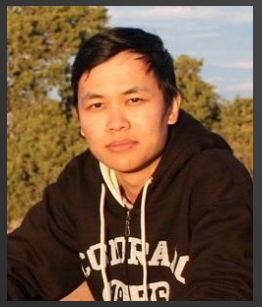
Boa

A Language and Infrastructure for Analyzing Ultra-Large-Scale Software Repositories



Robert Dyer Hoan Anh Nguyen Hridesh Rajan Tien N. Nguyen

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

Iowa State University

What is actually practiced

Keep doing what works

To find better designs

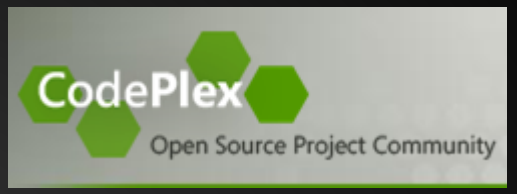
Empirical validation

Spot (anti-)patterns

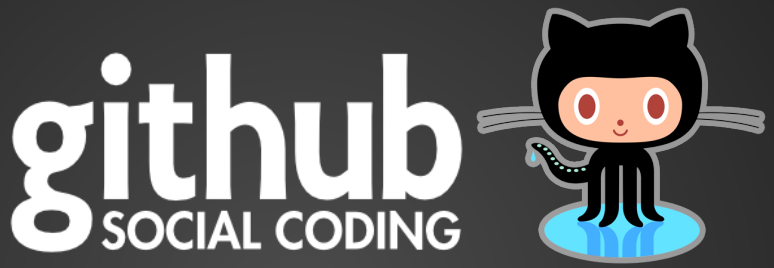
Why mine software repositories?

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


Google code


CodePlex
Open Source Project Community


Apache


github
SOCIAL CODING


Java.net


SOURCEFORGE.NET®

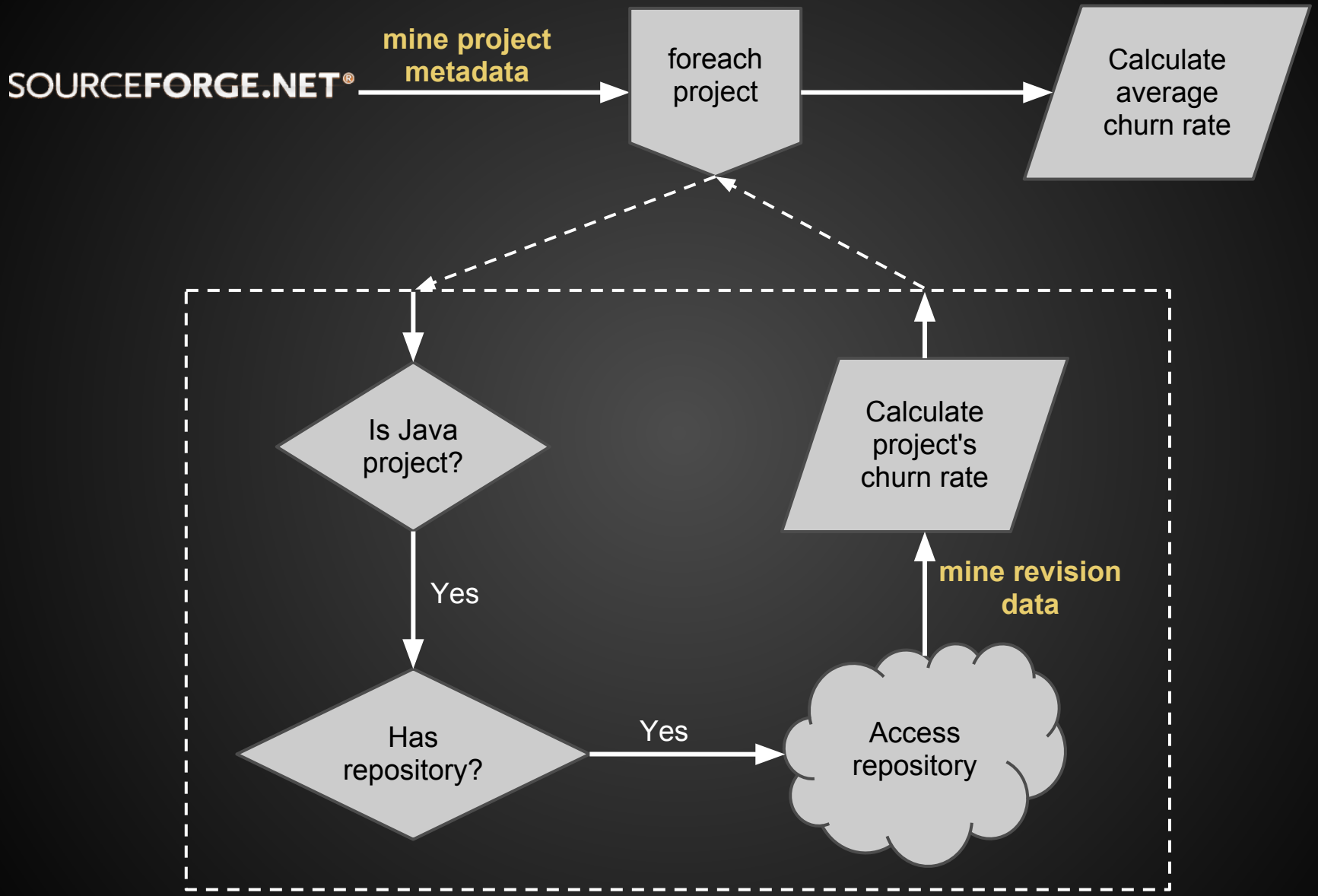

Atlassian
bitbucket


launchpad

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®
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...

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

exists (i: int; lowercase(p.programming_languages[i]) == "java")
     foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
          foreach (k: 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**!

A better solution...

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

exists (i: int; lowercase(p.programming_languages[i]) == "java")
    foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
        foreach (k: int; def(p.code_repositories[j].revisions[k]))
            rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

The Boa language and data-intensive infrastructure

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

Research Questions

1. Can we abstract and simplify the software mining process to make it more accessible to non-experts?

2. Can software repository mining be done efficiently at a large scale?

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.

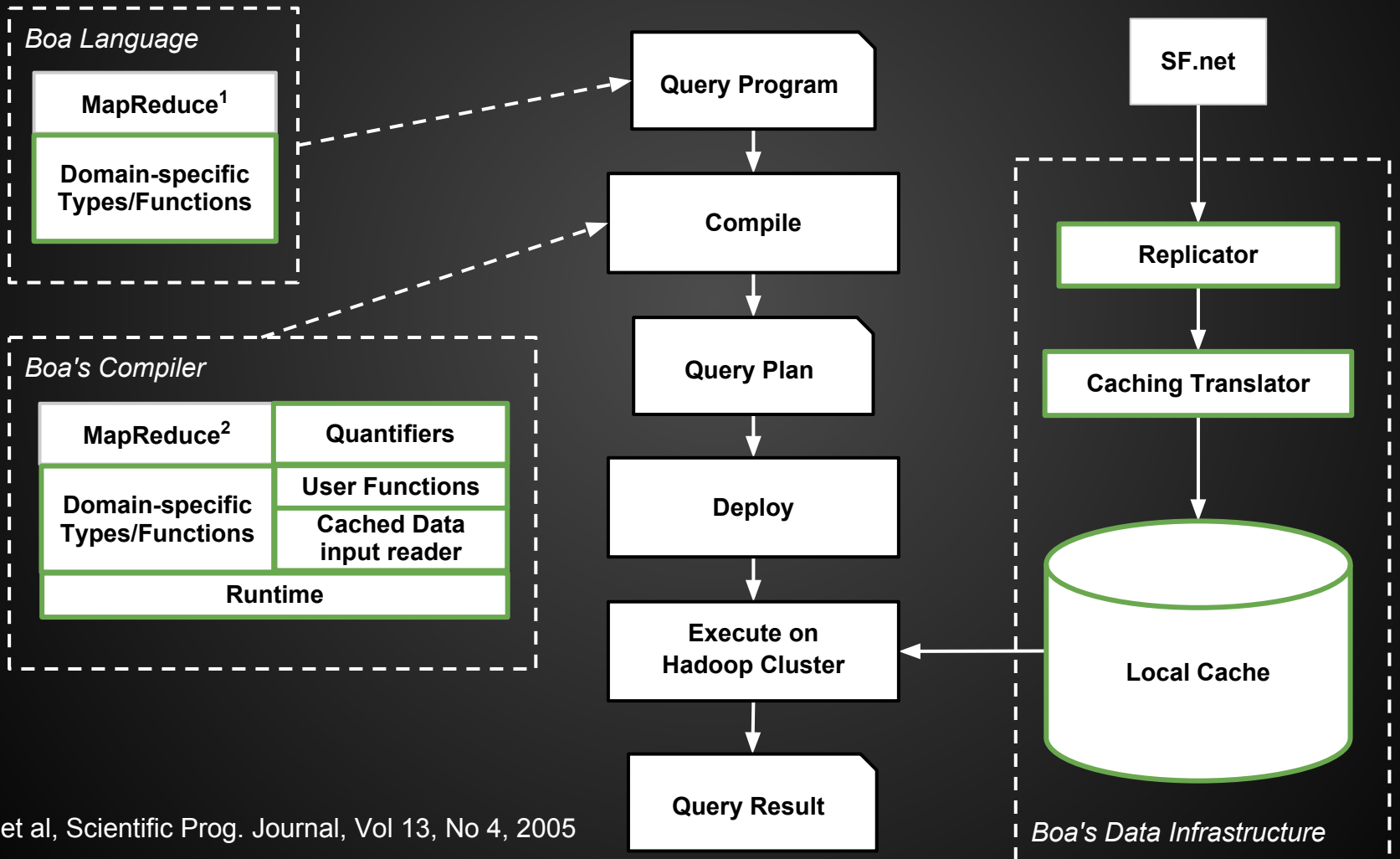
The rest of the paper is structured as follows: in the next section, the method used for this study is presented. Then some general remarks on the MSR conference are given,

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

Design goals

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

Domain-specific types

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

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

exists (i: int; lowercase(p.programming_languages[i]) == "java")
     foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
          foreach (k: 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
kind      : RepositoryKind
revisions : array of Revision
```

Revision

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

File

```
name   : string
kind   : FileKind
change : ChangeKind
```

Domain-specific functions

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

```
hasfiletype := function (rev: Revision, ext: string) : bool {
    exists (i: 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 (a1: t1, ..., an: tn) [: ret] {
    ... # body
    [return ...;]
};
```

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

Quantifiers

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

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

exists (i: int; lowercase(p.programming_languages[i]) == "java")
     foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
          foreach (k: int; def(p.code_repositories[j].revisions[k]))
               rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

- foreach, exists, ifall
- Bounds are inferred from the conditional

Output and aggregation

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

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

exists (i: int; lowercase(p.programming_languages[i]) == "java")
    foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
        foreach (k: int; def(p.code_repositories[j].revisions[k]))
            rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

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

Design goals

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

Let's see it in action!

<<demo>>

Why are we waiting for results?

Program is analyzing...

699,332 projects

494,159 repositories

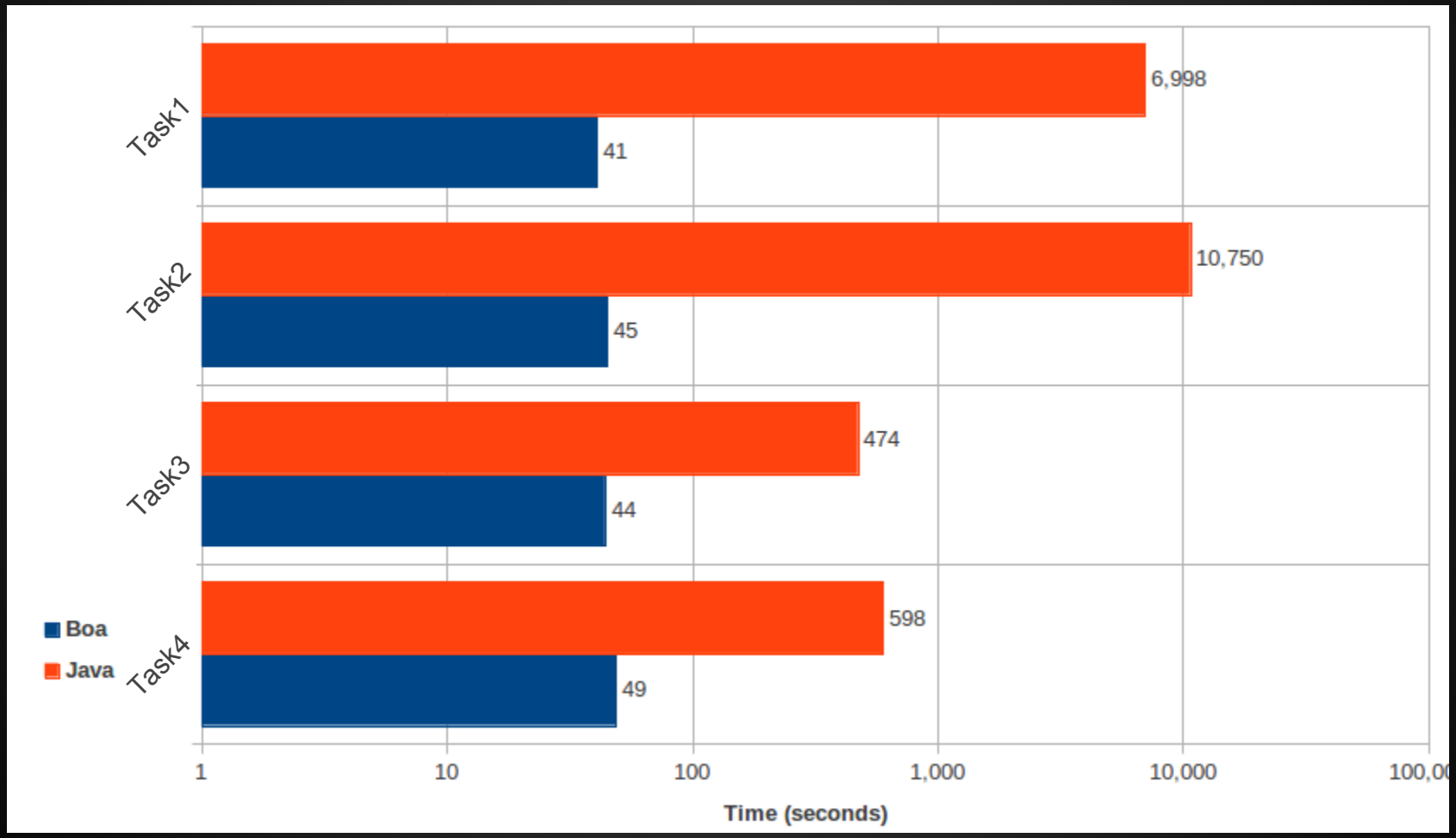
6,385,666 revisions

57,304,233 files

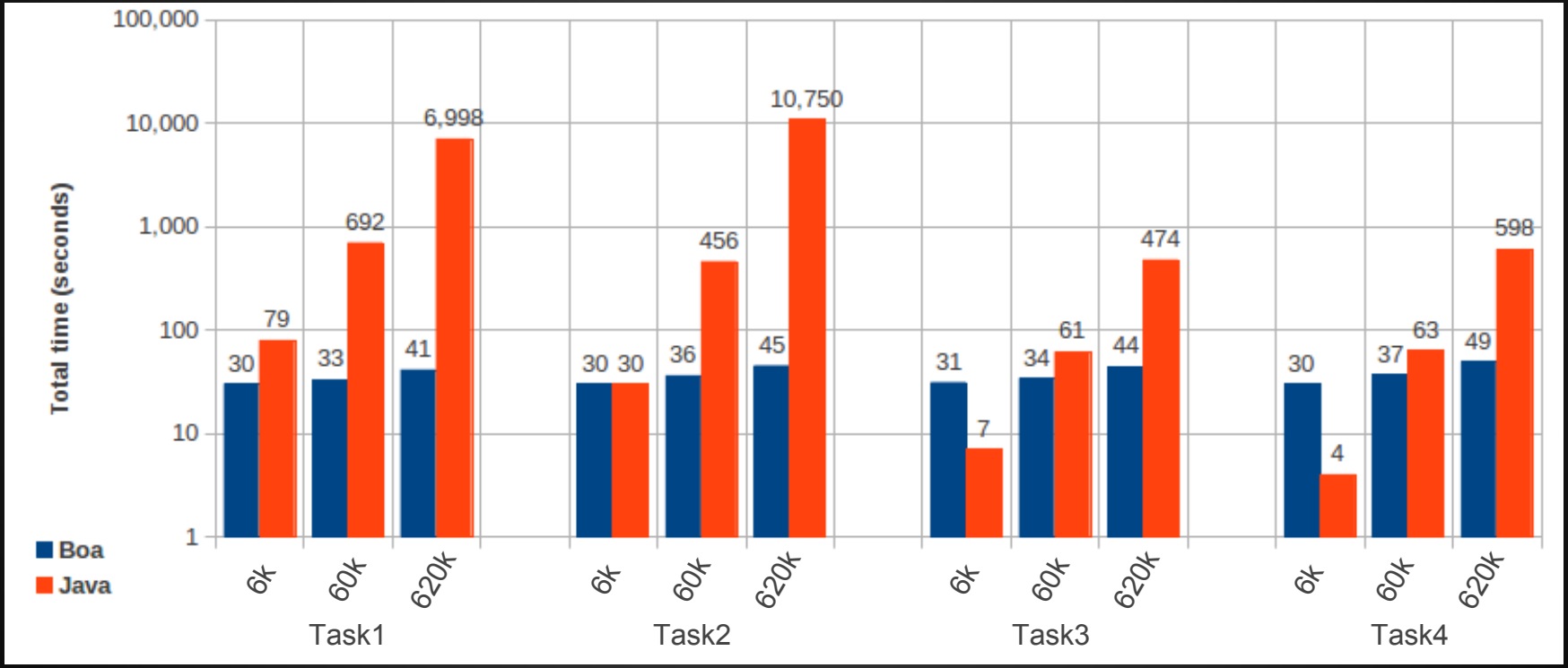
Let's check the results!

<<demo>>

Efficient execution



Scalability of input size



Design goals

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

Controlled Experiment

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

		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.

Related Works

Sourcerer [Linstead et al. Data Mining Know. Disc.'09]

- SQL database on 18k projects

Kenyon [Bevan et al. ESEC/FSE'05]

- Centralized database of metadata and source code

PROMISE [Boetticher, Menzies, Ostrand 2007]

- Online data repository for SE datasets
- Boa provides raw, un-processed data

Boa provides better scalability

Related Works

Sawzall [Pike et al. Sci.Prog.'05]

- Similar syntax to Boa
- Abstracts details of the MapReduce runtime

Pig Latin [Olston et al. SIGMOD'08]

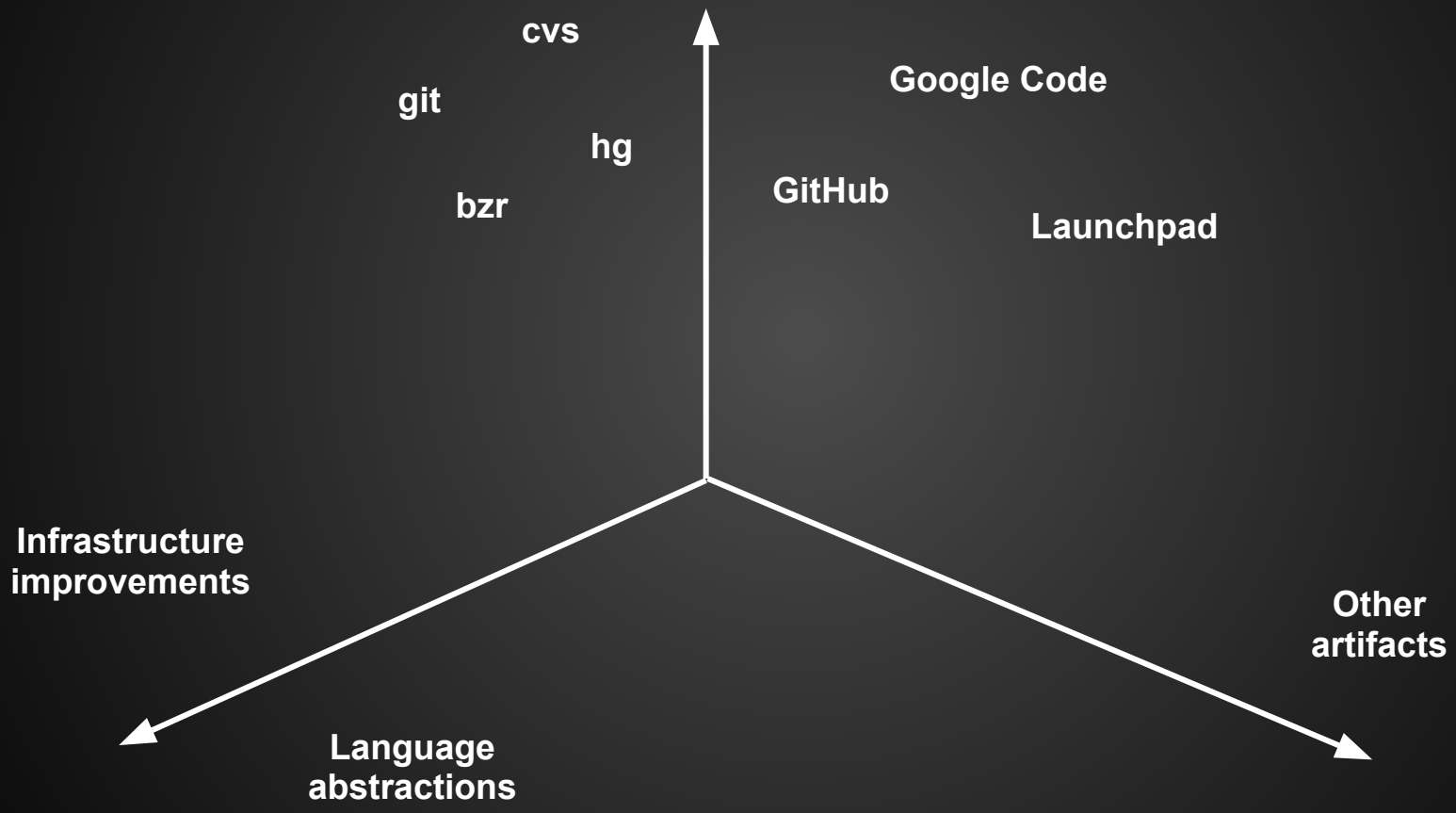
- Declarative syntax, similar to SQL

DryadLINQ [Yu et al. OSDI'08]

- Syntax based on .Net's LINQ
- Compiles to Dryad framework, a DAG of processes

None provide direct support
for mining software repositories

Ongoing work



Recent Work

- Support for mining source code
  - Down to expression level

- Currently for Java
  - Over 23k projects, with full history
  - Over 14 Billion AST nodes

Conclusions

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

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