A Large-Scale Study of Programminng Languages and Code Quality in GitHub

Authors
Baishakhi Ray, Daryl Posnett, Premkumar Devanbu, and Vladimir Fillkov

m5231153 Yuta Nemoto

Outlines

1 Introduction

Methodology 2

3 Results

Related Work 4

5 Threats to Validity

Conclusion 6

1. Introduction

A variety of ways to achieve the programming work.

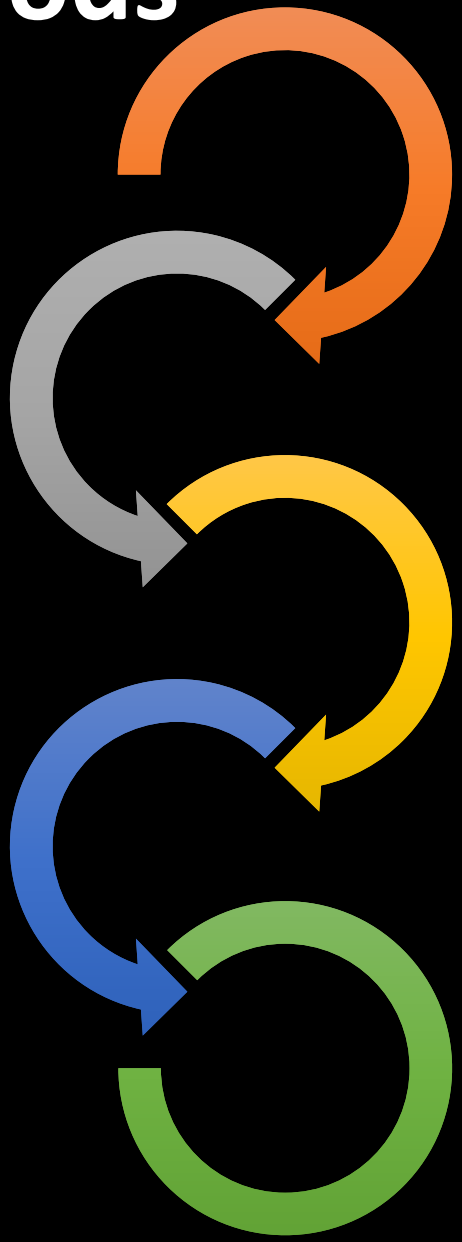
“What is the RIGHT tool for the job?”

Explore the effects of the features of
the Programming Language used in the practical projects.

Practical information can be obtained
only by exploring the REAL programs.

2. Methodology

Research Methods



Data Collection

Categorizing Languages

Identifying Project Domain

Categorizing Bugs

Statistical Analysis

Research Methods

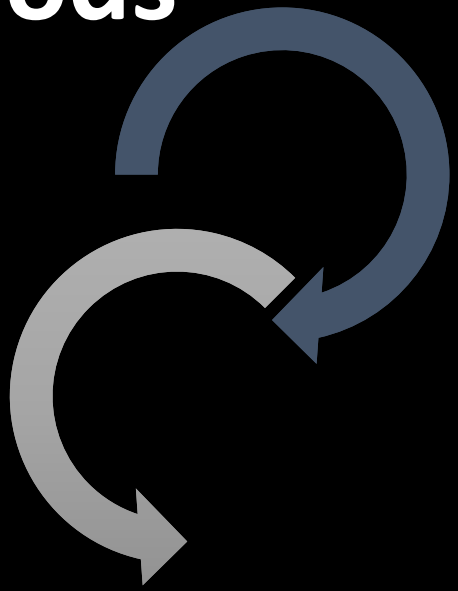
Select
Top **50 projects**
of
Top **19 Languages**

Data Collection

- ✓ Identify the top language of the project
- ✓ Retrieve the popular project
- ✓ Retrieve the project evolution history

Obtained **728** projects in **17** languages

Research Methods



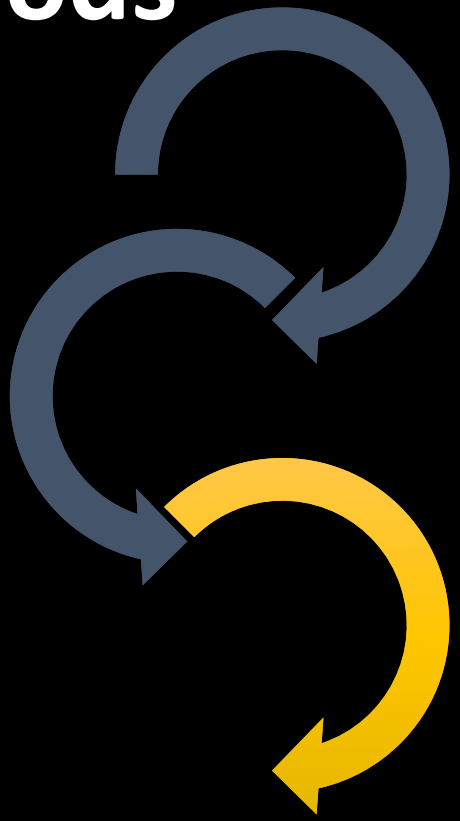
Categorizing Languages

- Programming Paradigm
  - Paradigm
    - Procedural
    - Scripting
    - Functional
- Type Checking — **Static** or **Dynamic**
- Implicit Type Conversion — **Allowing** or **Disallowing**
- Memory Class — **Managed** or **Unmanaged**

Research Methods

Identified Domains

- Application
- Database
- CodeAnalyzer
- Middleware
- Library
- Framework
- Others



Identifying Project Domain

LDA: Latent Dirichlet Allocation

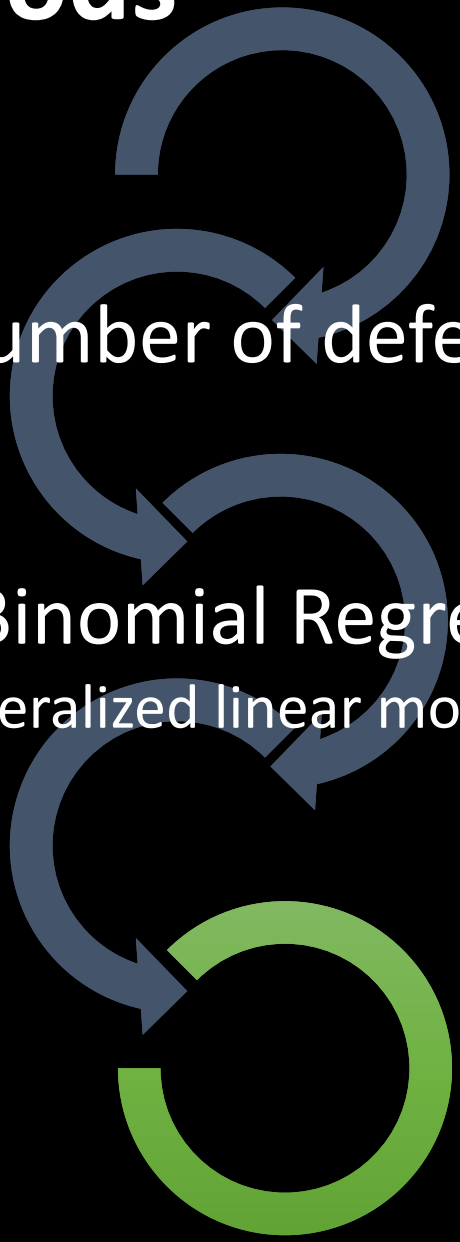
Estimates the probability of assigning that document to each topic.

Research Methods

1. Keyword Search
2. Supervised Classification

Categorizing Bugs

Research Methods



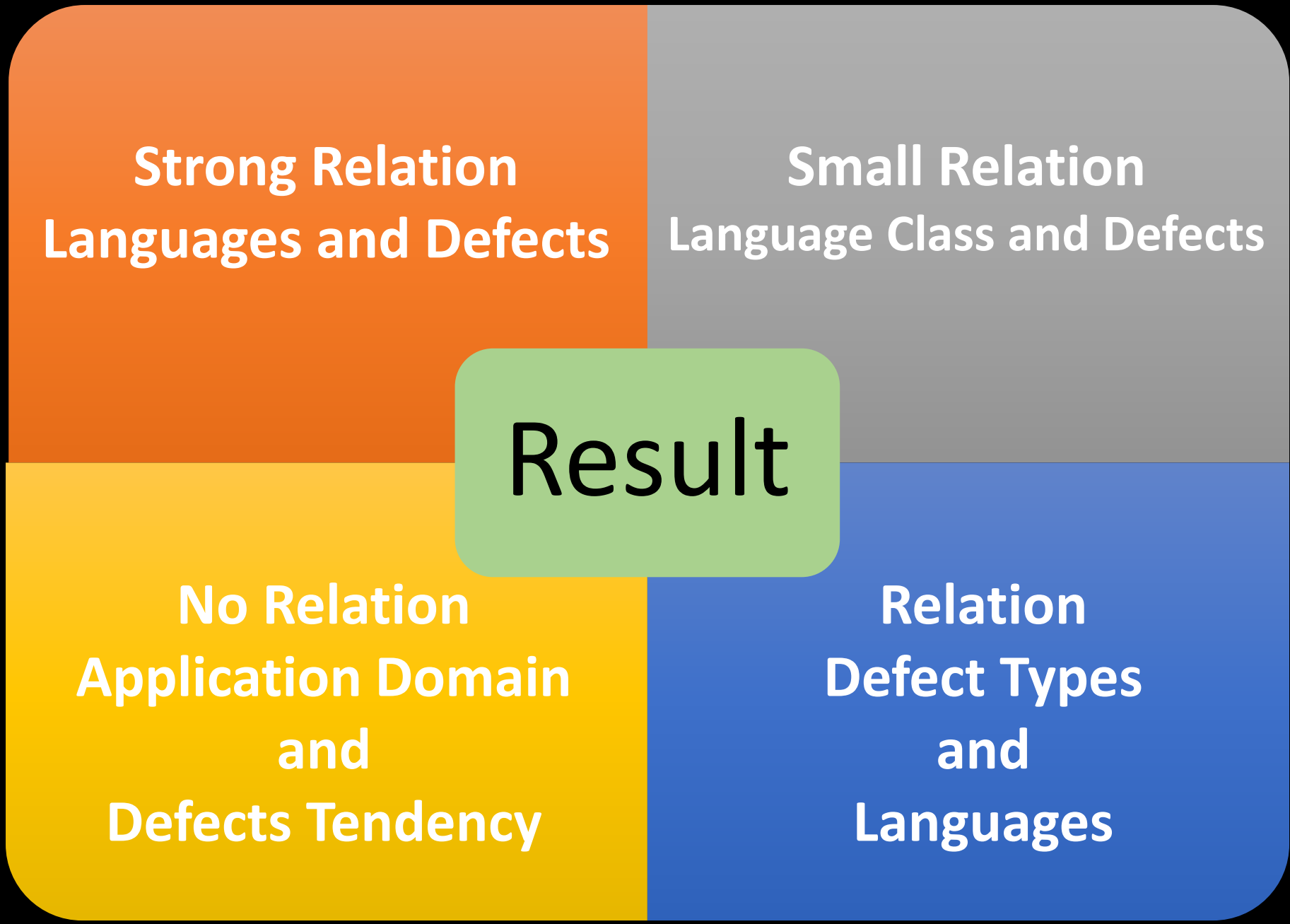
Modeling the number of defective commits

NBR: Negative Binomial Regression

A type of generalized linear model

Statistical Analysis

3. Results


Strong Relation
Languages and Defects
Small Relation
Language Class and Defects
Result
No Relation
Application Domain
and
Defects Tendency
Relation
Defect Types
and
Languages

Strong Relation Languages and Defects

Examples of Coefficient of Langs.	
DefectiveCommits Model	**Coefficient**
C	0.11
Python	0.08
JavaScript	0.03
C#	-0.02
Scala	-0.24
Haskell	-0.26

Bigger Coef. : much defect fixes
Smaller Coef. : less defect fixes

Some languages **have a greater association with defects** than other languages, although the effect is small.

Functional  Others 

Explicit  Implicit 

Static  Dynamic 

Small Relation
Language Class and Defects

Paradigm	Type	Im/Explicit	Memory	Coefficient
Functional	Static	Explicit	Managed	-0.25
Functional	Dynamic	Explicit	Managed	-0.17
Procedural	Static	Explicit	Managed	-0.06
Script	Dynamic	Explicit	Managed	0.001
Script	Dynamic	Implicit	Managed	0.04
Procedural	Static	Implicit	Unmanaged	0.14

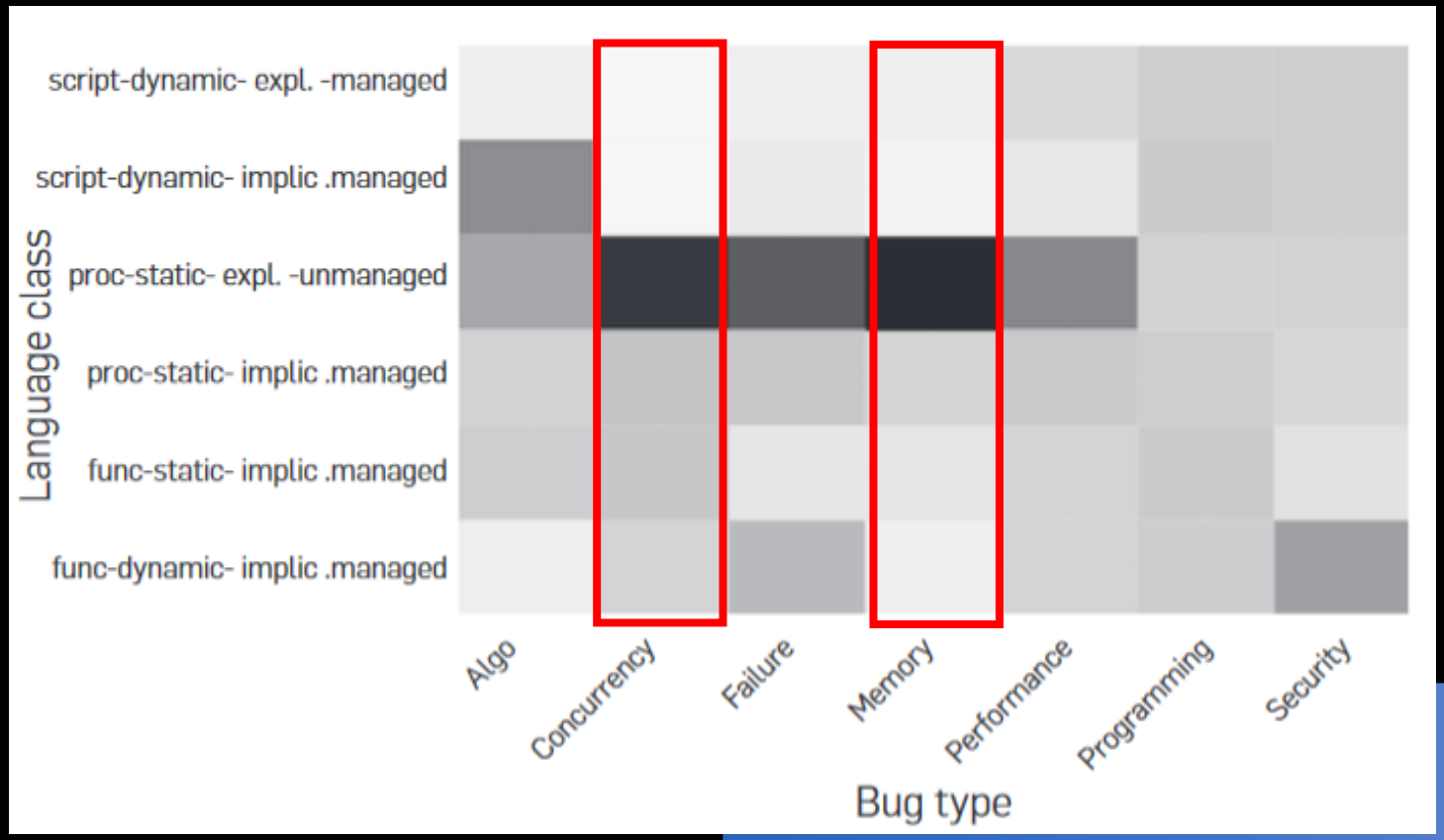
No meaningful relation between the number of bugs and domains except “Data Base” domain.

The relation between Language and Domain has stronger correlation than the error prone by the domain itself.

Significant if p-Value < 0.01

No Relation Application Domain and Defects Tendency

Domains	APP	CA	DB	FW	LIB	MW
Spearman Corr.	0.71	0.56	0.30	0.76	0.90	0.46
p-Value	0.00	0.02	0.28	0.00	0.00	0.09

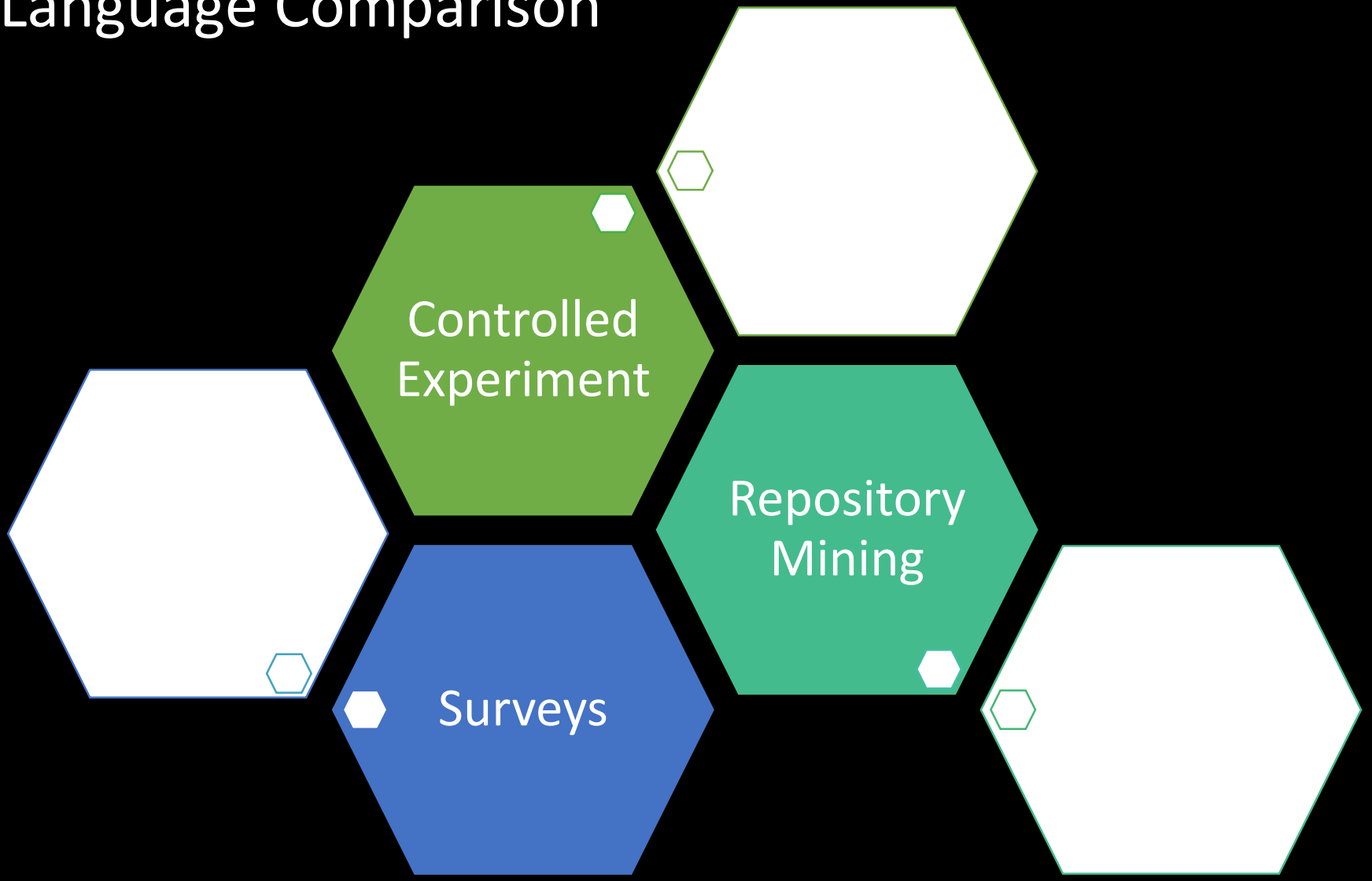


Relation Defect Types and Languages

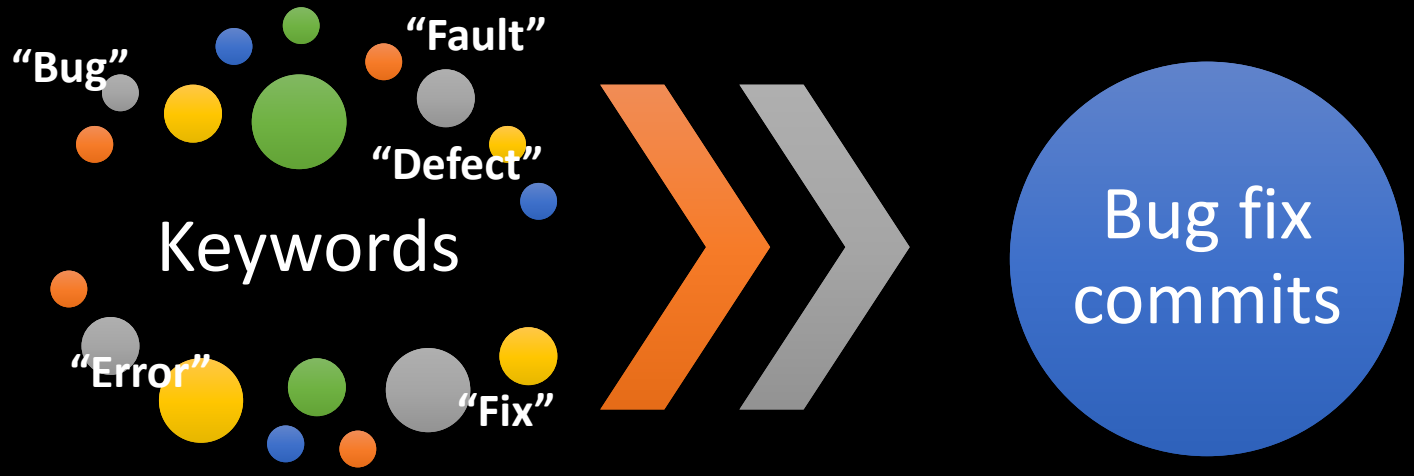
Strong association between Language primitives and bug types

4. Related Work

3 Categories of Programming Language Comparison



5. Threats to Validity



Keywords in the commit logs

Identifying the bug fix commits

Intentional?

Descriptiveness of commit logs **vary** across projects.

Evaluate these classification manually for random sample.

Average Accuracy: **84 %** for bug identification

Determining the program file **Language** by its **Extension**

C, Java, Python …. .c .java .py …

Might be wrong?

Common Language Extension (CLE)

middleware to support mixed programming of multiple languages.

Manually **verified** language **categorization**

against randomly sampled file set.

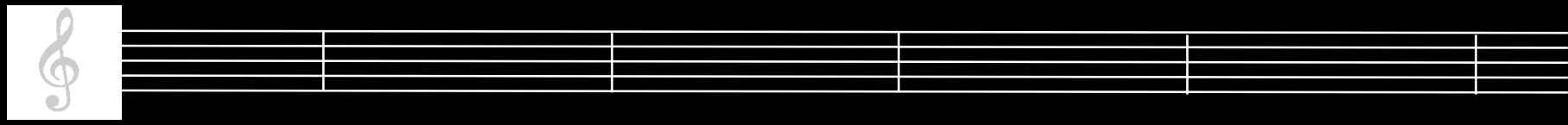
Associating **defect fixing commits** to the language **properties**

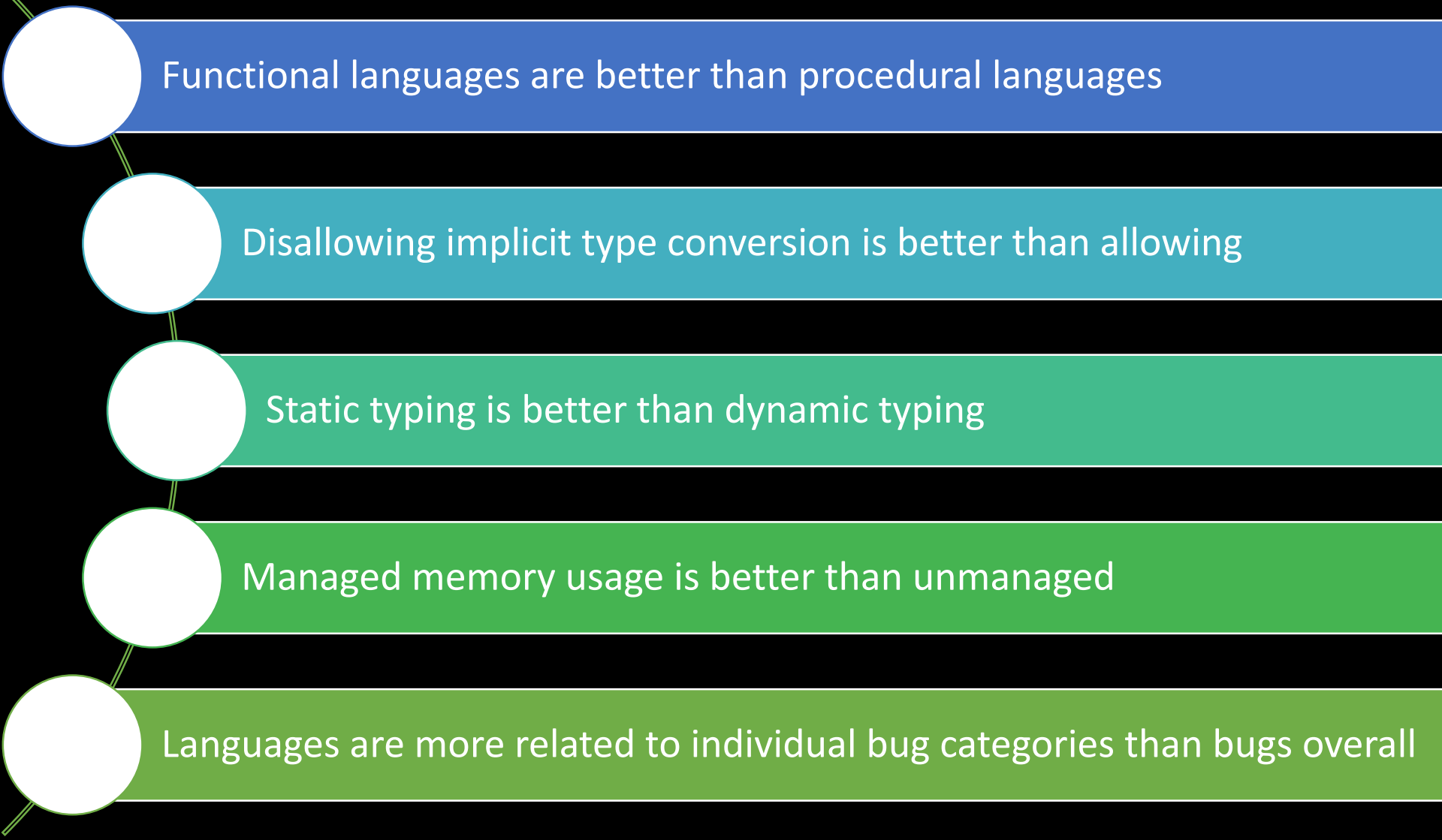
Actually,

- ✓ Reporting Style
- ✓ Other developer properties
- ✓ Availability of external tools or libraries

May also impact the extent of bugs associated with a language.

6. Conclusion




Functional languages are better than procedural languages
Disallowing implicit type conversion is better than allowing
Static typing is better than dynamic typing
Managed memory usage is better than unmanaged
Languages are more related to individual bug categories than bugs overall

Increasing number of **dependent variables** to evaluate

Difficult to answer questions about a specific variable's effect

Unable to quantify the specific effects of language type on usage

Thank you for your attention

m5231153 Yuta Nemoto