Pre-proposal Oral Presentation

Scalable Clone Detection

He Feng

Department of Physics

fenghe@vt.edu

Liuqing Li

Department of Computer Science

liuqing@vt.edu

VirginiaTech

Invent the Future

Outline

- Importance of Clone Detection
- Background of Clone Detection
- Our proposed solution

Importance of Clone Detection

- Code Clone
  - In coding process
    - Common
    - You & Me

Importance of Clone Detection

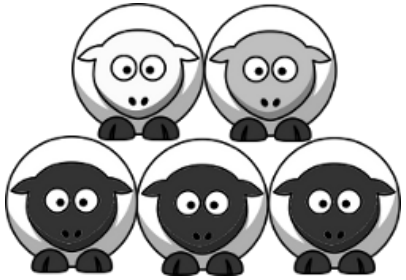
- Code Clone
  - In coding process
    - Common
    - You & Me
  - In software systems
    - X Windows System : about 19%
    - Core parts of Linux : between 15% and 25%

Importance of Clone Detection

- Code Clone
  - In coding process
    - Common
    - You & Me
  - In software systems
    - X Windows System : about 19%
    - Core parts of Linux : between 15% and 25%
  - Pros
    - can improve the efficiency of a program
  - Cons
    - Can be difficult to add or cut features
    - Can be difficult to fix bugs

Importance of Clone Detection

- Code Clone
  - In coding process
    - Common
    - You & Me
  - In software systems
    - X Windows System : about 19%
    - Core parts of Linux : between 15% and 25%
  - Pros
    - can improve the efficiency of a program
  - Cons
    - Can be difficult to add or cut features
    - Can be difficult to fix bugs



Scalable Clone Detection

Background of Clone Detection

- Q: Which code fragments should be considered as copied code?
- A: Code clone can be categorized into 4 types

Background of Clone Detection

- Q: Which code fragments should be considered as copied code?
- A: Code clone can be categorized into 4 types
  - Type 1
    - Includes the variations in whitespace and comments

Background of Clone Detection

- Q: Which code fragments should be considered as copied code?
- A: Code clone can be categorized into 4 types
  - Type 1
    - Includes the variations in whitespace and comments
  - Type 2
    - Allows more variations in identifiers, literals and types

Background of Clone Detection

- Q: Which code fragments should be considered as copied code?
- A: Code clone can be categorized into 4 types
  - Type 1
    - Includes the variations in whitespace and comments
  - Type 2
    - Allows more variations in identifiers, literals and types
  - Type 3
    - Allows further modifications such as changed, added or removed statements

Background of Clone Detection

- Q: Which code fragments should be considered as copied code?
- A: Code clone can be categorized into 4 types
  - Type 1
    - Includes the variations in whitespace and comments
  - Type 2
    - Allows more variations in identifiers, literals and types
  - Type 3
    - Allows further modifications such as changed, added or removed statements
  - Type 4
    - Perform the same computation but are implemented in different ways

Background of Clone Detection

- Q: How to detect the code clone?
- A: Techniques and Tools

Background of Clone Detection

- Q: How to detect the code clone?
- A: Techniques and Tools
  - Text-based
  - Token-based
  - Tree-based
  - Graph-based

Background of Clone Detection

- Q: How to detect the code clone?
- A: Techniques and Tools
  - Text-based
  - Token-based
  - Tree-based
  - Graph-based
- Token-based method performs well in
  - Clone type detection (Type 1, 2 ,3)
  - Time cost

Our proposed solution

- Goal
  - Develop and implement a scalable clone detection based on tokens
  - Detect Type 1, 2, 3 code clone
- Difficulty
  - Many researchers have done similar job before
- Breakthrough
  - New and aggressive method
- Premise
  - Developers would not make dramatic changes in code clone process except for some modifications such as types, identifiers and statements

Our proposed solution

- Method
  - Design and improve a classification statistical method to calculate the similarity between code fragments
- Fragment A and B
  - Use a parser to catch all the tokens
  - Categorize key tokens into types, variables, identifiers and even operators if needed
  - Accumulate the occurrences of each key token
  - Transform the fragment into a list of key token and frequency
  - Calculate the similarity between two lists
  - Set a threshold to the final output

Our proposed solution

- To be discussed
  - Key token selection
  - Variable comparison (e.g. changes to the token order)
  - Weights in taxonomy (e.g. type > identifier > operator ?)
  - Similarity algorithm
- To be expected
  - Time Cost
    - Classification statistical method
    - No complex lookup or comparison
  - Precision & Recall
    - ? ? ?

Thank you !