000,000+ issue reports
1,000,000+ projects

What is the most used PL

![Bar Chart]

<table>
<thead>
<tr>
<th>Language</th>
<th>Number of Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td>50,692</td>
</tr>
<tr>
<td>c++</td>
<td>40,934</td>
</tr>
<tr>
<td>php</td>
<td>32,696</td>
</tr>
<tr>
<td>c</td>
<td>30,580</td>
</tr>
<tr>
<td>python</td>
<td>15,352</td>
</tr>
<tr>
<td>c#</td>
<td>15,305</td>
</tr>
<tr>
<td>javascript</td>
<td>12,748</td>
</tr>
<tr>
<td>perl</td>
<td>9,783</td>
</tr>
<tr>
<td>unix shell</td>
<td>4,379</td>
</tr>
<tr>
<td>delphi/kylix</td>
<td>3,842</td>
</tr>
</tbody>
</table>
1,000,000,000+ lines of code

How many methods are named "test"?

32,203

How many methods use JUnit’s @Test annotation?

870,181 in 4,578 projects
10,000,000+ revisions

How many words are in log messages?
Running example task

"How many bug fixes add checks for null?"
Has repository?

- Yes
  - Access repository
  - mine revisions

- No

mine project metadata

foreach project

Output count of all null checks

Find null checks in each source

mine source code

Find all Java source files

Fixes bug?

- Yes
A solution in Java...

```java
class AddNullCheck {
    static void main(String[] args) {
        /* create and submit a Hadoop job */
    }
}
static class AddNullCheckMapper 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(Expression exp) {
                if (exp.kind == ExpressionKind.EQ || exp.kind == ExpressionKind.NEQ)
                    for (Expression exp : exp.expressions)
                        if (exp.kind == ExpressionKind.LITERAL && exp.literal.equals("null")) {
                            context.write(new Text("count"), new LongWritable(1));
                            break;
                        }
                }
                }.visit(p);
        }
    }
}
static class AddNullCheckReducer 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
The Boa language and data-intensive infrastructure

http://boa.cs.iastate.edu/

[ICSE’13]
Challenges and Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
Boa architecture

Boa Language
- MapReduce
- Domain-specific Types
- Visitors

Boa's Compiler
- MapReduce
- Quantifiers
- User Functions
- Visitors
- Cached Data input reader
- Runtime

Boa's Data Infrastructure
- SF.net
  - Replicator
  - Caching Translator
- Local Cache

Recall: A solution in Java...

```java
class AddNullCheck {
    static void main(String[] args) {
        ... /* create and submit a Hadoop job */
    }
    static class AddNullCheckMapper 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(Expression e) {
                    if (e.kind == ExpressionKind.EQ || e.kind == ExpressionKind.NEQ)
                        for (Expression exp : e.expressions)
                            if (exp.kind == ExpressionKind.LITERAL && exp.literal.equals("null")) {
                                context.write(new Text("count"), new LongWritable(1));
                                break;
                            }
                        }
                        .visit(p);
                }
            }.visit(p);
        }
    }
    static class AddNullCheckReducer 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));
        }
    }
}
```

Too much code! Do not read!

Full program over 140 lines of code

Uses JSON, SVN, and Eclipse JDT libraries

Uses Hadoop framework

Explicit/manual parallelization
A better solution...

```java
p: Project = input;
count: output sum of int;

visit(p, visitor {
    before e: Expression ->
        if (e.kind == ExpressionKind.EQ || e.kind == ExpressionKind.NEQ)
            exists (i: int; isliteral(e.expressions[i], "null"))
                count << 1;
});
```

Full program **8 lines of code**!

Automatically parallelized!

**No external libraries** needed!

Analyzes **28.8 million** source files in about **15 minutes**!
(only 32 **micro**seconds each!)
p: Project = input;
count: output sum of int;

visit(p, visitor {
    before e: Expression ->
        if (e.kind == ExpressionKind.EQ || e.kind == ExpressionKind.NEQ)
            exists (i: int; isliteral(e.expressions[i], "null"))
                count << 1;
        count << 1;
});

count[] = 120789791

1 + 1 + 1 + 1 + ..
Challenges and Design goals

Easy to use

Scalable and efficient

Reproducible research results
Let's see it in action!

http://boa.cs.iastate.edu/boa/
Why are we waiting for results?

Program is analyzing...

699,331 projects
494,158 repositories
15,063,073 revisions
69,863,970 files
18,651,043,238 AST nodes
Let's check the results!

<<demo>>
Abstracts details of *how* to mine software repositories.
User-defined functions

http://boa.cs.iastate.edu/docs/user-functions.php

```plaintext
id := function (a_1: t_1, ..., a_n: t_n) [: ret] {
    ... # body
    [return ...;]
};
```

Return type is optional

- Allows for complex algorithms and code re-use
- Users can provide their own mining algorithms
Quantifiers

http://boa.cs.iastate.edu/docs/quantifiers.php

```plaintext
foreach (i: int; condition...) body;
```

For each value of i where condition holds, run body
Quantifiers
http://boa.cs.iastate.edu/docs/quantifiers.php

```plaintext
foreach (i: int; condition...) body;
```

For each value of i where condition holds, run body

```plaintext
exists (i: int; condition...) body;
```

If there `exists` a value of i where `condition` holds, run `body`
Quantifiers

http://boa.cs.iastate.edu/docs/quantifiers.php

`foreach (i: int; condition...) body;`

For each value of i where condition holds, run body

`exists (i: int; condition...) body;`

If there exists a value of i where condition holds, run body

`ifall (i: int; condition...) body;`

If for all values of i condition holds, run body
Output and aggregation

http://boa.cs.iastate.edu/docs/aggregators.php

- Output defined in terms of predefined data aggregators
  - sum, set, mean, maximum, minimum, etc

- Values sent to output aggregation variables

```java
p: Project = input;
count: output sum of int;

visit(p, visitor {
    before e: Expression ->
        if (e.kind == ExpressionKind.EQ || e.kind == ExpressionKind.NEQ)
            exists (i: int; isliteral(e.expressions[i], "null"))
            count << 1;
});
```
What about source code?

```java
p: Project = input;
count: output sum of int;

visit(p, visitor {
    before e: Expression ->
        if (e.kind == ExpressionKind.EQ || e.kind == ExpressionKind.NEQ)
            exists (i: int; isliteral(e.expressions[i], "null"))
                count << 1;

});
```
Declarative Visitors in Boa

[GPCE’13]
Basic Syntax

```plaintext
id := visitor {
  before id:T -> statement
  after  id:T -> statement
  ...
};
visit(startNode, id);
```

Execute `statement` either `before` or `after` visiting the children of a node of type `T`
Type Lists and Wildcards

visitor {
    before id:T -> statement
    after T2,T3,T4 -> statement
    after _ -> statement
}

Matching single type (with identifier)

Attributes of the node available via identifier
Type Lists and Wildcards

visitor {
  before id:T -> statement
  after T2,T3,T4 -> statement
  after _ -> statement
}

Type list (no identifier)

Executes statement when visiting nodes of type $T2$, $T3$, or $T4$
Type Lists and Wildcards

```typescript
visitor {
    before id:T   -> statement
    after T2,T3,T4 -> statement
    after _        -> statement
}
```

Wildcard (no identifier)

Executes `statement` for any node not already listed in
another similar clause (e.g., T but not T2/T3/T4)

Provides `default` behavior
Custom Traversals

A -> E -> B -> C -> D

before n: A -> {
  visit(n.E);
  visit(n.B);
  stop;
}
That’s the language…

what can we do with it?
Expressiveness

Treasure study reproduction [Grechanik10]
⇒ 22 tasks

Java language feature adoption
⇒ 18 tasks

Several additional tasks (on Boa website)

[GPCE’13]

[in submission ICSE’14]
How do projects adopt features?
Most features see low use

Research question: are there missed opportunities to use language features?

e.g., Underscore literals

22M occurrences before feature release
2M occurrences after feature release
Is old code refactored to use new features?
Yes!

e.g., diamond pattern (JDK7)

8.5k refactorings detected
3.8k files
72 projects
Challenges and Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
Controlled Experiment
- Subjects shown 5 source code mining tasks in Boa
- Asked to describe (in own words) each task
- Same tasks shown again (random order)
  - Multiple choice this time
- Experiment repeated 6 months later in Hadoop
  - Same tasks
  - Same wording for multiple choice answers
## Source Code Comprehension [3/3]

### Boa Programs

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>-Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>-Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>?</td>
<td>+N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-Y</td>
</tr>
<tr>
<td>N</td>
<td>-Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>+N</td>
<td>-Y</td>
<td>-Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### Hadoop Programs

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Y</td>
<td>-Y</td>
<td>N</td>
<td>-Y</td>
<td>-Y</td>
</tr>
<tr>
<td>?</td>
<td>-Y</td>
<td>-Y</td>
<td>-Y</td>
<td>N</td>
</tr>
<tr>
<td>-Y</td>
<td>Y</td>
<td>+N</td>
<td>Y</td>
<td>-Y</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>-Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>-Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>-Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>-Y</td>
<td>-Y</td>
</tr>
<tr>
<td>-Y</td>
<td>+N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>
### Source Code Comprehension [3/3]

Grading: Use Multiple Choice

<table>
<thead>
<tr>
<th>Boa Programs</th>
<th>Hadoop Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>-Y</td>
<td>Y</td>
</tr>
<tr>
<td>?</td>
<td>Y</td>
</tr>
<tr>
<td>-Y</td>
<td>Y</td>
</tr>
<tr>
<td>?</td>
<td>+N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>-Y</td>
</tr>
<tr>
<td>N</td>
<td>+N</td>
</tr>
</tbody>
</table>

- **77.5%** for Boa Programs
- **62.5%** for Hadoop Programs
Source Code Comprehension [3/3]

Grading: Use Free-form

<table>
<thead>
<tr>
<th>Boa Programs</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>80%</td>
</tr>
<tr>
<td>Q2</td>
<td>-Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>80%</td>
</tr>
<tr>
<td>Q3</td>
<td>?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>80%</td>
</tr>
<tr>
<td>Q4</td>
<td>-Y</td>
<td>?</td>
<td>Y</td>
<td>Y</td>
<td>80%</td>
</tr>
<tr>
<td>Q5</td>
<td>?</td>
<td>+N</td>
<td>Y</td>
<td>Y</td>
<td>60%</td>
</tr>
</tbody>
</table>

**67.5%**

<table>
<thead>
<tr>
<th>Hadoop Programs</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>-Y</td>
<td>-Y</td>
<td>N</td>
<td>-Y</td>
<td>0%</td>
</tr>
<tr>
<td>Q2</td>
<td>?</td>
<td>-Y</td>
<td>-Y</td>
<td>-Y</td>
<td>0%</td>
</tr>
<tr>
<td>Q3</td>
<td>-Y</td>
<td>Y</td>
<td>+N</td>
<td>Y</td>
<td>60%</td>
</tr>
<tr>
<td>Q4</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>-Y</td>
<td>20%</td>
</tr>
<tr>
<td>Q5</td>
<td>N</td>
<td>-Y</td>
<td>N</td>
<td>N</td>
<td>0%</td>
</tr>
</tbody>
</table>

**30%**
Challenges and Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
Efficient execution
Efficient execution

![Chart showing efficient execution comparison between Java and Boa for different project sizes.](chart.png)
Scalability of input size
Scalability of input size
Scales to more cores

Execution time (seconds)

Task A.3
- 1 map: 92
- 2 maps: 36
- 4 maps: 25
- 8 maps: 21
- 16 maps: 22
- 32 maps: 22

Task B.6
- 1 map: 102
- 2 maps: 66
- 4 maps: 40
- 8 maps: 27
- 16 maps: 23

Task C.1
- 1 map: 90
- 2 maps: 57
- 4 maps: 36
- 8 maps: 26
- 16 maps: 24
- 32 maps: 22

Task D.5
- 1 map: 90
- 2 maps: 57
- 4 maps: 35
- 8 maps: 26
- 16 maps: 22
- 32 maps: 21
Optimizations
Computing Cluster
Submit Task
MapReduce, Hadoop, etc
FIFO Queue
T1 T2 T3 T4 T5
Task Result
Computing Cluster

Submit Task

Computing Cluster

MapReduce, Hadoop, etc

Time Sharing

Task Result

T1 T2 T3 T4 T5
Solutions?

● Scale the hardware
  ○ Expensive
  ○ Not always feasible (small businesses, MOOCs, researchers, etc)

● Optimize the software
  ○ Optimize individual tasks
    ■ standard program optimizations
    ■ chain folding [MinerShook12], sibling/MSCR fusion [Chambers10]
  ○ Optimize multiple tasks
    ■ manual job merging [MinerShook12]

[Chambers10] Craig Chambers et al., “FlumeJava”, PLDI 2010
[MinerShook12] Donald Miner and Adam Shook, “MapReduce Design Patterns”, O’Reilly, 2012
Research Questions

1. Can we **automatically** merge related tasks from **different users**?

   **Answer:** Task Fusion  
   [SPLASH’13 SRC]

2. Does *Task Fusion* decrease user wait times in shared computing clusters?
Technical Challenge: map output == side effect
Solution: modify maps to output composite keys

Custom partitioner ensures proper routing

Single, Fused Task

Input → Custom Partitioner → Reduce 1 → Result 1

Input → Custom Partitioner → Reduce 2 → Result 2

Input → Custom Partitioner → Reduce ... → Result ...

Input → Custom Partitioner → Reduce N → Result N
## Results

<table>
<thead>
<tr>
<th>Task Size</th>
<th># of Tasks</th>
<th>Times</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Task Fusion</td>
<td>Task Fusion</td>
</tr>
<tr>
<td>Small(^1)</td>
<td>21</td>
<td>8.1m</td>
<td>0.8m</td>
</tr>
<tr>
<td>Medium(^2)</td>
<td>22</td>
<td>2.3h</td>
<td>1.8h</td>
</tr>
<tr>
<td>Large(^2)</td>
<td>18</td>
<td>4.6h</td>
<td>3.9h</td>
</tr>
<tr>
<td>Mixed(^3)</td>
<td>9</td>
<td>1.3h</td>
<td>0.9h</td>
</tr>
</tbody>
</table>

\(^1\) queries on project and revision metadata only
\(^2\) queries on metadata and millions of source files
\(^3\) 3 small, 3 medium, 3 large
Results

![User Wait Time Improvement Graph](image)

- **Y-axis:** Time (seconds)
- **X-axis:** Number of Tasks Fused
- The graph shows an improvement in user wait time as the number of tasks fused increases.
Can we do better?

Yes! ⇒ Visitor fusion

[to submit PLDI'14]

\[
\text{visit}(p, \text{visitor} \{ \\
\quad \text{before } T1 \rightarrow s1; \\
\quad \text{before } T2 \rightarrow s3; \\
\});
\]

\[
\text{visit}(p, \text{visitor} \{ \\
\quad \text{before } T1 \rightarrow s2; \\
\quad \text{after } T3 \rightarrow s4; \\
\});
\]

\[
\text{visit}(p, \text{visitor} \{ \\
\quad \text{before } T1 \rightarrow \{ \\
\quad \quad s1; \\
\quad \quad s2;
\quad \}); \\
\quad \text{before } T2 \rightarrow s3; \\
\quad \text{after } T3 \rightarrow s4;
\});
\]
## Results

<table>
<thead>
<tr>
<th>Task Size</th>
<th># of Tasks</th>
<th>Times</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Task Fusion</td>
<td>Visitor Fusion</td>
</tr>
<tr>
<td>Medium</td>
<td>22</td>
<td>1.8h</td>
<td>1.8h</td>
</tr>
<tr>
<td>Large</td>
<td>18</td>
<td>3.9h</td>
<td>0.5h</td>
</tr>
<tr>
<td>Mixed</td>
<td>9</td>
<td>0.9h</td>
<td>0.6h</td>
</tr>
</tbody>
</table>
## Combined Results

<table>
<thead>
<tr>
<th>Task Size</th>
<th># of Tasks</th>
<th>Times</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Fusion</td>
<td>Task+Visitor Fusion</td>
</tr>
<tr>
<td>Small(^1)</td>
<td>21</td>
<td>8.1m</td>
<td>0.8m</td>
</tr>
<tr>
<td>Medium(^2)</td>
<td>22</td>
<td>2.3h</td>
<td>1.8h</td>
</tr>
<tr>
<td>Large(^2)</td>
<td>18</td>
<td>4.6h</td>
<td>0.5h</td>
</tr>
<tr>
<td>Mixed(^3)</td>
<td>9</td>
<td>1.3h</td>
<td>0.6h</td>
</tr>
</tbody>
</table>

\(^1\) queries on project and revision metadata only
\(^2\) queries on metadata and millions of source files
\(^3\) 3 small, 3 medium, 3 large
Challenges and Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
Reproducing MSR results

Robles, MSR'10

2/154 experimental papers "replication friendly."

48 due to lack of published data

---

Reproducing MSR: A study of the potential replicability of papers published in the Mining Software Repositories Proceedings

Gregorio Robles
GRC/UVic-Saar
Universidad Rey Juan Carlos
Alcorcón, Madrid, Spain
Email: gre@grc.urjc.es

Abstract—This paper is the result of reviewing all papers published in the proceedings of the former International Workshop on Mining Software Repositories (MSR) (2004-2006) and now Working Conference on MSR (2007-2009). We have analyzed the papers that contained any experimental analysis of software projects for their potential of being replicated. In this regard, three main issues have been addressed: (1) the public availability of the data used in each case study, (2) the public availability of the processed dataset used by researchers and (3) the public availability of the tools and scripts. A total number of 174 papers have been analyzed from the six workshops/workking conferences up to date. Results show that MSR authors use in general publicly available data sources, mainly from free software repositories, but that the amount of publicly available processed datasets is very low. Regarding tools and scripts, for a majority of papers we have not been able to find any tool, even for papers for which the authors explicitly state that they have built one. Lessons learned from the experience of reviewing the whole MSR literature and some potential solutions to lower the barriers of replicability are finally presented and discussed.

Keywords: replication, tools, public datasets, mining software repositories

Replication package: http://grc.urjc.es/~gre/rep MSR 2010

I. INTRODUCTION

Mining software repositories (MSR) has become a fundamental area of research for the Software Engineering community, and of vital importance in the case of empirical studies. Software repositories contain a large amount of valuable information that includes source control systems storing all the history of the source code, defect tracking systems that host defects, enhancements and other issues, and other communication means such as mailing lists or forums. As a result of the possibilities that mining software repositories offer, an annual workshop first, then working conference on this topic has been organized with an extraordinary success in participation and research output.

Being mainly focused on empirical research, we wanted to evaluate how much of the research presented at the MSR can be potentially replicated. Replication is a fundamental task in empirical sciences and one of the main threats to validity that empirical software engineering may suffer [1].

Among these threats, we may encounter: lack of independent validation of the presented results, changes in practices, tools or methodologies, or generalization of knowledge although a limited amount of case studies have been performed. A simple taxonomy of replication studies provides us with two main groups: exact replications and conceptual replications. The former ones are those in which the procedures of an experiment are followed as closely as possible to determine whether the same results can be obtained, while the latter ones are those “one in which the same research question or hypothesis is evaluated by using a different experimental procedure, i.e. many or all of the variables described above are changed.” [2]. In this paper, we will target exact replications in the requirements that have to be met to perform an exact replication are more severe, and in general make a conceptual replication feasible.

We are focusing in this paper on potential replication as we have actually not replicated any of the studies presented in the papers under review. Our aim in this sense is more humble: we want to check if the necessary conditions that make a replication possible are met.

The rest of the paper is structured as follows: in the next section, the method used for this study is presented. Then some general remarks on the MSR conference are given, to give the reader a sense of the type of papers that are published in the MSR proceedings. Results will be presented in section IV: first, the replication-friendliness of the papers will be shown and then each of the individual characteristics that we have defined will be studied independently. MSR has a special track called the “Mining Challenge”, a section is devoted to analyze it with the aim of finding if results differ from those for the rest of the papers. Then, other non-quantitative facts from the review are enumerated. Section VII discusses the findings of the paper and lists at possible solutions. Then, conclusions are drawn. In a final section, the replicability of this paper is considered.

II. METHOD

The method that has been used to perform this study is a complete literature review of the papers published in

---

978-1-4244-6803-4/10/$26.00 © 2010 IEEE
Prior research results are difficult (or impossible) to reproduce.

Boa makes this easier!
Controlled Experiment

- Published artifacts (Boa website):
  - Boa source code
  - Dataset used (timestamp of data)
  - Results

<table>
<thead>
<tr>
<th>Expert</th>
<th>Education</th>
<th>Intro Time</th>
<th>Task 1 Task</th>
<th>Task 1 Time</th>
<th>Task 2 Task</th>
<th>Task 2 Time</th>
<th>Task 3 Task</th>
<th>Task 3 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Post-doc</td>
<td>6</td>
<td>B.1</td>
<td>1</td>
<td>B.6</td>
<td>4</td>
<td>B.9</td>
<td>3</td>
</tr>
<tr>
<td>Yes</td>
<td>PhD</td>
<td>5</td>
<td>A.1</td>
<td>3</td>
<td>B.6</td>
<td>2</td>
<td>B.7</td>
<td>6</td>
</tr>
<tr>
<td>No</td>
<td>PhD</td>
<td>4</td>
<td>B.6</td>
<td>1</td>
<td>B.10</td>
<td>4</td>
<td>B.9</td>
<td>4</td>
</tr>
<tr>
<td>No</td>
<td>PhD</td>
<td>4</td>
<td>A.2</td>
<td>2</td>
<td>B.6</td>
<td>2</td>
<td>D.5</td>
<td>4</td>
</tr>
<tr>
<td>No</td>
<td>MS</td>
<td>4</td>
<td>A.1</td>
<td>4</td>
<td>B.6</td>
<td>1</td>
<td>D.3</td>
<td>2</td>
</tr>
<tr>
<td>No</td>
<td>MS</td>
<td>3</td>
<td>B.6</td>
<td>2</td>
<td>C.1</td>
<td>2</td>
<td>D.4</td>
<td>10</td>
</tr>
<tr>
<td>No</td>
<td>MS</td>
<td>6</td>
<td>A.1</td>
<td>2</td>
<td>B.7</td>
<td>3</td>
<td>B.10</td>
<td>3</td>
</tr>
<tr>
<td>No</td>
<td>BS</td>
<td>2</td>
<td>A.2</td>
<td>2</td>
<td>D.1</td>
<td>2</td>
<td>D.3</td>
<td>2</td>
</tr>
</tbody>
</table>

Fig. 16. Study results. All times given in minutes.
Ongoing work

Infrastructure improvements

Language abstractions

GitHub

Google Code

Launchpad

Other artifacts

Language abstractions

cvs

git

hg

bzt
Boa
http://boa.cs.iastate.edu/

- Domain-specific language and infrastructure for software repository mining that is:
  - Easy to use
  - Efficient and scalable
  - Amenable to reproducing prior results
Related - MSR

Sourcerer  
  Linstead et.al. 2009

PROMISE
  Menzies et.al. 2009

Kenyon
  Bevan et.al 2005
Related - Data-Parallel

MapReduce
  Dean and Ghemawat 2004

Hadoop

Dryad
  Isard et.al. 2007
Related - Data-Parallel

Sawzall
  Pike et.al. 2005

PigLatin
  Olston et.al. 2008

FlumeJava
  Chambers et.al. 2010
Related - Visitors

GOF Visitor pattern 1994

DemeterJ/DJ
Orleans and Lieberherr 2001

Recursive Traversals
Ovlinger and Wand 1999
Related - Studies

Java generics
  Parnin et.al. 2011

Treasure
  Grechanik et.al 2010

Language adoption
  Meyerovich and Rabkin 2013
Related - Optimizations

Chain folding, job merging
   Miner and Shook 2012

sibling fusion, MSCR fusion (FlumeJava)
   Chambers et.al. 2012

ChainMapper/ChainReducer (Hadoop)
Domain-specific functions

Mines a revision to see if it contains any files of the type specified.

```hasfiletype := function (rev: Revision, ext: string) : bool {
    exists (i: int; match(format(`\.%s$`, ext), rev.files[i].name))
    return true;
    return false;
};
```

Mines a revision to see if it contains any files of the type specified.
Domain-specific functions

http://boa.cs.iastate.edu/docs/dsl-functions.php

```dynamorio
isfixingrevision := function (log: string) : bool {
    if (match(`\bfix(s|es|ing|ed)?\b`, log)) return true;
    if (match(`\b(error|bug|issue)(s)` \b`, log)) return true;
    return false;
}
```

Mines a revision log to see if it fixed a bug.