

A Comparative Study of Coding Styles and Method Naming Conventions in Open Source Software

発表者：ダニエル

背景

- **Open source software development** plays a vital role in fostering global collaboration and innovation in technology.
- **Code readability and consistent style guidelines**, including method naming conventions, are key to successful development and collaboration.
- Coding Styles and Naming impacts the readability and comprehension code.
- Understanding these differences can lead to **improved code readability, enhanced international collaboration**, and the **standardization of coding practices**.

先行研究

■ Open Source Development Around the World: A Comparative Study

- number of projects per country
- popularity and programming language of each country's project
- number of projects in a country correlates to its GDP
- International vs Domestic Projects

Thaís Mombach¹, Marco Tulio Valente¹, Cuiting Chen²,
Magiel Bruntink², Gustavo Pinto³

¹Federal University of Minas Gerais, Brazil; ²Software Improvement Group, Netherlands; ³Federal University of Pará, Brazil

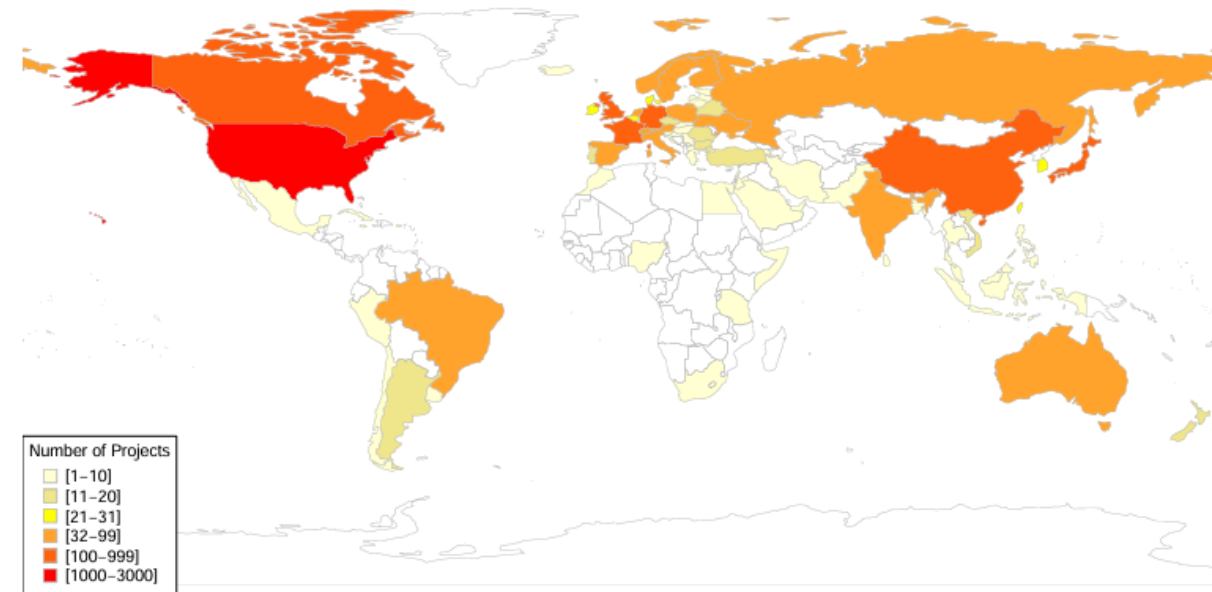


Figure 1: Number of GitHub projects per country

Approach

- Finland (47%) , Brazil (40%) has a higher percentage of international projects
- The Asian countries have the highest percentage of domestic projects
- Japan is one of the highest percentage of domestic projects (73%)
- Investigate relationship and the differences in coding style and naming convention
- How coding style and naming convention affect international and domestic project

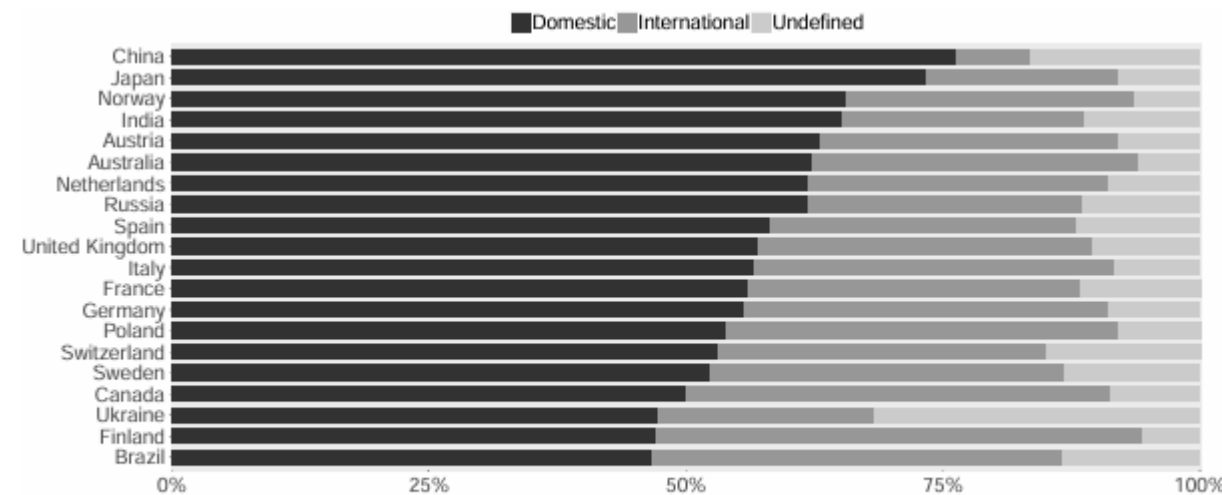


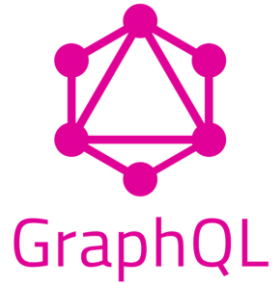
Figure 6: International and domestic projects per country

Aim

- **Identify** differences in coding styles and method naming conventions
- **Examine the impact of these differences** on international collaboration and project participation rates.
- Provide insights to **enhance code readability**

Research method

1. **Extract** repositories from users in different country



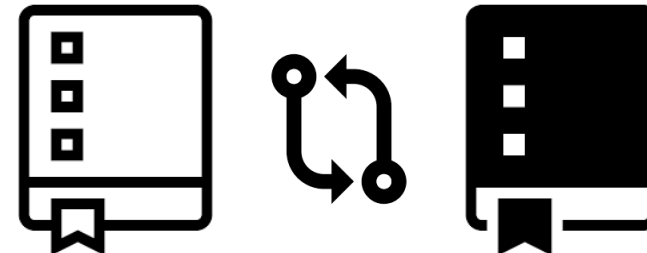
2. **Filtering** extracted repositories



3. Filtered repository **analysis**



4. **Comparing** the analysis results



Query

検索クエリ:

- 場所: {日本/複数の国(Brazil/Finland)}
- followers数: <10, <100, <1000 (Cluster Collection)
- 異なるプログラミング言語で検索

抽出されたリポジトリ:

- ユーザーごとに 3 つのリポジトリ
- 合計 300 ユーザー

1時間あたり5,000リクエスト

```
Rate Limit: 5000 requests remaining
Rate limit resets at: 2024-07-24 03:22:25 JST
Current time: 2024-07-24 02:22:26 JST
Time left until rate limit reset: 3598.1920986175537 seconds
```

Extracted Data

Extracted **300 users**, each with **3 repositories**

Collected a total of **900 repositories per language**

- Languages collected include:

- C
- C++
- Java
- JavaScript
- PHP
- Python
- Ruby

- Filters:

- Allow extraction of only certain **programming extensions**
- File limit for each repository is **30**

Total of **6300** repositories (target)

Filtering

Remove comment → Filter by language → Graph the result

CLOC

- cloc counts blank lines, comment lines, and physical lines of source code in many programming languages

```
github.com/AlDanial/cloc v 1.90 T=0.56 s (434.7 files/s, 63184.2 lines/s)
-----
Language          files      blank      comment      code
-----
Java                242        4455        7245        23473
-----
SUM:                242        4455        7245        23473
-----
```

cloc

Count Lines of Code

Naming Conventions

Importance of naming convention:

- **maintain** organization
- makes data **easier to process** and analyze
- Lack of structured naming can lead to **chaos** when dealing with large volumes of data.

Naming Conventions

Common elements influence most conventions:

1. Length of Identifiers:

- Rules determine how many characters are allowed in an identifier.
- Short identifiers
- Long identifiers

2. Letter Case and Numerals:

- Some conventions specify rules for letter case (uppercase/lowercase) or restrict the use of numerals.

3. Multiple-Word Identifiers:

- Multiple words can make identifiers more meaningful.

Naming Conventions

Multiple-Word Identifiers:

Multiple-word identifier formats

Formatting	Name(s)
<code>twowords</code>	flatcase ^{[13][14]}
<code>TWOWORDS</code>	UPPERCASE, SCREAMINGCASE ^[13]
<code>twoWords</code>	(lower) camelCase, dromedaryCase
<code>TwoWords</code>	PascalCase, UpperCamelCase
<code>two_words</code>	snake_case, snail_case, pothole_case
<code>TWO_WORDS</code>	ALL_CAPS, SCREAMING_SNAKE_CASE, ^[15] MACRO_CASE, CONSTANT_CASE
<code>two_Words</code>	camel_Snake_Case
<code>Two_Words</code>	Pascal_Snake_Case, Title_Case
<code>two-words</code>	kebab-case, dash-case, lisp-case, spinal-case
<code>TWO-WORDS</code>	TRAIN-CASE, COBOL-CASE, SCREAMING-KEBAB-CASE
<code>Two-Words</code>	Train-Case, ^[13] HTTP-Header-Case ^[16]

Naming Conventions

Standard Naming Convention by programming language

Language	Naming Conventions
C	Lowercase for standard library (e.g., <code>isalnum</code>); macros in UPPERCASE_WITH_UNDERSCORES .
C++	Similar to C; uses underscores for word separation (e.g., <code>out_of_range</code>); macros in UPPERCASE_WITH_UNDERSCORES .
Java	Classes: UpperCamelCase ; Methods/Variables: lowerCamelCase ; Constants: SCREAMING_SNAKE_CASE .
JavaScript	Similar to Java. Classes: UpperCamelCase ; Methods/Variables: lowerCamelCase .
PHP	Classes: PascalCase ; Constants: MACRO_CASE ; Methods/Functions: camelCase .
Python	Classes: UpperCamelCase ; Constants: CAPITALIZED_WITH_UNDERSCORES ; Variables/Methods: snake_case .
Ruby	Similar to Python. Classes: UpperCamelCase ; Constants: CAPITALIZED_WITH_UNDERSCORES ; Variables/Methods: snake_case .

Approach

1. Define Standard Naming Conventions for Comparison

2. Extract data(in progress)

3. Programmatic Analysis of Naming Patterns

- In Python, look for `def <method_name>`.
- In Java, look for `<return_type> <method_name>(...)`.
- In C++, look for `<return_type> <method_name>(...)`.

Approach

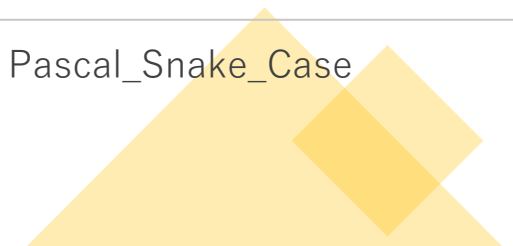
4. Categorize the Method Names

- Use a combination of **string manipulations** and **regular expressions** to determine the category:
- CamelCase: Starts with a lowercase letter, and words are capitalized (e.g., $^[a-z]+([A-Z][a-z]^*)^*\$$).
- PascalCase: Starts with an uppercase letter (e.g., $^[A-Z][a-z]+([A-Z][a-z]^*)^*\$$).
- snake_case: Words are separated by underscores (e.g., $^[a-z]+(_[a-z]+)^*\$$).
- Verb-based Names: Check if the method name starts with a common verb (e.g., $^(get|set|calculate|fetch|create|update|delete).^*\$$).

5. Measure Compliance

Metrics to calculate:

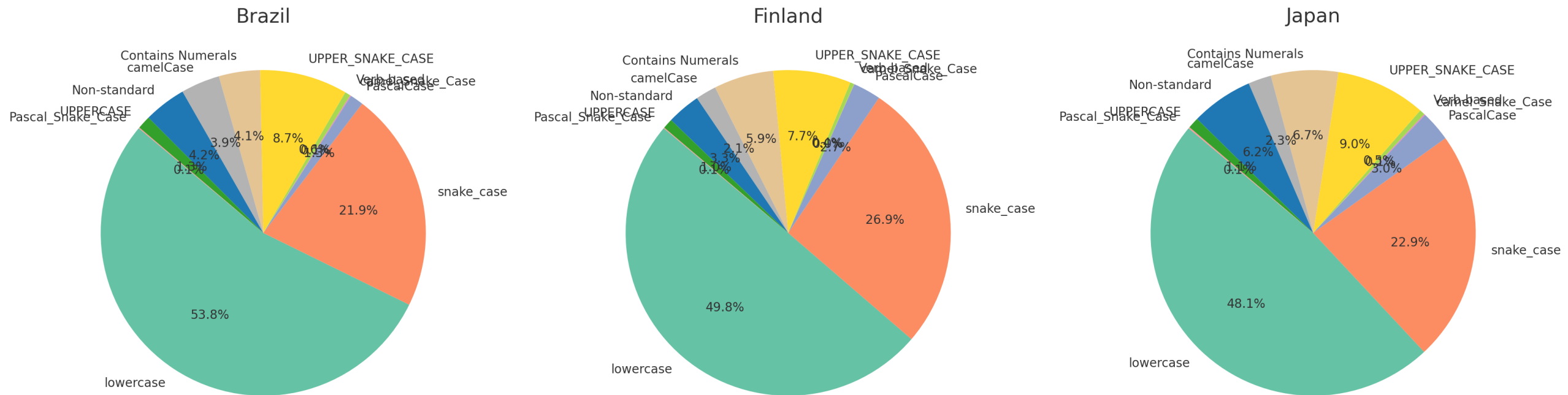
- **Adherence Rate:** The proportion of method names that match a standard naming convention.
- **Deviation Rate:** The proportion of method names that do not follow any recognized naming convention.



Naming Convention	Brazil (%)	Brazil Total	Finland (%)	Finland Total	Japan (%)	Japan Total
lowercase	53.85%	259874	49.80%	345834	48.07%	707753
snake_case	21.86%	105486	26.89%	186730	22.92%	337532
PascalCase	1.33%	6438	2.75%	19128	3.03%	44664
camel_Snake_Case	0.10%	491	0.02%	149	0.11%	1630
Verb-based	0.56%	2683	0.41%	2836	0.53%	7768
UPPER_SNAKE_CASE	8.73%	42134	7.73%	53664	8.96%	131888
Contains Numerals	4.10%	19778	5.94%	41243	6.69%	98491
camelCase	3.87%	18664	2.06%	14279	2.29%	33790
Non-standard	4.20%	20278	3.32%	23087	6.17%	90844
UPPERCASE	1.32%	6353	1.00%	6976	1.10%	16222
Pascal_Snake_Case	0.09%	418	0.08%	567	0.13%	1866

Naming Convention(C language)

Naming Convention Distribution by Country



Conclusion

- C言語の場合、すべての国で小文字表記が優勢であることを示した
- non-standard conventionsの割合は、すべての国で比較的低く、日本は 6.17%、ブラジルは 4.20%、フィンランドは 3.32% です。

non-standard conventionsは、特定のプロジェクト内での命名方法のばらつきを示唆したり、一部のコードに命名ガイドラインが欠けていることを示している可能性があります。