LUND UNIVERSITY

Copy & Paste Redeemed

Christoph Reichenbach

Copy & Paste

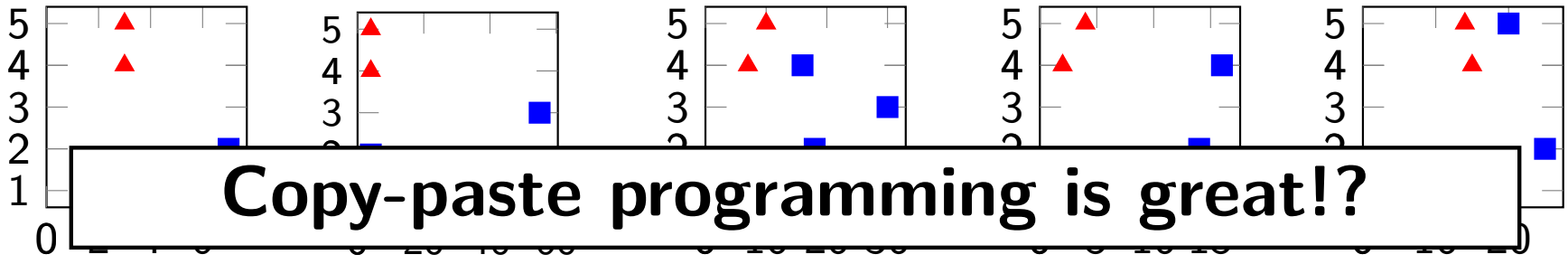
```
int dist(coord a, coord b) {
  int dx = abs(a.x - b.x);
  int dy = abs(a.y - b.y);
  return std::max(dx, dy);
}
```

```
int dist(coord a, coord b) {
  int dx = abs(a.x - b.x);
  int dy = abs(a.y - b.y);
  return std::max(dx, dy); std:
}
```

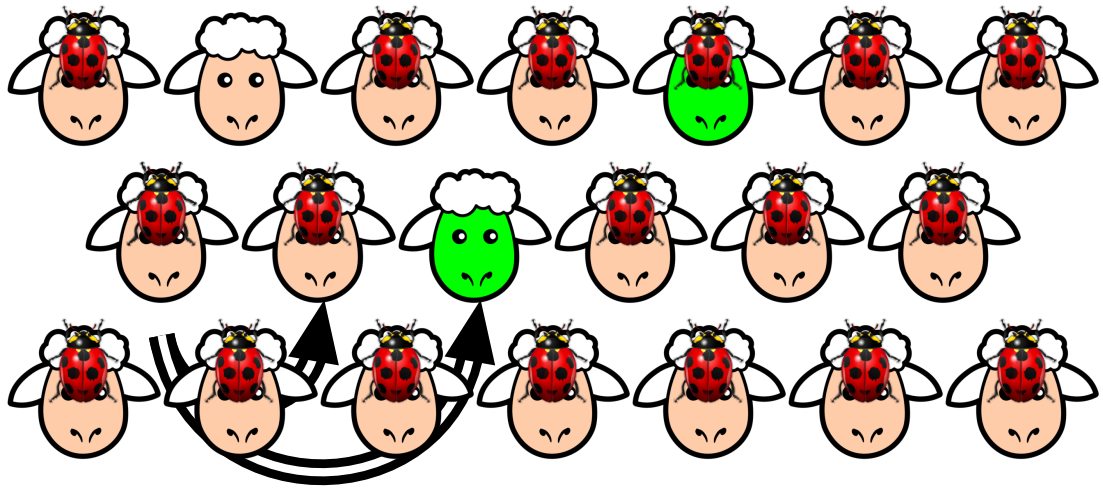
```
int distG(coord a, coord b,
          int (*f)(int, int) ) {
  int dx = abs(a.x - b.x);
  int dy = abs(a.y - b.y);
  return f (dx, dy);
}
```

The Benefits of Copy & Paste

- Five C++ programmers asked to reimplement missing clone
- **Abstraction**: ■
  - Users tend to prefer abstraction
  - *7*/1012/15 tasks completed in time
  - Factor 2-3 slower than abstraction
- **Copy-Paste**: ▲
  - 15/15 tasks completed in time
  - Universally faster (except for a single tie)



The Cost of Clones



"In this paper we provide strong evidence that inconsistent clones constitute a major source of faults, which means that cloning can be a substantial problem during development and maintenance unless special care is taken to find and track existing clones and their evolution."

Do Code Clones Matter?

Elmar Juergens, Florian Deissenboeck, Benjamin Hummel, Stefan Wagner
Institut für Informatik, Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
{juergens,deissenb,hummelb,wagnerst}@in.tum.de

"Several studies show that software systems with code clones are more difficult to maintain than the ones without them [118, 18]. The tendency of cloning not only produces code that is difficult to maintain, but may also introduce subtle errors [51, 168, 169]."

A Survey on Software Clone Detection Research*

Chanchal Kumar Roy and James R. Cordy

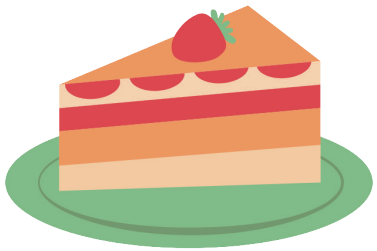
The Reuse Discrepancy

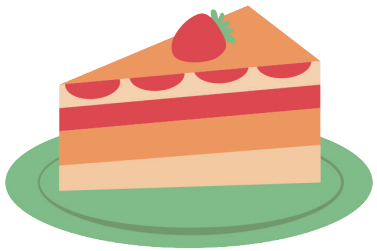
Copy-Paste-Modify

- \+ Fast
- – Creates technical debt 
- – Clones bugs 
- – Bloats code base

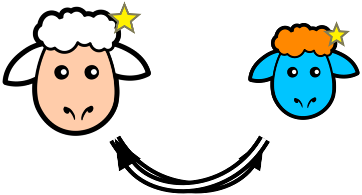
Create Abstractions

- \+ Avoids cloning
- \+ Opportunity to develop insights
- – Slow
- – Complex $\Rightarrow$ error-prone





A New Workflow



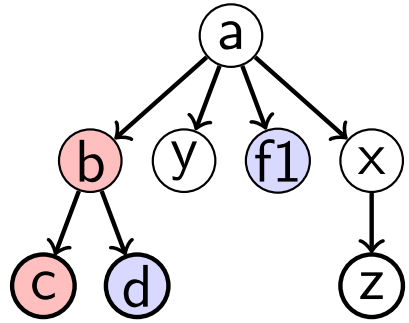
```
int distG(coord a, coord b,
          int (*f)(int, int)) {
  int dx = abs(a.x - b.x);
  int dy = abs(a.y - b.y);
  return f(dx, dy);
}
```

```
int dist(coord a, coord b) {
  int dx = abs(a.x - b.x);
  int dy = abs(a.y - b.y);
  return std::max(dx, dy);
}
```

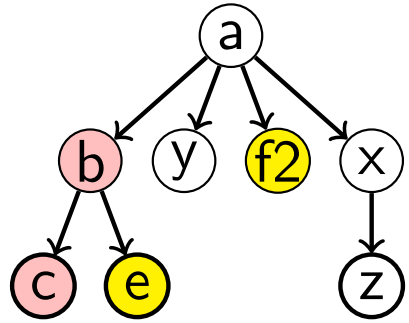
```
int dist(coord a, coord b) {
  int dx = abs(a.x - b.x);
  int dy = abs(a.y - b.y);
  return std::max(dx, dy);  std::hyp
}
```

An Example

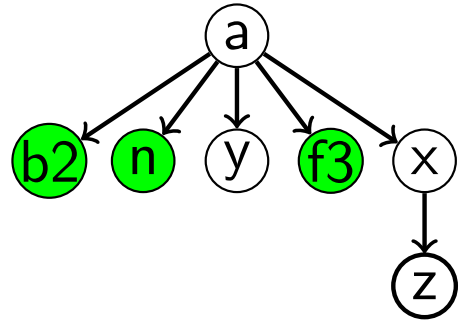
```
void function1()
{
   fake_db_init(&fakedb, true);
   run_mode = MODE_DEBUG;
   start(&service);
}
```




```
void function2()
{
   fake_db_init(&fakedb, false);
   run_mode = MODE_INFO;
   start(&service);
}
```



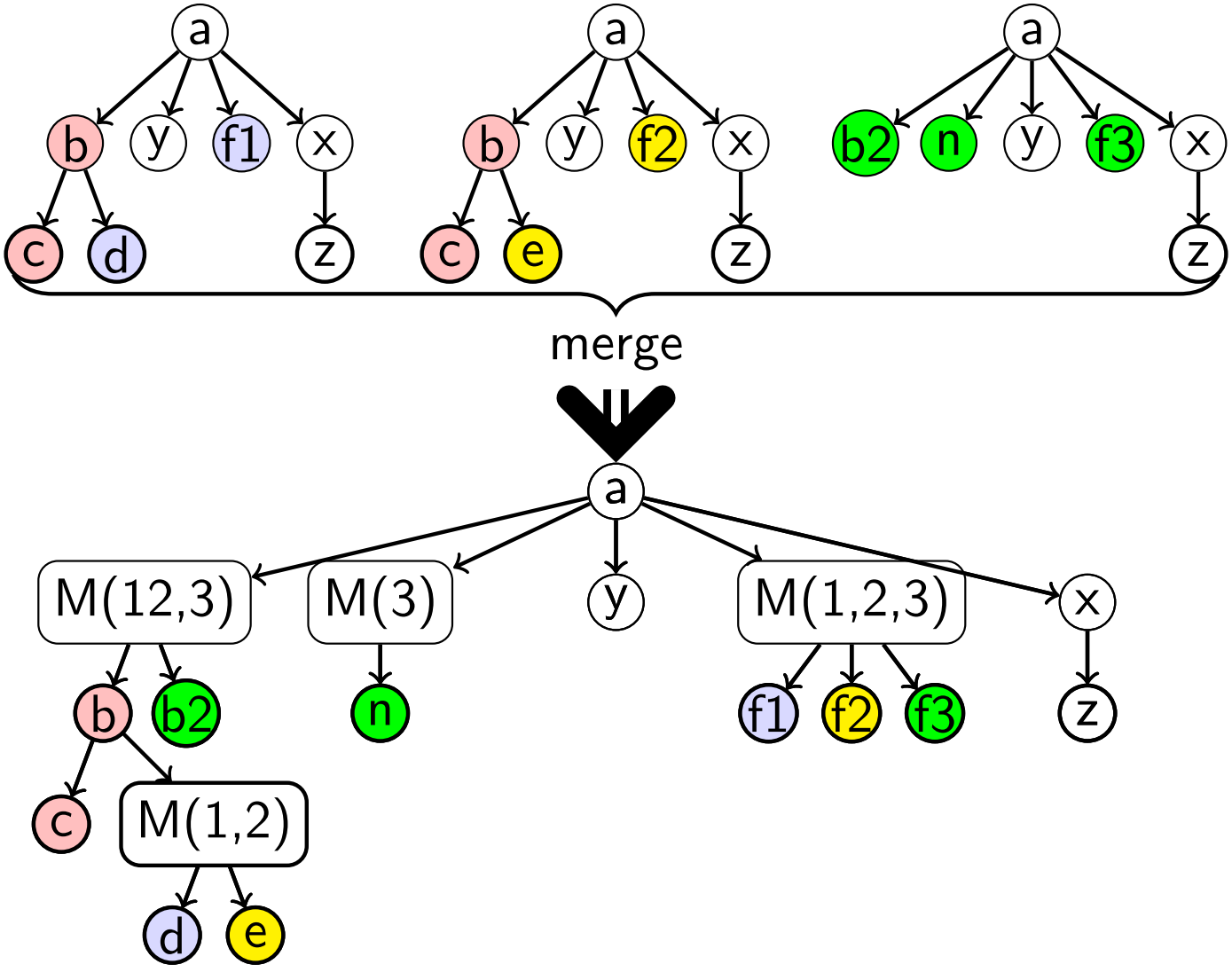

```
void function3()
{
   db_init();
   db_sanity_check();
   run_mode = MODE_REGULAR;
   start(&service);
}
```



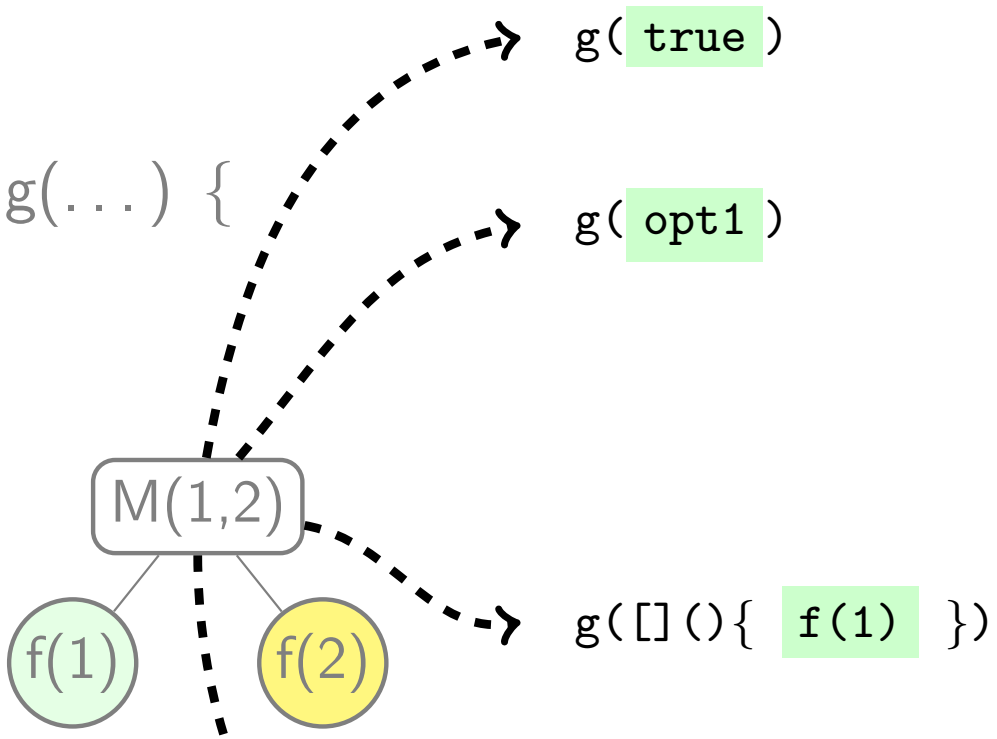

```
void function1()
```

```
void fnMerged(int functionId,
```

Automating Abstraction



Merge Strategies




```
g( true )
```

```
g( opt1 )
```

```
g([](){ f(1) })
```

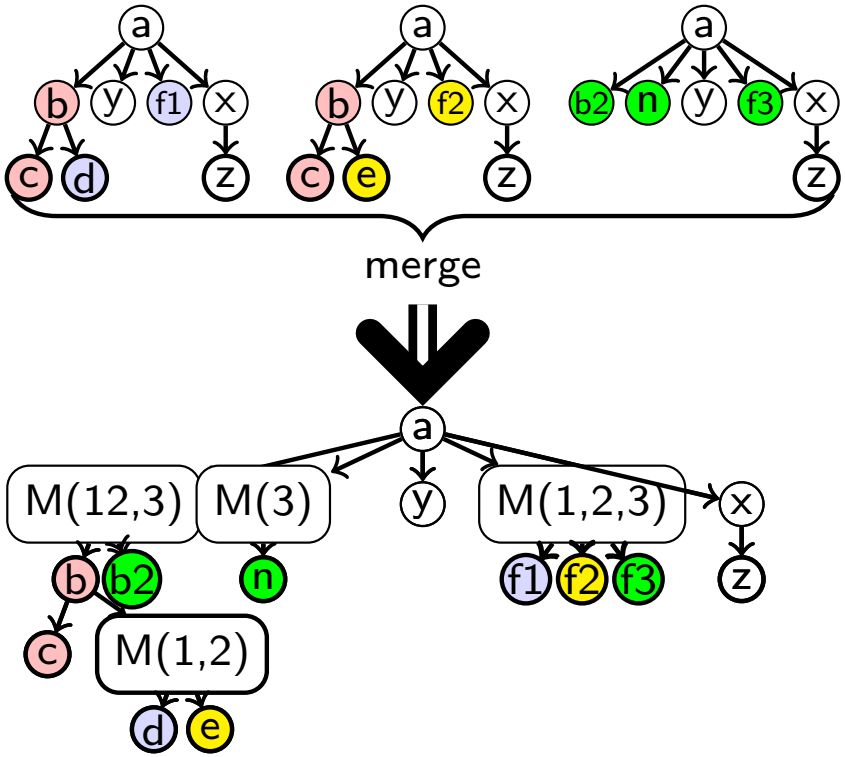
```
g(new ArgObj() {
  int getf() {
    return f(1) ;
  }
});
```

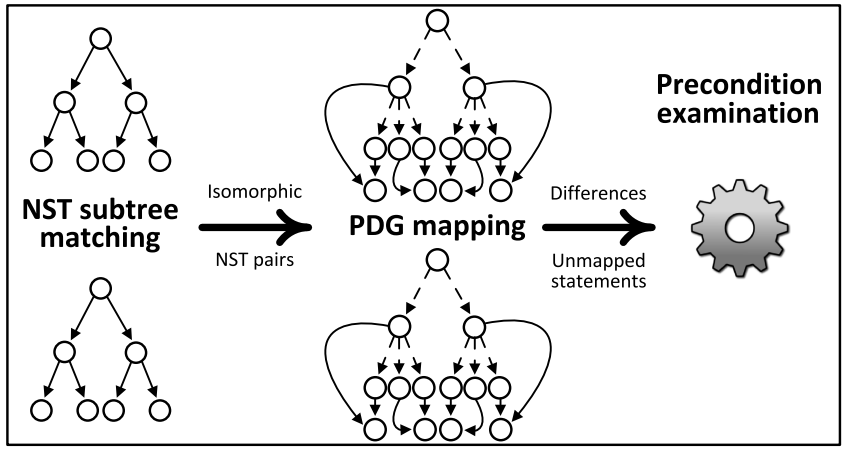
```
void g(boolean opt)
      if opt  f(1)  else  f(2)
      ...
void g(enumty opt)
      switch (opt) {
      case opt1: f(1) ; break;
      case opt2: f(2) ; break;
      }
      ...
```

```
void g(int (*fun)(void))
      x = fun();
      ...
```

```
void g(ArgObj arg)
      x = arg.getf();
      ...
```

Clone Fusion Approaches





Cleaning up copy–paste clones with interactive merging

Krishna Narasimhan[1] · Christoph Reichenbach[2] · Julia Lawall[3]

Received: 19 September 2016 / Accepted: 11 June 2018

Abstract
Copy-paste-modify is a form of software reuse in which developers explicitly duplicate source code. This duplicated source code, amounting to a code clone, is adapted for a new purpose. Copy-paste-modify is popular among software developers, however, empirical evidence shows that it complicates software maintenance and increases the frequency of bugs. To allow developers to use copy-paste-modify without having to

Tsantalis, Krishnan, Mazinanian:

- Alternative approach
- *JDeodorant* tool
- Combined with clone detection

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 41, NO. 11, NOVEMBER 2015 1055

Assessing the Refactorability of Software Clones

Nikolaos Tsantalis, *Member, IEEE*, Davood Mazinanian, and Giri Panamoottil Krishnan

Abstract—The presence of duplicated code in software systems is significant and several studies have shown that clones can be potentially harmful with respect to the maintainability and evolution of the source code. Despite the significance of the problem, there is

Research Results

- Selected near-clones from Open Source C++ projects (Facebook rocksdb, Google protobuf, Oracle node-oracledb, mongodb, . . .)
- Used our tool to merge
- Manual cleanup: Formatting, variable renaming
- Phase 1 (Early prototype, limited to two-way merge):

Submitted	**Accepted**	**Rejected**	**Pending**
8	1	4	3

  $\Rightarrow$ Feedback: *Need multi-way merge, separate merge-points*

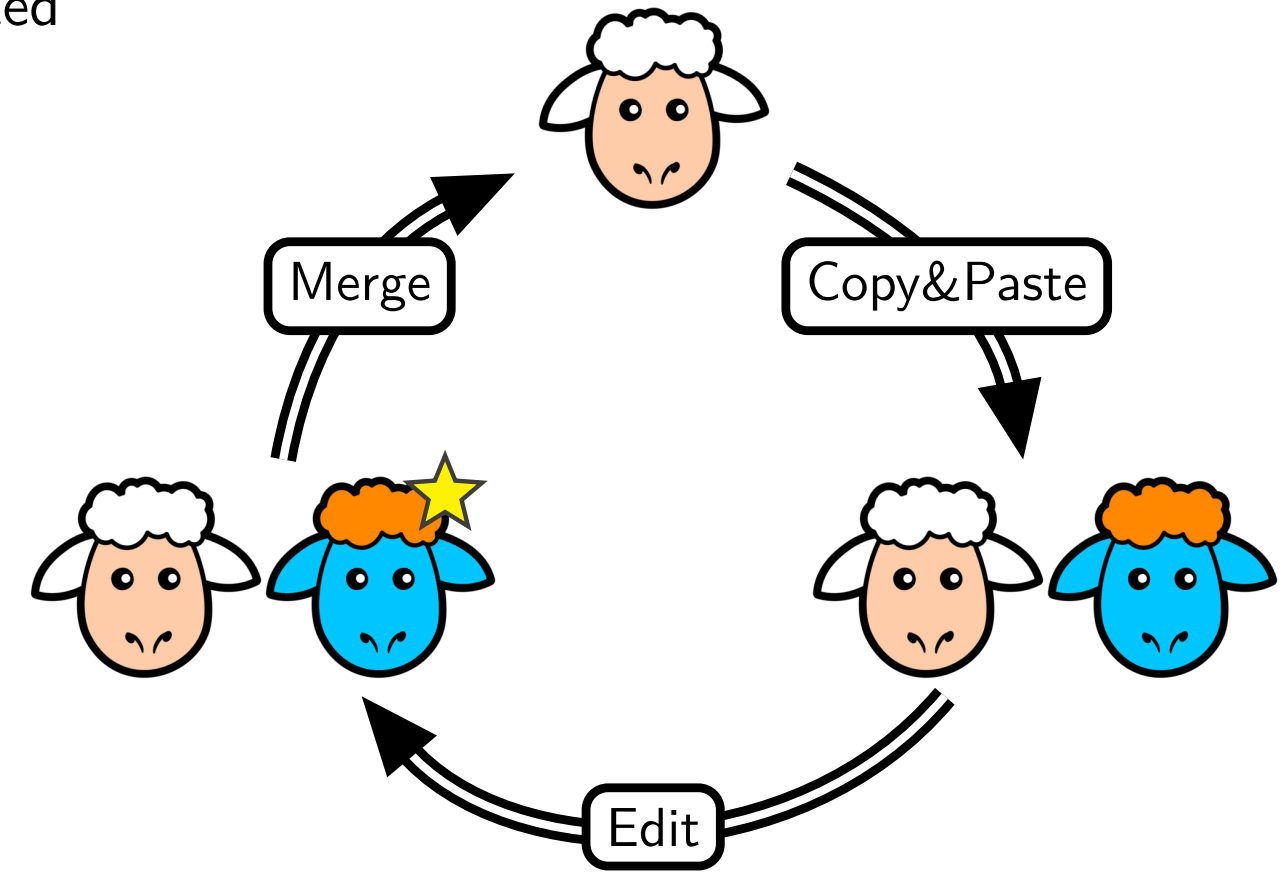
- Phase 2:

Submitted	**Accepted**	**Rejected**	**Pending**
10	9	0	1

Effective as a Refactoring Tool

Summary: Copy-Paste-Edit-Merge Cycle

- Clones are technical debt
- Clone-merging can be atomated
- Automated merge support WIP
- Initial results promising



http://transparent-abstractions.org