Mining Ultra-Large-Scale Software Repositories with

Boa

Robert Dyer, Hoan Nguyen, Hridesh Rajan, and Tien Nguyen

{rdyer,hoan,hridesh,tien}@iastate.edu

Iowa State University

Why mine software repositories?

Why mine software repositories?

Learn from the past

What is actually practiced

Spot anti-patterns

Why mine software repositories?

Learn from the past

Why mine software repositories?

Learn from the past → **Inform the future**

Keep doing what works

To find better designs

Empirical validation

Why mine software repositories?

Learn from the past → **Inform the future**

Open source repositories

Google code

CodePlex
Open Source Project Community

Apache

github
SOCIAL CODING

Java.net

SOURCEFORGE.NET®

Atlassian
bitbucket

launchpad

Open source repositories

1,000,000+ projects

1,000,000,000+ lines of code

10,000,000+ revisions

3,000,000+ issue reports

Open source repositories

1,000,000+ projects

What is the most used PL?

1,000,000,000+ lines of code

How many methods are named "test"?

10,000,000+ revisions

How many words are in log messages?

3,000,000+ issue reports

How many issue reports have duplicates?

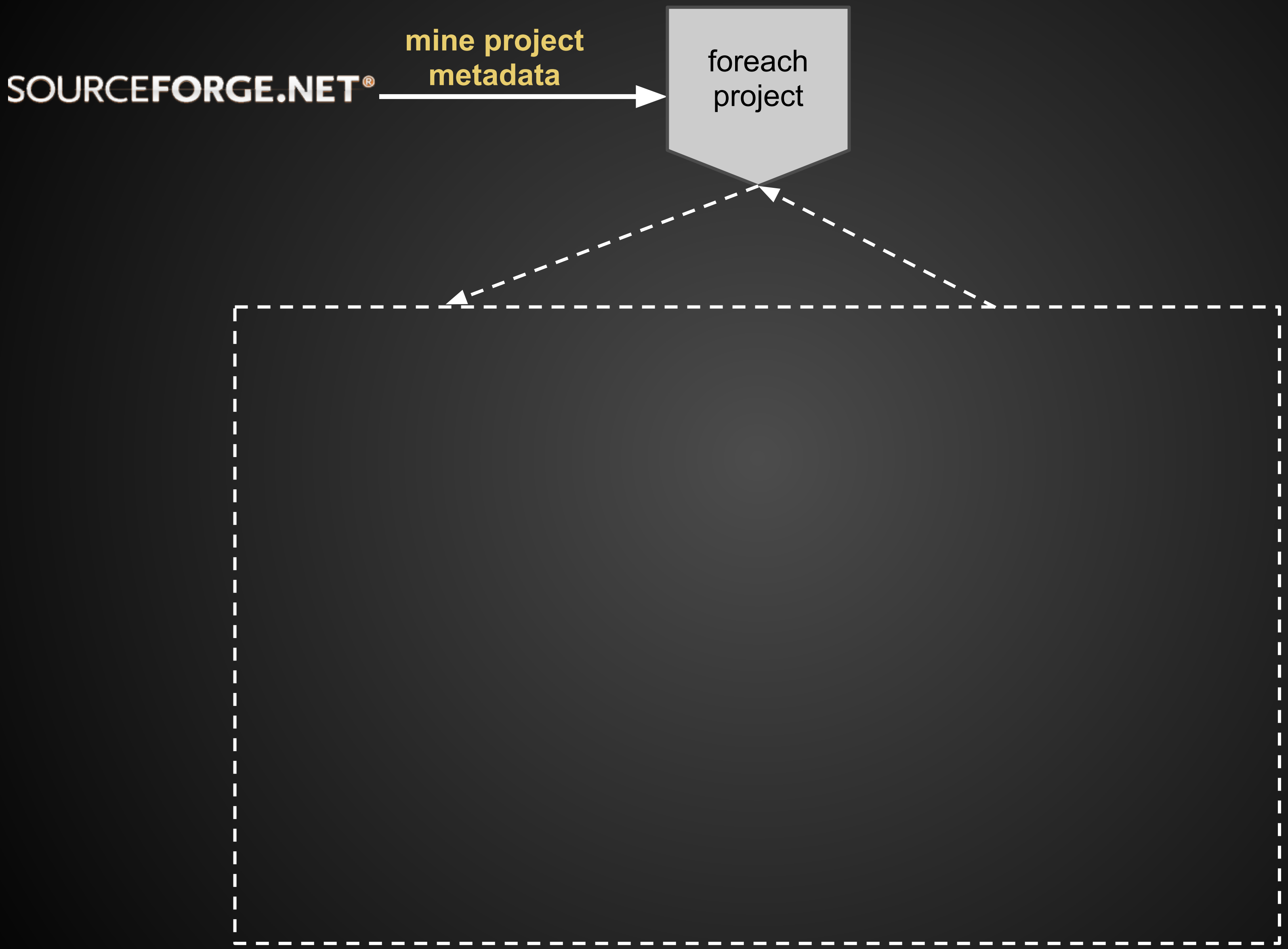
Consider a task that answers

"What is the average churn rate for Java projects on SourceForge?"

Note: churn rate is the average number of files changed per revision

SOURCEFORGE.NET®

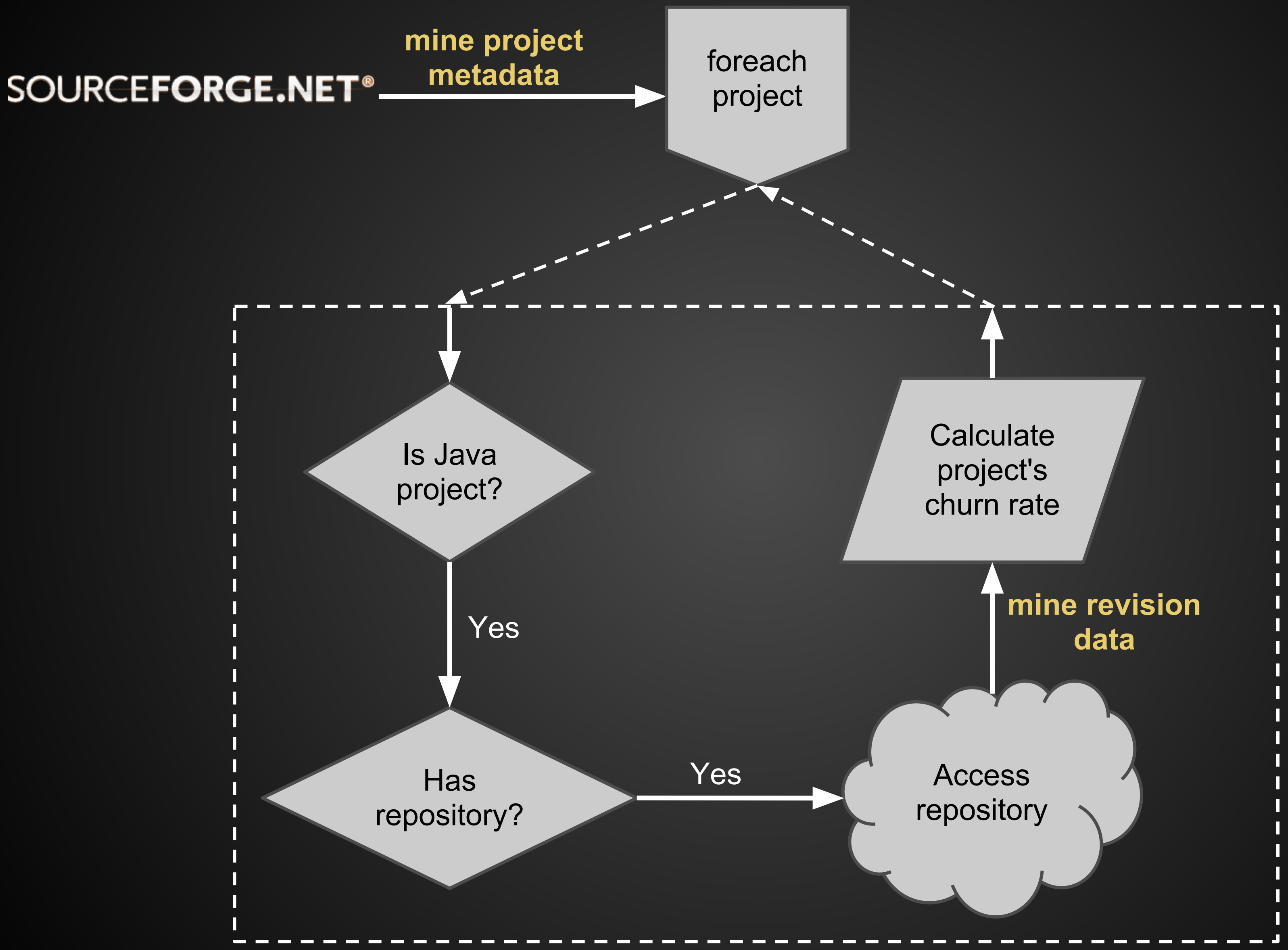
SOURCEFORGE.NET®
mine project metadata

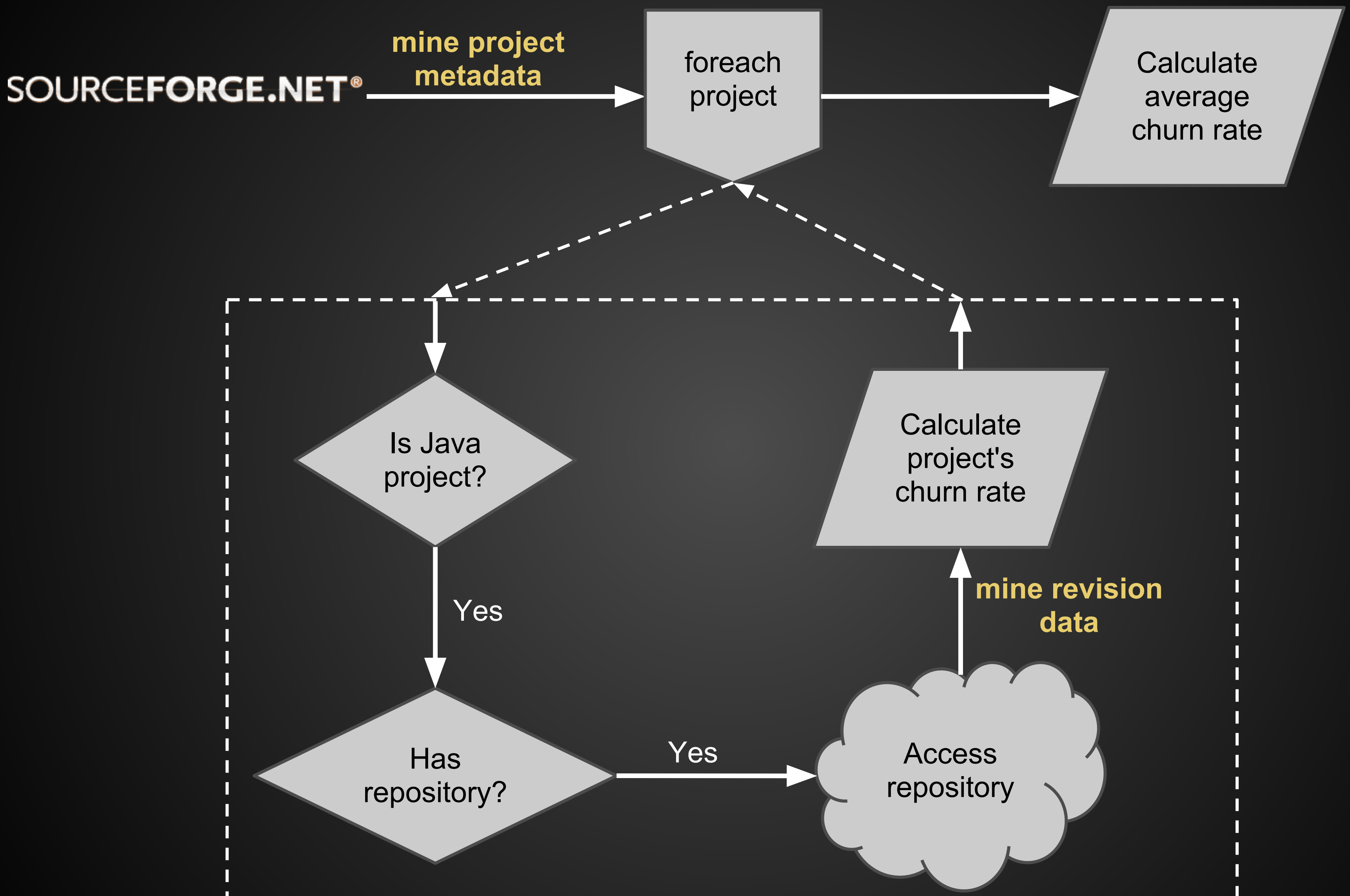


SOURCEFORGE.NET®

mine project metadata

foreach project


SOURCEFORGE.NET®
mine project metadata
foreach project
Is Java project?
Yes
Has repository?
Yes
Access repository
mine revision data
Calculate project's churn rate


SOURCEFORGE.NET®

mine project metadata

foreach project

Calculate average churn rate

Is Java project?

Yes

Has repository?

Yes

Access repository

mine revision data

Calculate project's churn rate

A solution in Java...

```
public class GetChurnRates {
    public static void main(String[] args) { new GetChurnRates().getRates(args[0]); }
    public void getRates(String cachePath) {
        for (File file : (File[])FileIO.readObjectFromFile(cachePath)) {
            String url = getSVNUrl(file);
            if (url != null && !url.isEmpty())
                System.out.println(url + "," + getChurnRateForProject(url));
        }
    }

    private String getSVNUrl(File file) {
        String jsonTxt = "";
        ... // read the file contents into jsonTxt
        JSONObject json = null, jsonProj = null;
        ... // parse the text, get the project data
        if (!jsonProj.has("programming-languages")) return "";
        if (!jsonProj.has("SVNRepository")) return "";
        boolean hasJava = false;
        ... // is the project a Java project?
        if (!hasJava) return "";
        JSONObject svnRep = jsonProj.getJSONObject("SVNRepository");
        if (!svnRep.has("location")) return "";
        return svnRep.getString("location");
    }

    private double getChurnRateForProject(String url) {
        double rate = 0;
        SVNURL svnUrl;
        ... // connect to SVN and compute churn rate
        return rate;
    }
}
```

Too much code!

Do not read

Full program
over 70 lines of code

Uses ***JSON and SVN libraries***

Runs ***sequentially***

Takes ***over 24 hrs***

Takes ***almost 3 hrs*** - *with data locally cached*!

A better solution...

```
rates: output mean[string] of int;
p: Project = input;

when (i: some int; match(`^java$`, lowercase(p.programming_languages[i])))
     when (j: each int; p.code_repositories[j].repository_type == RepositoryType.SVN)
          when (k: each int; def(p.code_repositories[j].revisions[k]))
               rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

Full program **6 lines of code**!

Automatically parallelized!

No external libraries needed!

Results in about **1 minute**!

The Boa language and data-intensive infrastructure

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

Design goals

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

Design goals

➡ Easy to use

- Simple language
- No need to know details of
  - Software repository mining
  - Data parallelization

Design goals

➡ Scalable and efficient

- Study ***millions*** of projects
- Results in minutes, not days

Design goals

Reproducible research results

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

Gregorio Robles
GSyC/LibreSoft
Universidad Rey Juan Carlos
Madrid, Spain
Email: grex@gsyc.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 potentiality of being replicated. In this regard, three main issues have been addressed: i) the public availability of the data used as case study, ii) the public availability of the processed dataset used by researchers and iii) the public availability of the tools and scripts. A total number of 171 papers have been analyzed from the six workshops/working 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 where 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://gsyc.urjc.es/~grex/msr2010.

I. INTRODUCTION

Mining software repositories (MSR) has become a fundamental area of research for the Software Engineering

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 as 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.

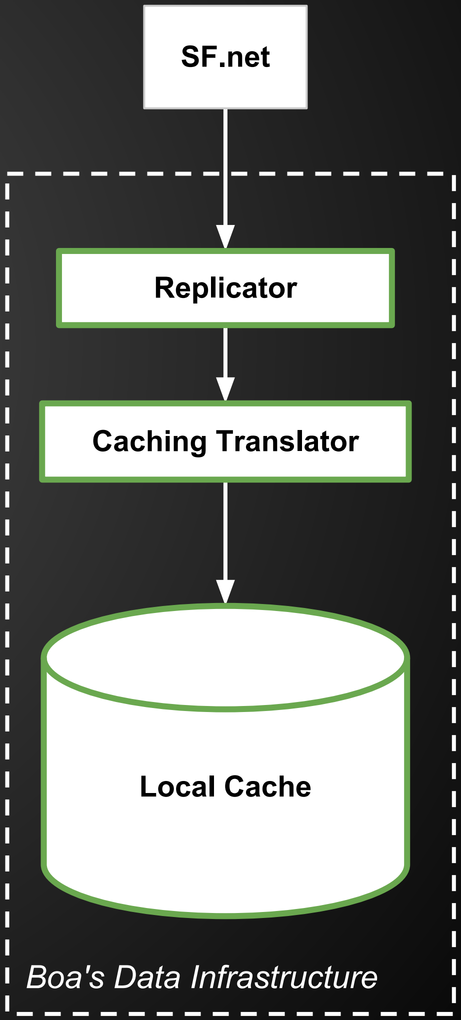
The rest of the paper is structured as follows: in the next section, the method used for this study is presented. Then some general remarks on the MSR conference are given, to give the reader a sense of the type of papers that are

Robles, MSR'10

Studied 171 papers

Only 2 were "replication friendly"

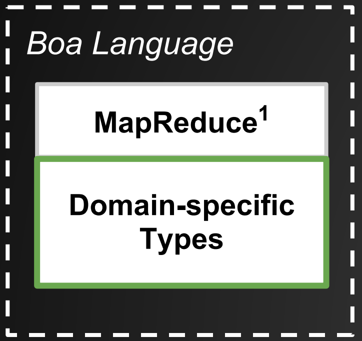
Boa architecture



[1] Pike et al, Scientific Prog. Journal, Vol 13, No 4, 2005

[2] Anthony Urso, http://github.com/anthonyu/Sizzle

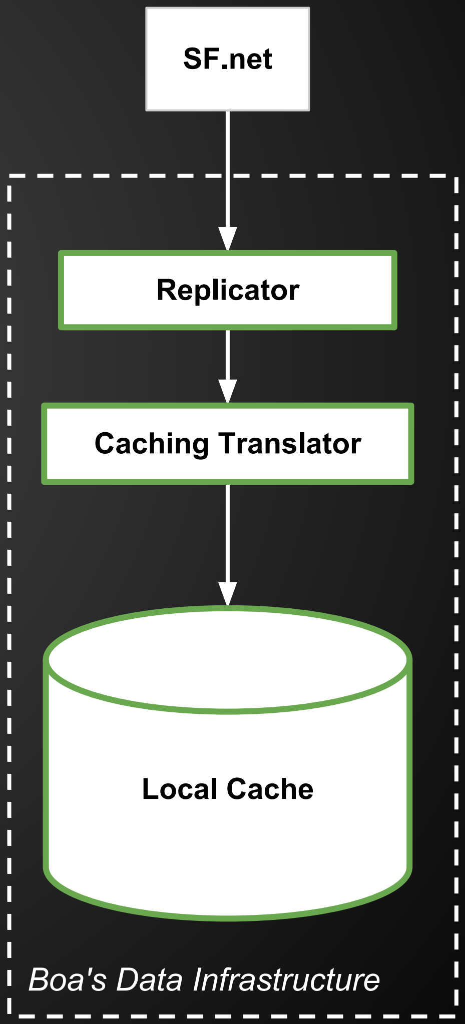
Boa architecture



Boa Language

MapReduce[1]

Domain-specific Types



SF.net

Replicator

Caching Translator

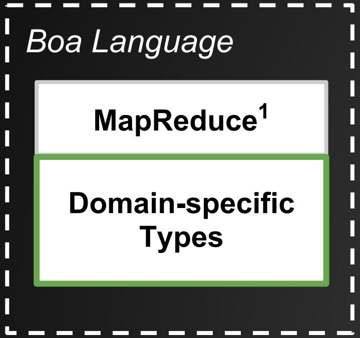
Local Cache

Boa's Data Infrastructure

[1] Pike et al, Scientific Prog. Journal, Vol 13, No 4, 2005

[2] Anthony Urso, http://github.com/anthonyu/Sizzle

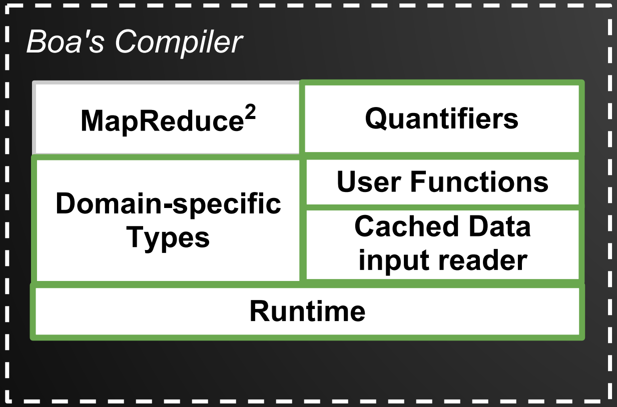
Boa architecture



Boa Language

MapReduce[1]

Domain-specific Types



Boa's Compiler

MapReduce[2]

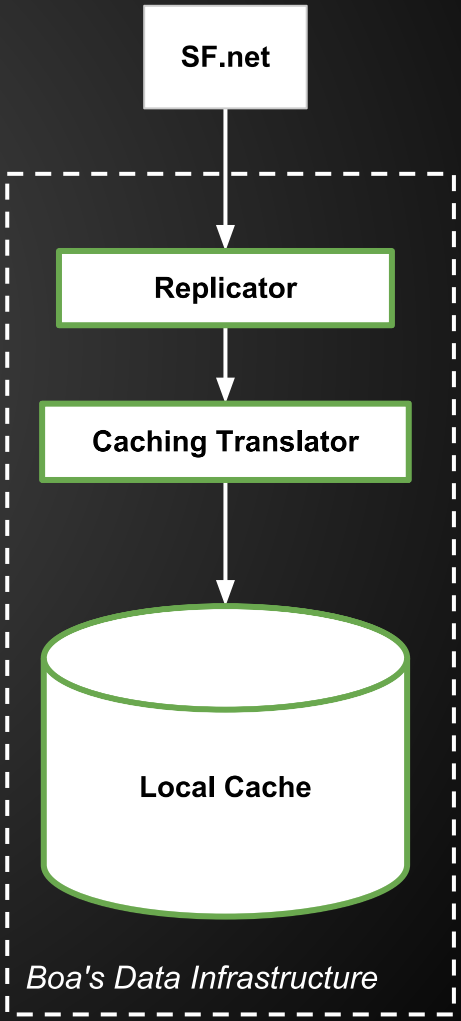
Quantifiers

Domain-specific Types

User Functions

Cached Data input reader

Runtime



SF.net

Replicator

Caching Translator

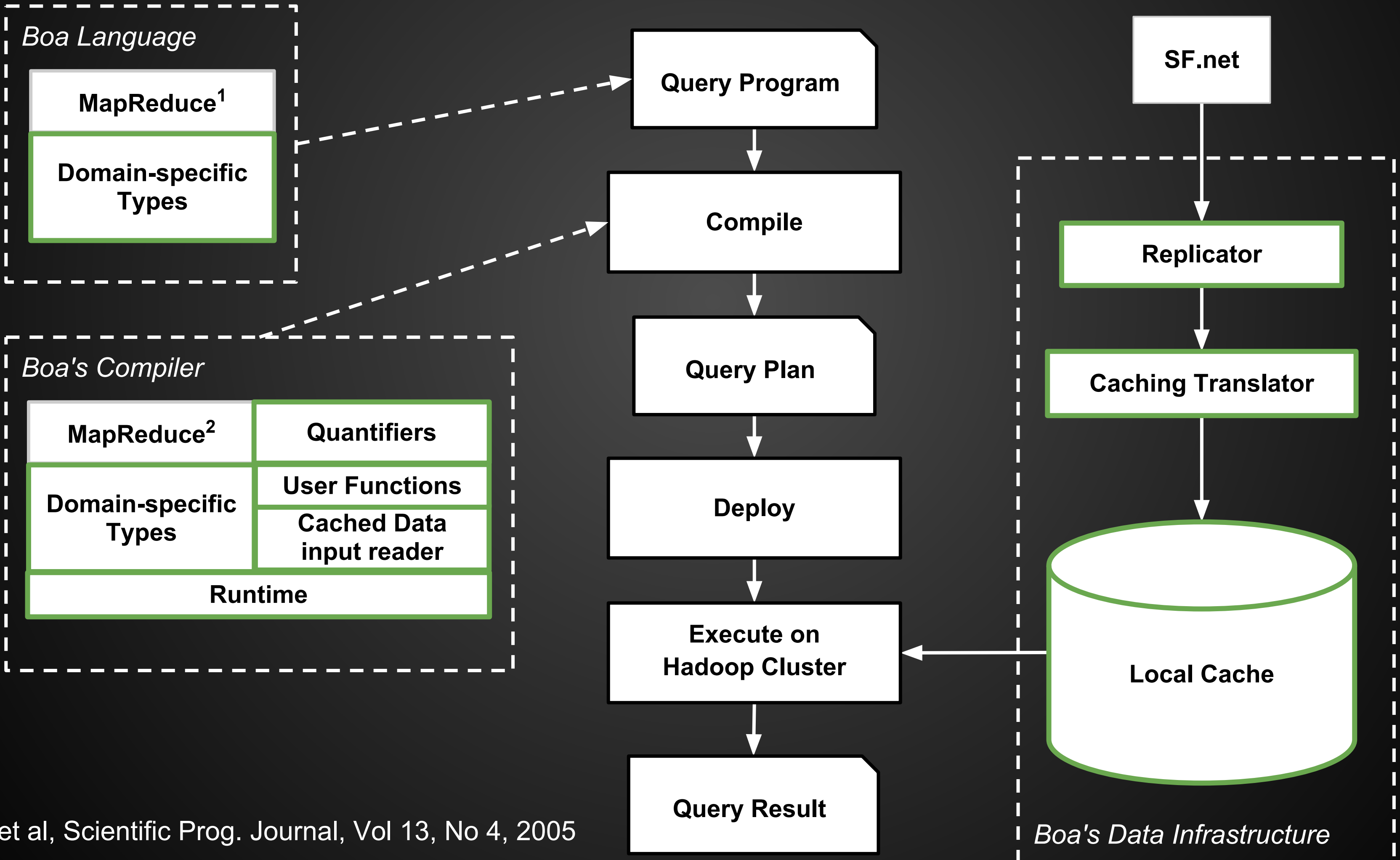
Local Cache

Boa's Data Infrastructure

[1] Pike et al, Scientific Prog. Journal, Vol 13, No 4, 2005

[2] Anthony Urso, http://github.com/anthonyu/Sizzle

Boa architecture



[1] Pike et al, Scientific Prog. Journal, Vol 13, No 4, 2005

[2] Anthony Urso, http://github.com/anthonyu/Sizzle

Domain-specific types

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

```
rates: output mean[string] of int;
p: Project = input;

when (i: some int; match(`^java$`, lowercase(p.programming_languages[i])))
     when (j: each int; p.code_repositories[j].repository_type == RepositoryType.SVN)
          when (k: each int; def(p.code_repositories[j].revisions[k]))
               rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

Abstracts details of *how* to mine software repositories

Domain-specific types

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

Project

id	:	`string`
name	:	`string`
description	:	`string`
homepage_url	:	`string`
programming_languages	:	`array of string`
licenses	:	`array of string`
maintainers	:	`array of Person`
	....	
code_repositories	:	`array of CodeRepository`

Domain-specific types

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

CodeRepository

url	:	`string`
repository_type	:	`RepositoryType`
revisions	:	`array of Revision`

Revision

id	:	`int`
author	:	`Person`
committer	:	`Person`
commit_date	:	`time`
log	:	`string`
files	:	`array of File`

File

name	:	`string`

Domain-specific functions

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

```
hasfiletype := function (rev: Revision, ext: string) : bool {
    when (i: some int; matches(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

```
isfixingrevision := function (log: string) : bool {
    if (matches(`\s+fix(es|ing|ed)?\s+`, log))             return true;
    if (matches(`(bug|issue)(s)?[\s]+(#)?\s*[0-9]+`, log)) return true;
    if (matches(`(bug|issue)\s+id(s)?\s*=\s*[0-9]+`, log)) return true;
    return false;
}
```

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

User-defined functions

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

```
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 and when statements

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

```
rates: output mean[string] of int;
p: Project = input;

when (i: some int; match(`^java$`, lowercase(p.programming_languages[i])))
     when (j: each int; p.code_repositories[j].repository_type == RepositoryType.SVN)
          when (k: each int; def(p.code_repositories[j].revisions[k]))
               rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

- Easily expresses loops over data
- Bounds are inferred from condition

Quantifiers and when statements

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

```
when (i: each int; condition...)
    body;
```

For *each* value of **i**,

if **condition** holds
then
run **body** (with i bound to the value)

Quantifiers and when statements

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

```
when (i: some int; condition...)
    body;
```

For *some* value of **i**,

if **condition** holds
then
run **body** *once* (with i bound to the value)

Quantifiers and when statements

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

```
when (i: all int; condition...)
    body;
```

For *all* values of **i**,

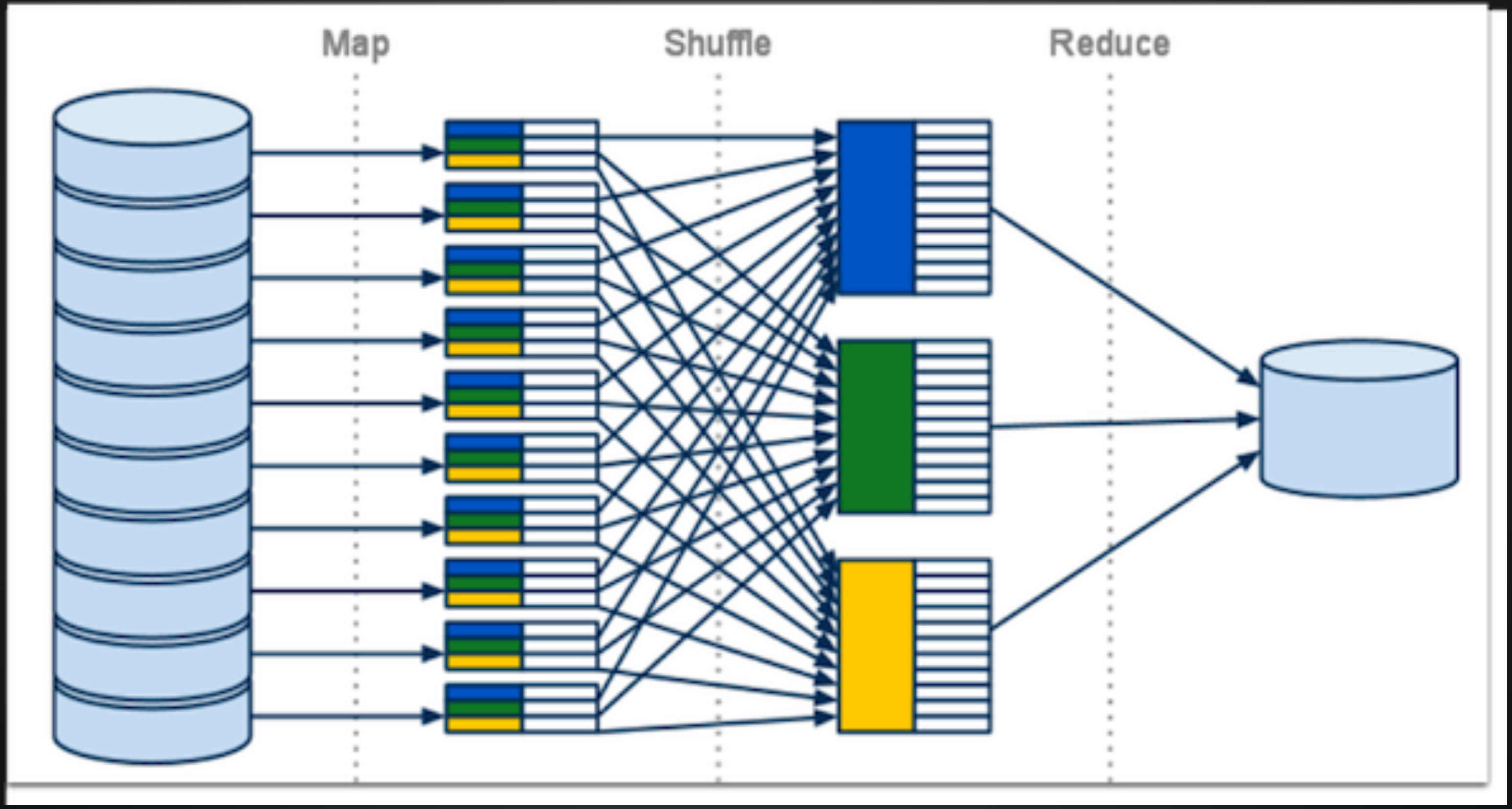
if **condition** holds
then
run **body** *once* (with i not bound)

Output and aggregation

- Boa uses MapReduce [Dean & Ghemawat 2004]
- Most details abstracted from users

What is MapReduce?

Output and aggregation



source: https://developers.google.com/appengine/docs/python/dataprocessing/overview

Output and aggregation

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

```
rates: output mean[string] of int;
p: Project = input;

when (i: some int; match(`^java$`, lowercase(p.programming_languages[i])))
     when (j: each int; p.code_repositories[j].repository_type == RepositoryType.SVN)
          when (k: each int; def(p.code_repositories[j].revisions[k]))
               rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

- Output defined in terms of predefined data aggregators
    - sum, set, mean, maximum, minimum, etc
- Values sent to output aggregation variables
- Output can be indexed

Let's see it in action!

<<demo>>

Why are we waiting for results?

Program is analyzing...

621,671 projects

370,554 repositories

4,137,763 revisions

39,629,911 files

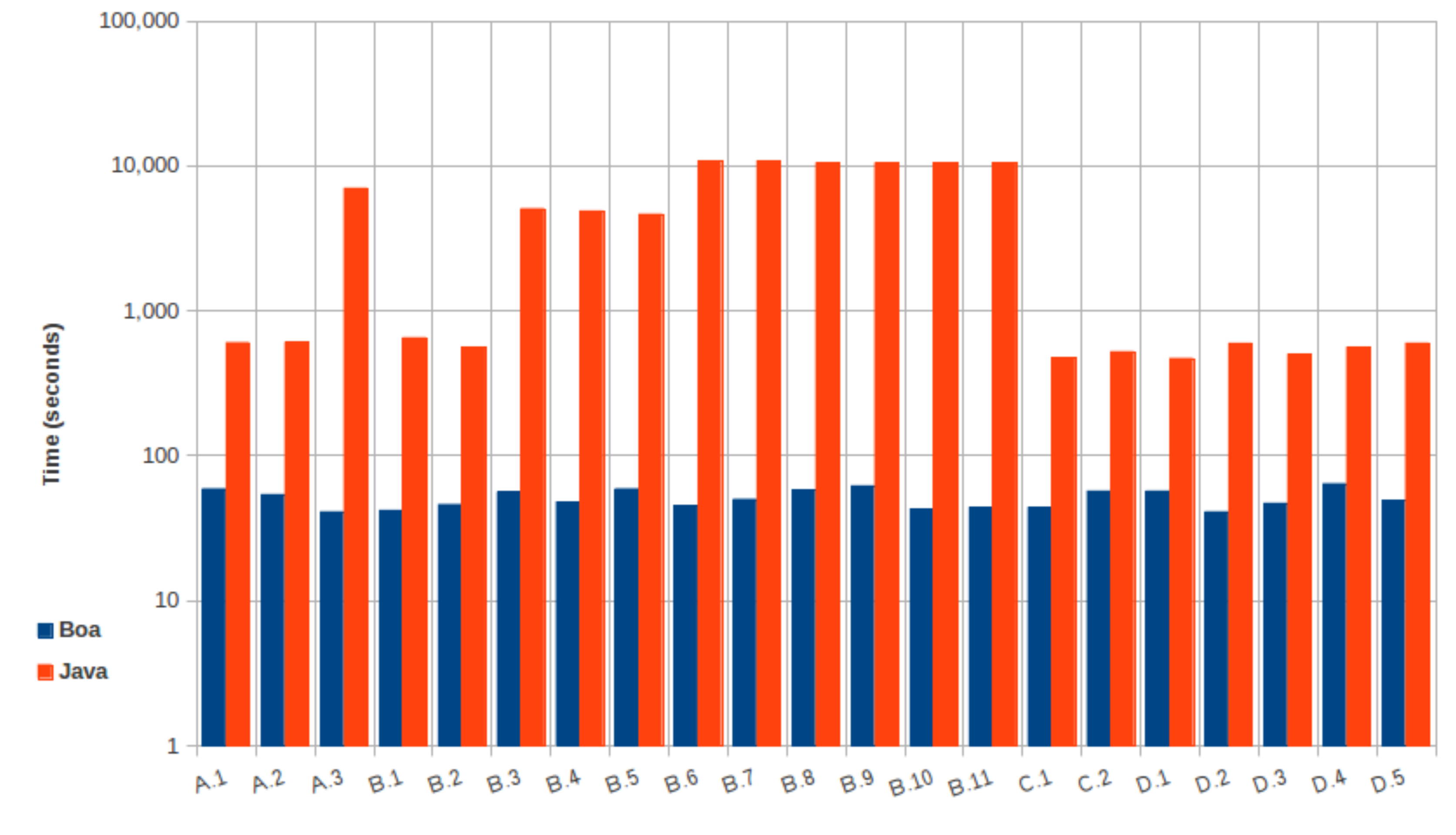
Let's check the results!

<<demo>>

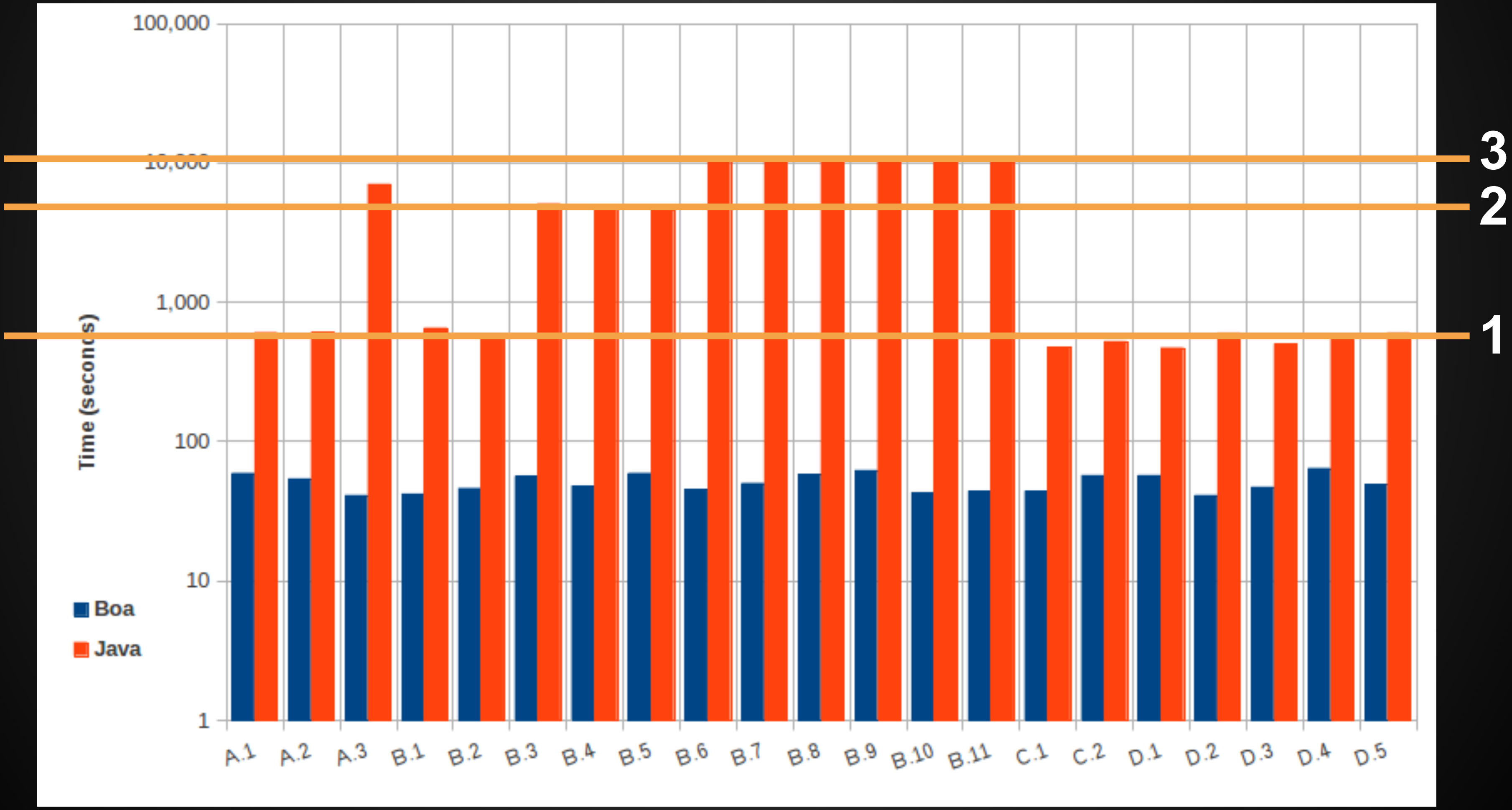
Efficient execution

Time (seconds)
100,000
10,000
1,000
100
10
1
■ Boa
■ Java
A.1
A.2
A.3
B.1
B.2
B.3
B.4
B.5
B.6
B.7
B.8
B.9
B.10
B.11
C.1
C.2
D.1
D.2
D.3
D.4
D.5

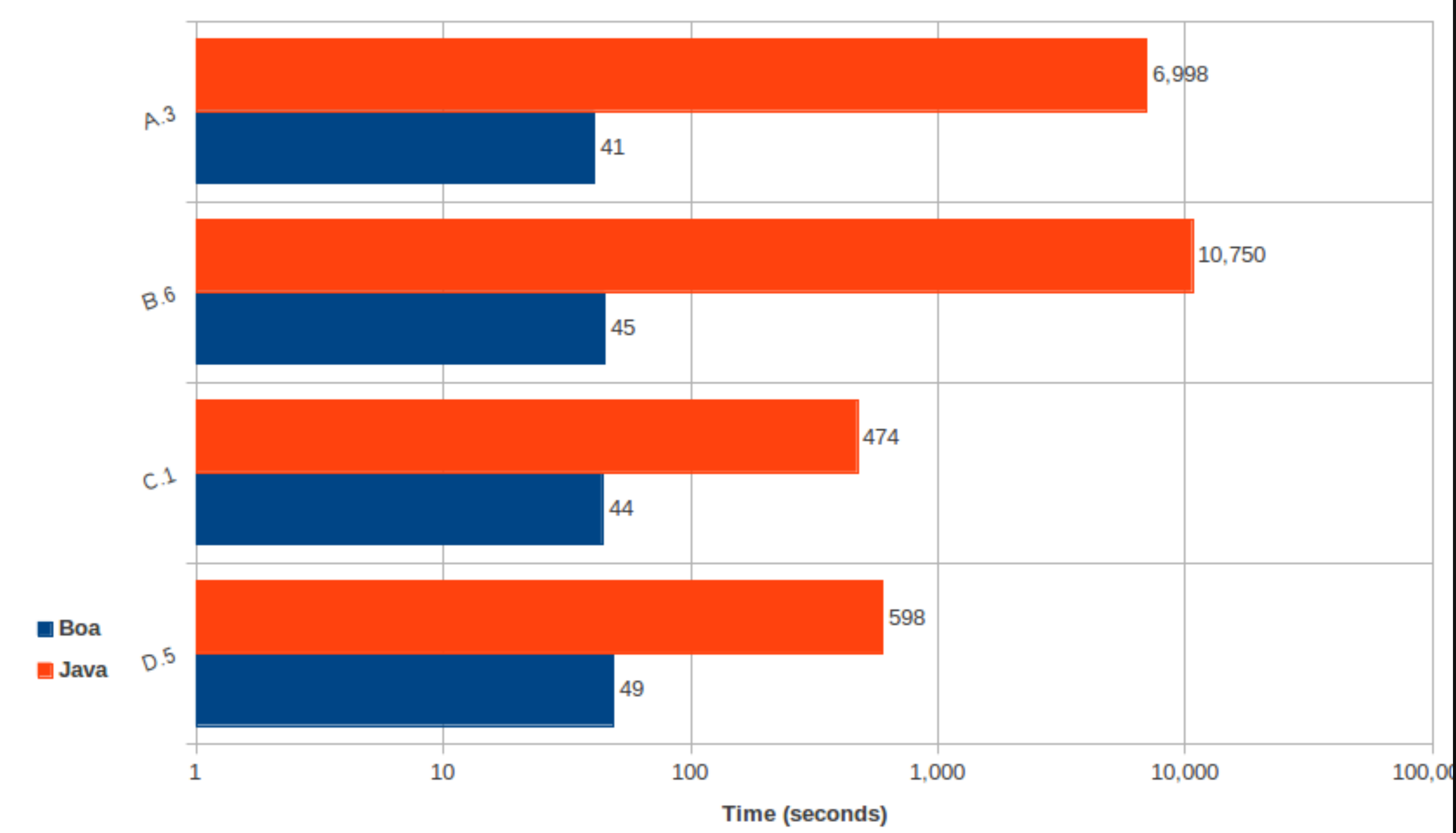
Efficient execution



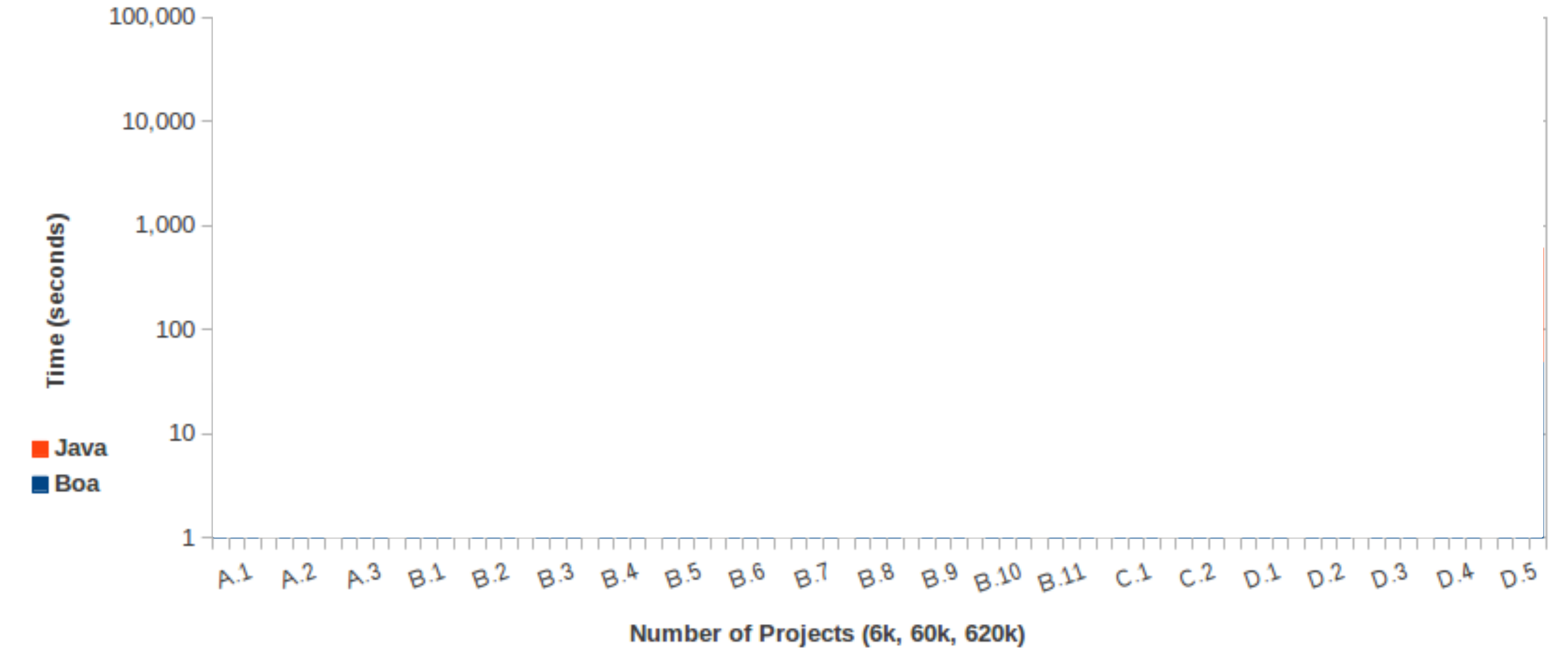
Efficient execution



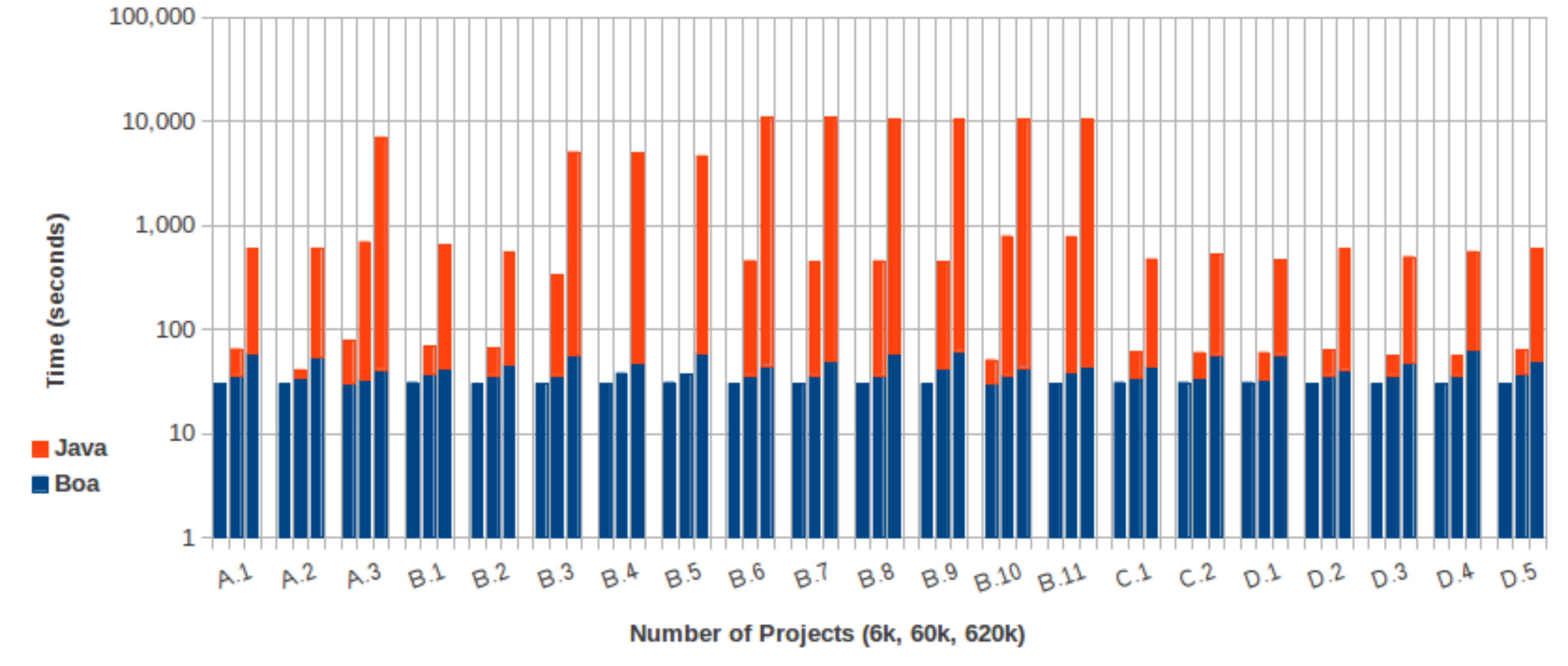
Efficient execution



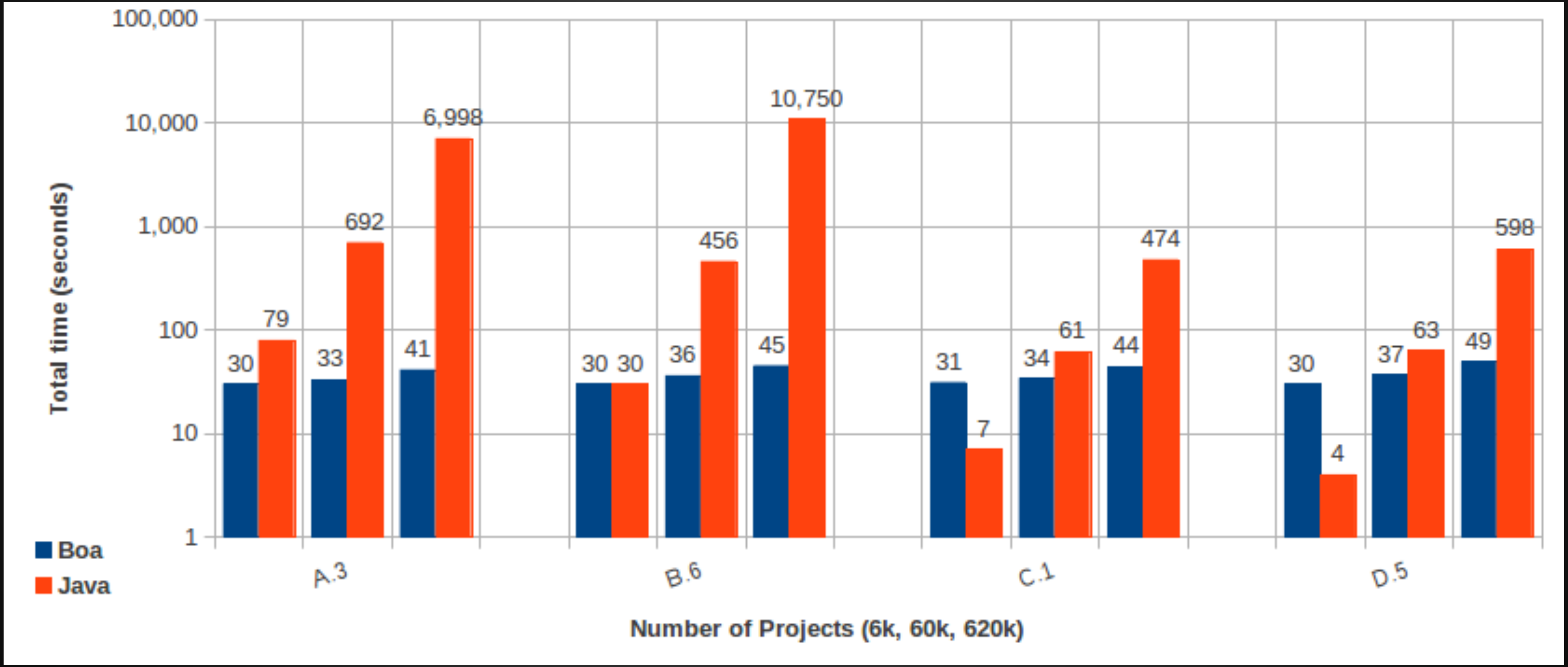
Scalability of input size



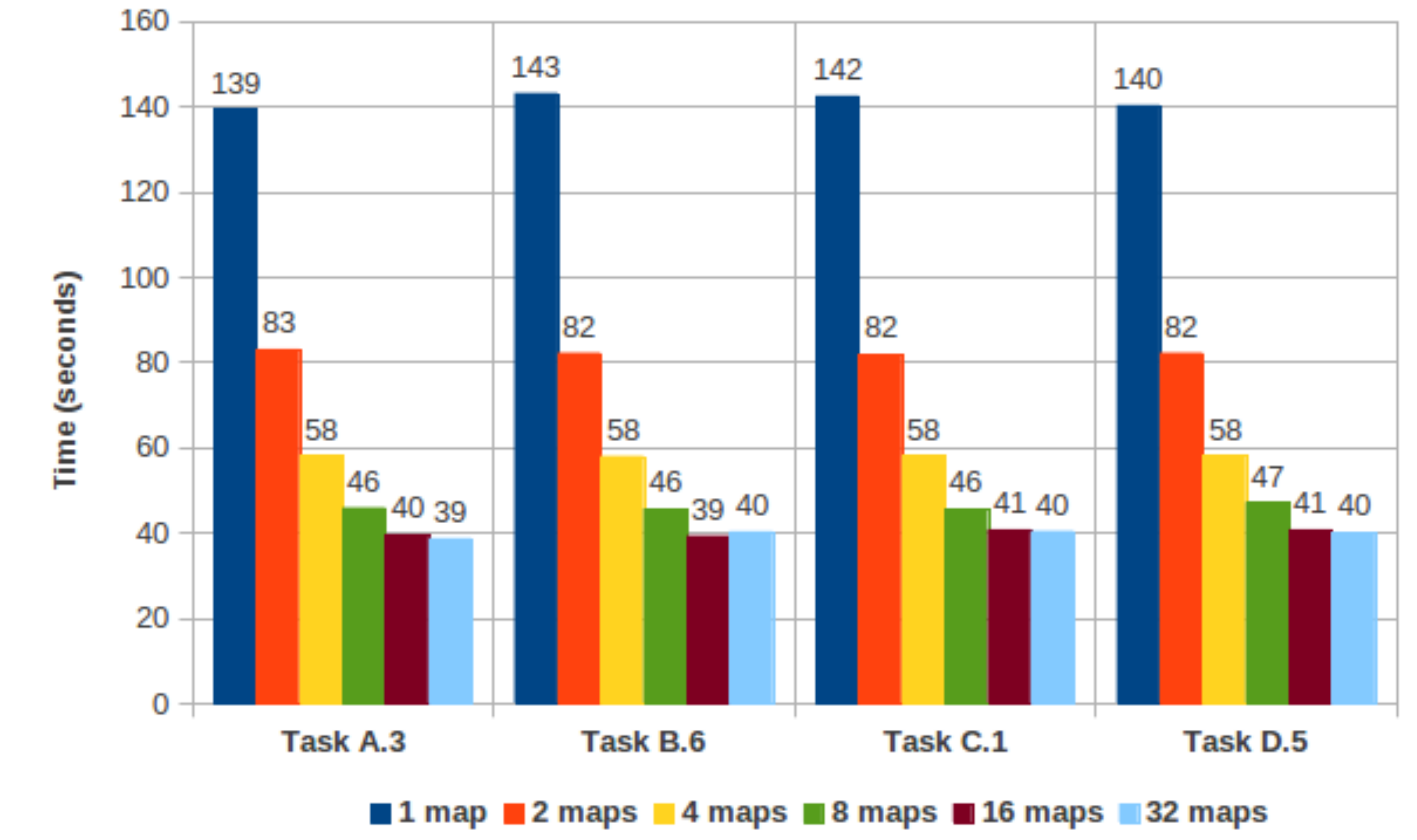
Scalability of input size



Scalability of input size



Scales to more cores



Reproducing MSR results

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

Gregorio Robles
GSyC/LibreSoft
Universidad Rey Juan Carlos
Madrid, Spain
Email: grex@gsyc.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 potentiality of being replicated. In this regard, three main issues have been addressed: i) the public availability of the data used as case study, ii) the public availability of the processed dataset used by researchers and iii) the public availability of the tools and scripts. A total number of 171 papers have been analyzed from the six workshops/working 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 where 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://gsyc.urjc.es/~grex/msr2010.

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 as 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 papers. Then, other non-quantitative facts from the review are enumerated. Section VII discusses the findings of the paper and hints 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

Robles, MSR'10

2/154 experimental papers "replication friendly."

48 due to lack of published data

Prior research results are difficult (or impossible) to reproduce.

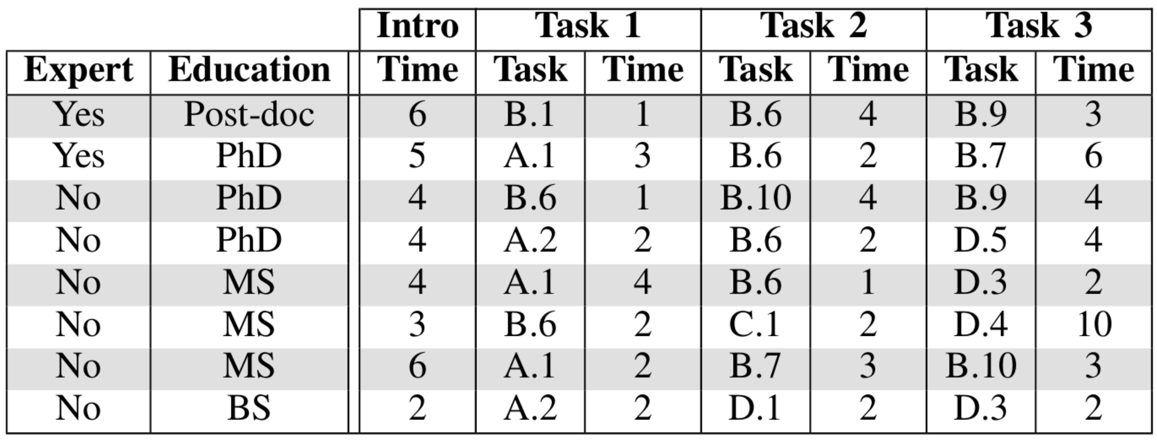
Boa makes this easier!

Let's reproduce some prior results!

<<demo>>

Controlled Experiment

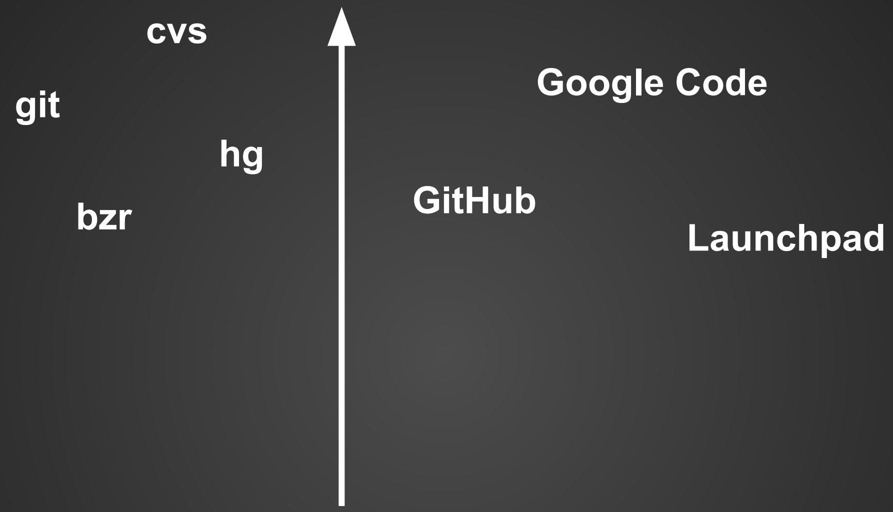
- Published artifacts (Boa website):
  - Boa source code
  - Dataset used (timestamp of data)
  - Results



		Intro	Task 1		Task 2		Task 3	
Expert	**Education**	**Time**	**Task**	**Time**	**Task**	**Time**	**Task**	**Time**
Yes	Post-doc	6	B.1	1	B.6	4	B.9	3
Yes	PhD	5	A.1	3	B.6	2	B.7	6
No	PhD	4	B.6	1	B.10	4	B.9	4
No	PhD	4	A.2	2	B.6	2	D.5	4
No	MS	4	A.1	4	B.6	1	D.3	2
No	MS	3	B.6	2	C.1	2	D.4	10
No	MS	6	A.1	2	B.7	3	B.10	3
No	BS	2	A.2	2	D.1	2	D.3	2

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

Ongoing work



cvs

git

hg

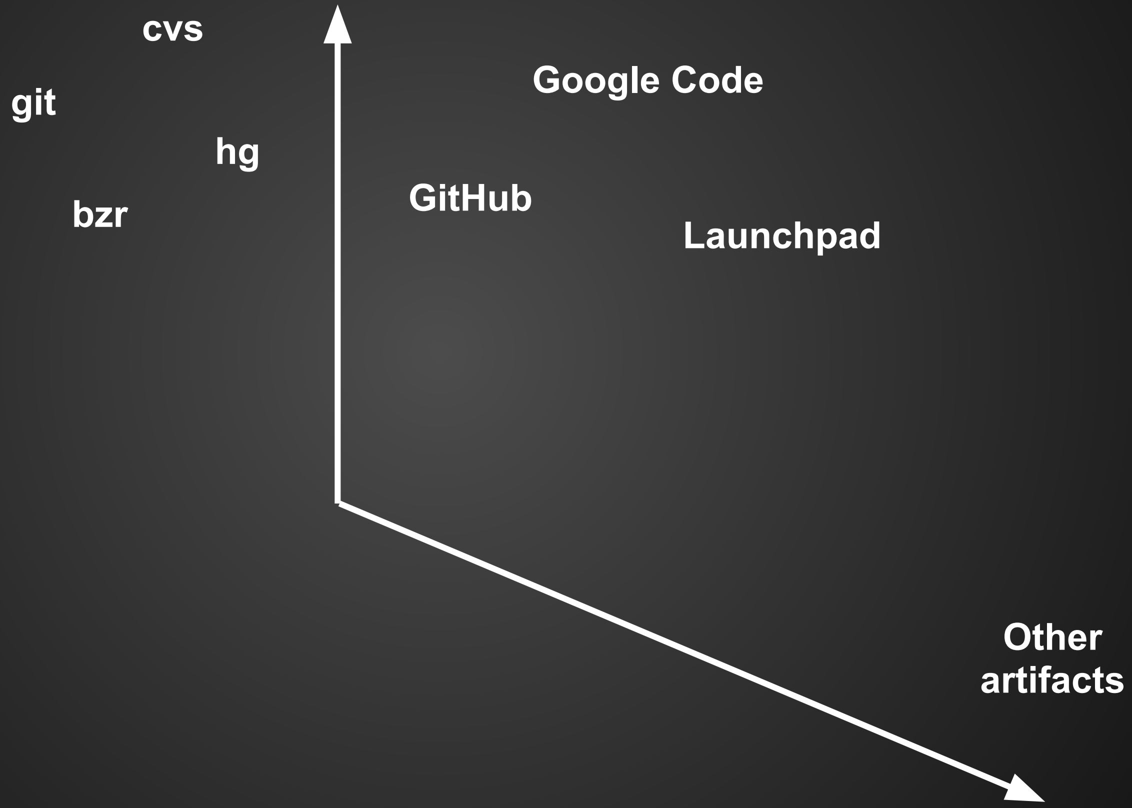
bzr

Google Code

GitHub

Launchpad

Ongoing work



cvs

git

hg

bzr

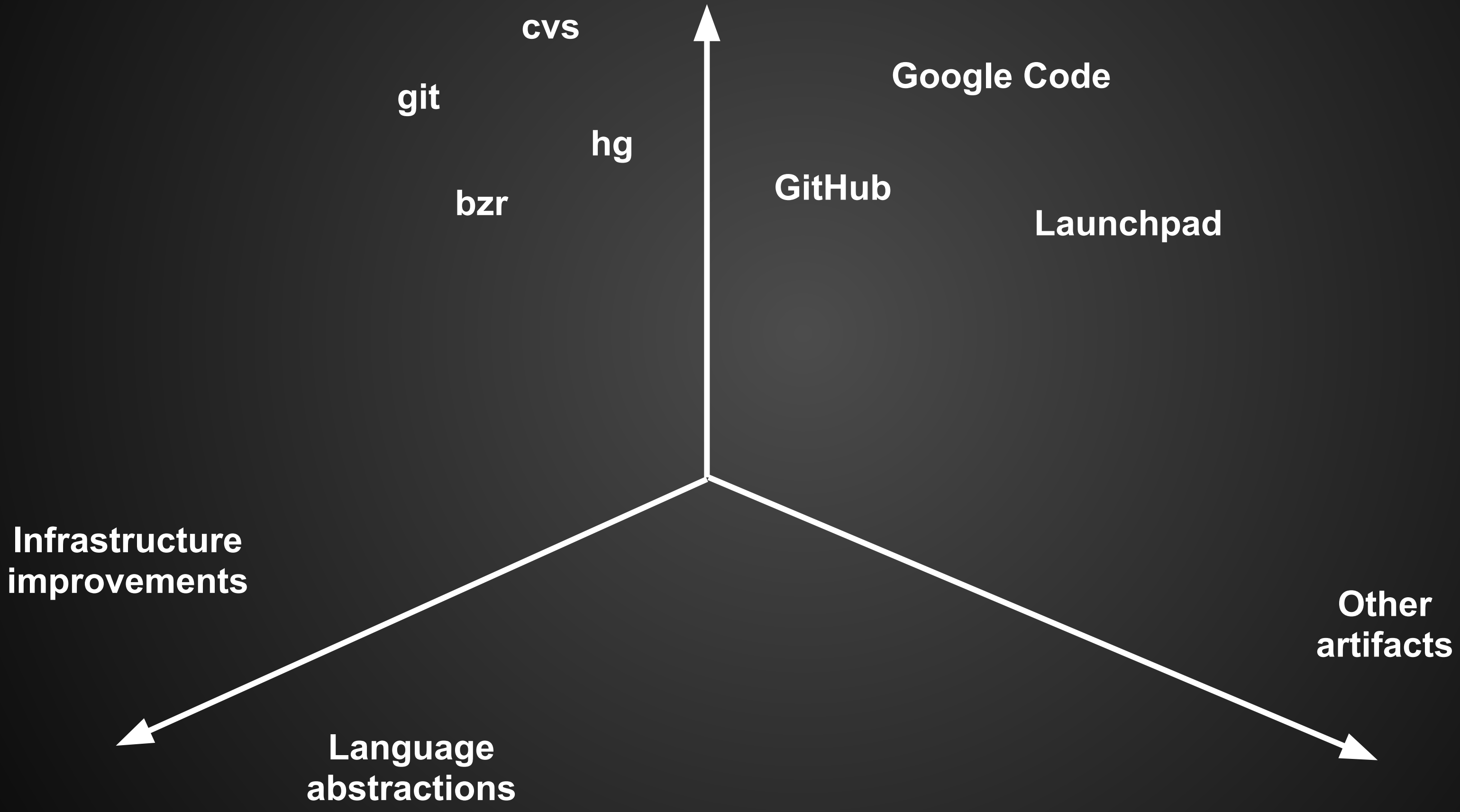
Google Code

GitHub

Launchpad

Other artifacts

Ongoing work



cvs

git

hg

bzr

Google Code

GitHub

Launchpad

Infrastructure improvements

Other artifacts

Language abstractions

Conclusions

- Domain-specific language and infrastructure for software repository mining
  - Easy to use
  - Efficient and scalable
  - Allows reproducing prior results

For more information...

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