Vaci:

Variation Analysis of Context-Sharing Identifiers with Code Clones

Toshihiro Kamiya
Service Innovation Management Research Team
National Institute of Advanced Industrial Science and Technology, Japan
t-kamiya@aist.go.jp
Naming, an old and new problem

Everyday developers give names to types, functions, variables, enums, …

Adequate (descriptive, consistent) name is important.
- contributes to maintainability and understandability
- → naming convention

However, it’s not always possible.
- Changelogs contain “rename”, “replace”…

Some tools have been proposed to support naming / renaming.
Tool support

- Support for renaming
  - Name **restructuring tool** (overhauling names in a product)
  - Rename **refactoring** menu (such as Eclipse IDE)
  - “@deprecated” JavaDoc, **annotation** (Java)
  - Source **code converter** (Python 2.5 → 3.0)
- Support for checking name
  - Consistency checker (name ⇔ type)
  - Spell checker (such as Eclipse IDE)
Tool Vaci

The tool collects and analyzes relations between each name and its contexts (i.e., code fragments where the name appears).

– Finding Variation of Names: to check consistency among the names that are used in the similar contexts.
– Tracking Change of Names: to track how names changes between versions.
Snapshot #1, finding variation of names

The viewer works as Eclipse plugin.

It reads: the name “metricFile” and “cloneSetMetricFile” are used in the similar context.

I should rename the former name when I coded the latter…
Snapshot #2, tracking change of names

The analysis result is rendered into html document, including graphics (Scalable Vector Graphics).

It reads: the identifiers named “HWMODE_A” is renamed to “IEEE80211_BAND_5GHZ”.

Also, some identifiers named “MODE_IEEE80211A” is renamed to “IEEE80211_BAND_5HZ.” The others is renamed to “AR5K_MODE_11A”.

Translation map # 17

Graph

Edges (name changed)

HWMODE_A → IEEE80211_BAND_5GHZ

MODE_IEEE80211A → AR5K_MODE_11A
Basic idea: “Identifiers share a context”

The cases where two identifiers of distinct names appear in the identical code fragments.

```java
String getFirstPartOfPath(String fpath) {
    int pos = fpath.indexOf(File.pathSeparator);
    if (pos >= 0)
        fpath = fpath.substring(0, pos);
    return fpath;
}
```

```java
String split1stPart(String fpath) {
    int pos = fpath.indexOf(File.pathSeparatorChar);
    if (pos >= 0)
        fpath = fpath.substring(0, pos);
    return fpath;
}
```
Finding Variation of Names

1. Detect “context-sharing” identifier pairs.
2. Generate their transitive closures (translation classes). 
3. Classify each translation class by the names of identifiers in the class.
Example from Linux Kernel

```c
....
    return -EINVAL;
}
return 0;
...

result = i915_emit_irq(dev);
if (DRM_COPY_TO_USER(emit->irq_seq, &result, sizeof(int))) {
    DRM_ERROR("copy_to_user
    return -EFAULT;
}
return 0;
....

if (db->sample_size == 16 && !mono && db->src_factor == 1) {
    /* no translation necessary, just copy *
    if (copy_to_user(userbuf, dmabuf, dmacount))
        return -EFAULT;
    return dmacount;
}
....

for (sample = 0; sample < num_samples; sample++) {
    if (copy_from_user(usersample, userbuf, db->user_bytes_per_sample)) {
        return -EFAULT;
    }
    for (i = 0; i < db->num_channels; i++) {
        ....
```
Tracking Change of Names

1. Detect context-sharing identifiers **between two versions**.
2. Generate their transitive closures (**translation maps**).
3. Classify each translation class **by number of identifiers**.
Implementation

Tool Vaci consists of:

- vacicmd
- vaci plug-in
- visualizetransmap

Total 5k lines of Python/C++ lines

- Except for reused code from a clone detection tool “CCFinderX”
www.ccfinder.net/vaci.html
The latest version of Vaci (used in this presentation) will be available soon.

Other useful materials

• Observations, theories

• Keyword → Code