Feature Trace Recording

Paul Maximilian Bittner[1]

Alexander Schultheiß[2]

Thomas Thüm[1]

Timo Kehrer[2]

Jeffrey M. Young[3]

Lukas Linsbauer[4]

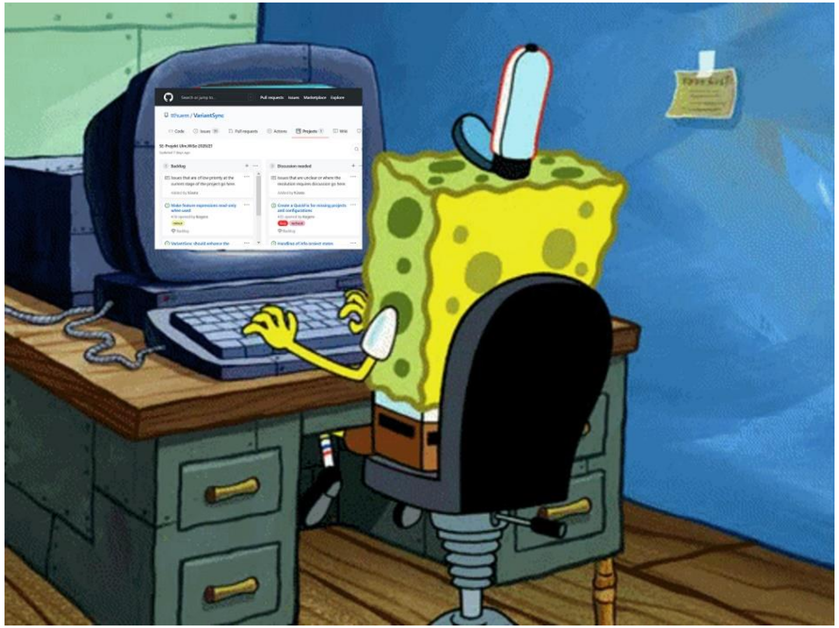
1 ulm university universität ulm

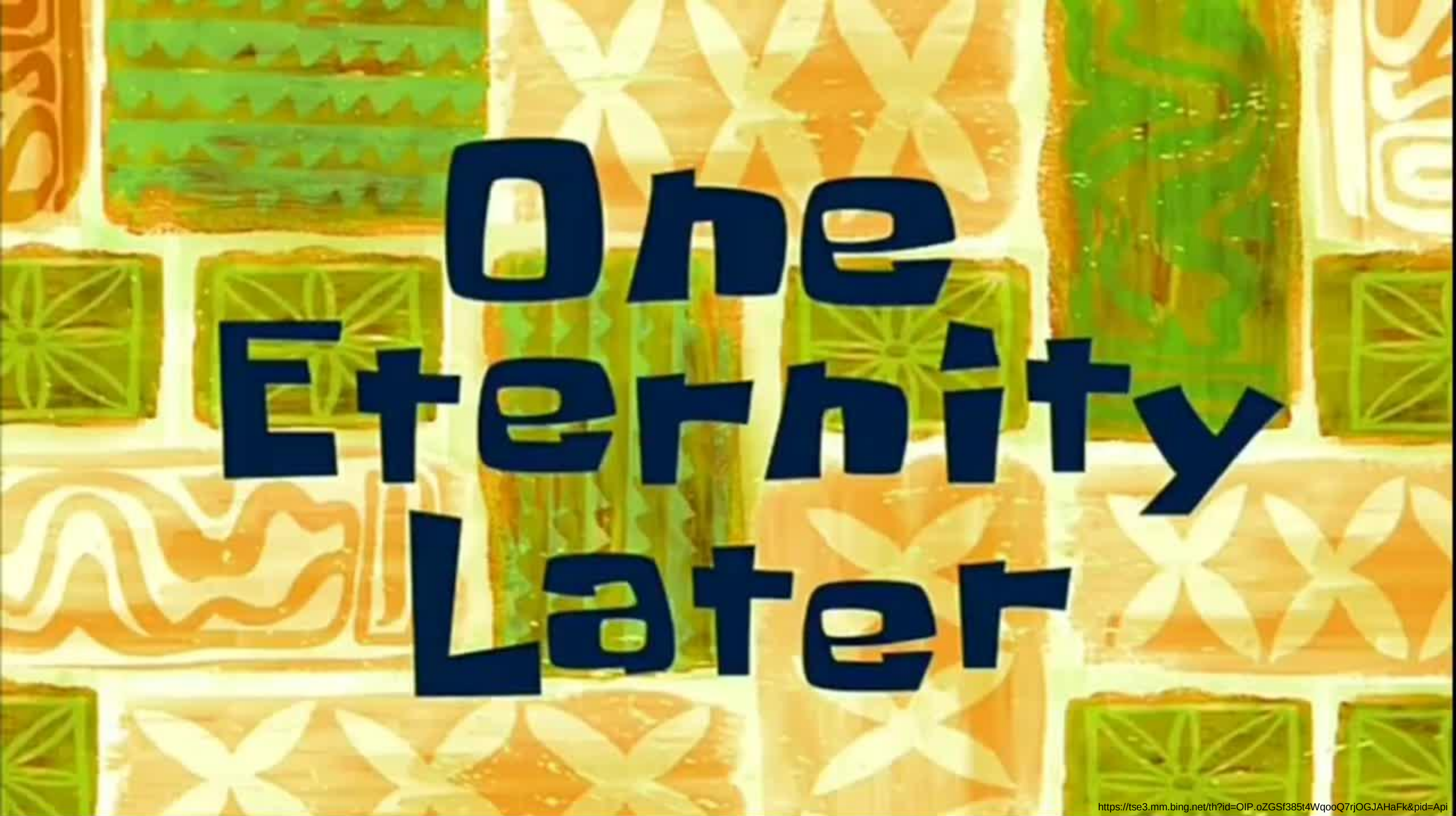
2 Humboldt-Universität zu Berlin

3 Oregon State University

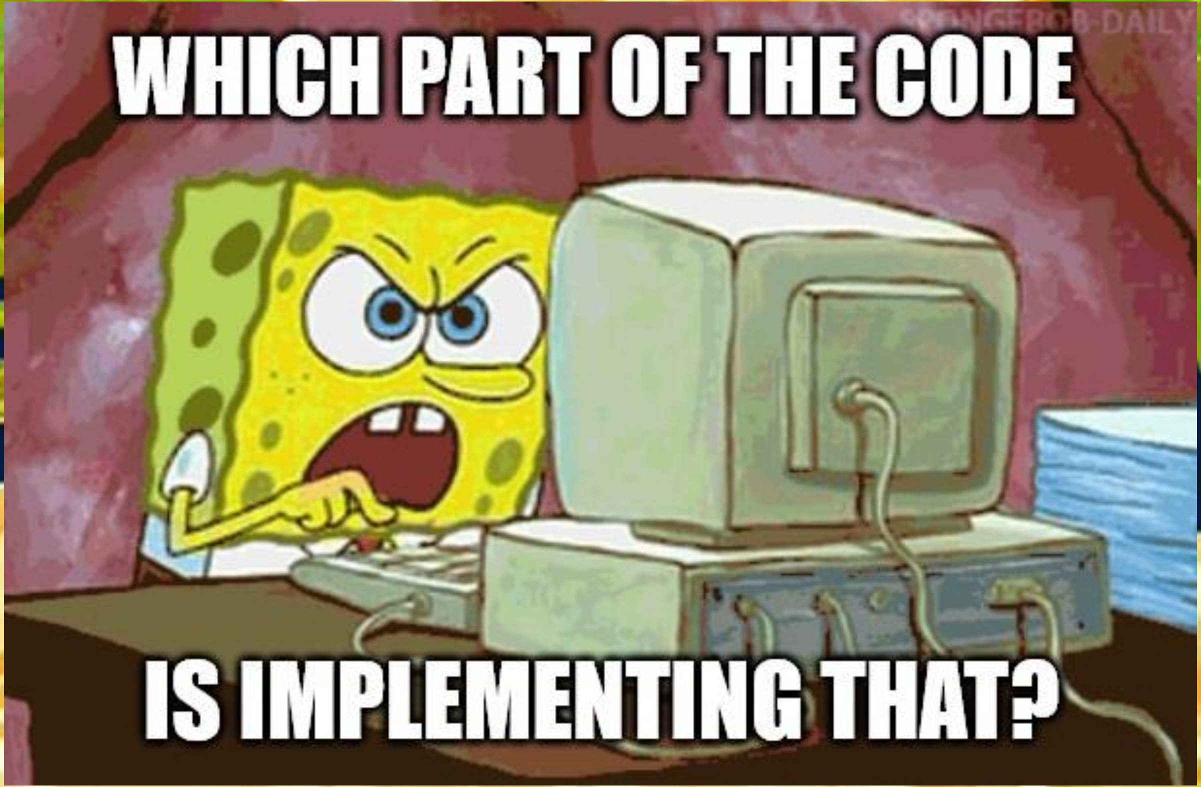
4 Technische Universität Braunschweig

YOUR NEW EMPLOYEE





https://tse3.mm.bing.net/th?id=OIP.oZGSf385t4WqooQ7rjOGJAHaFk&pid=Api

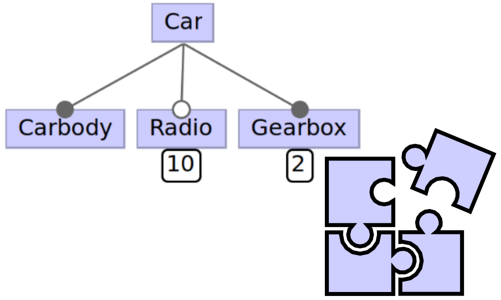


https://tse3.mm.bing.net/th?id=OIP.oZGSf385t4WqooQ7rjOGJAHaFk&pid=Api

Feature Traceability Problem

Feature Traceability is the knowledge where each feature is implemented.

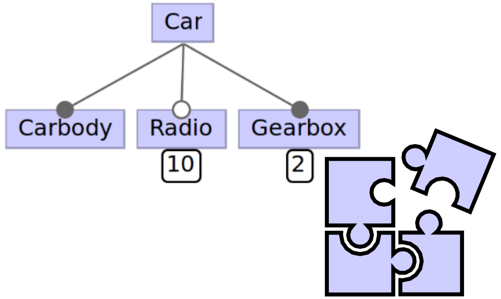
Traceability is given in *software product lines …*



complete
feature traces

Traceability is given in *software product lines …*

… but in practice variability is often implemented with *clone-and-own.*

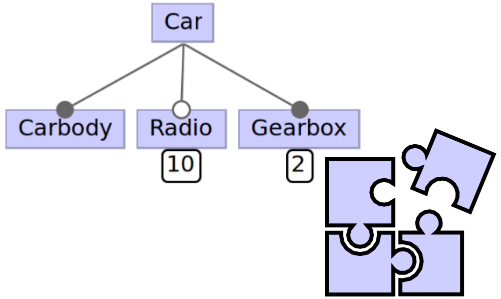


complete feature traces

(almost) no feature traces

Traceability is given in *software product lines* …

… but in practice variability is often implemented with *clone-and-own.*



complete feature traces

(almost) no feature traces

So how can we help developers to *document* and *maintain* feature traces here?

Feature traces can be documented …

Retroactively: after development (Variability Mining [Kästner et al.])

Requires to halt development

Not always possible because knowledge is lost

Proactively: during development (Embedded Annotations [Ji et al.])

No automation yet

→ Feature Trace Recording

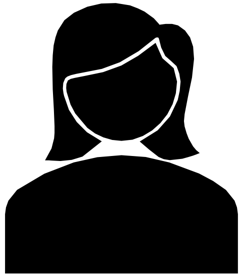
Example of Feature Trace Recording

This crashes when the stack is empty!
Alice

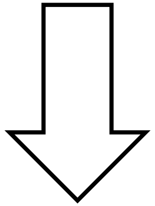
```
class Stack {

    /* ... */

    void pop() {
      storage[head--] = null;
    }
}
```

Software Engineering
Programming Languages

Example of Feature Trace Recording

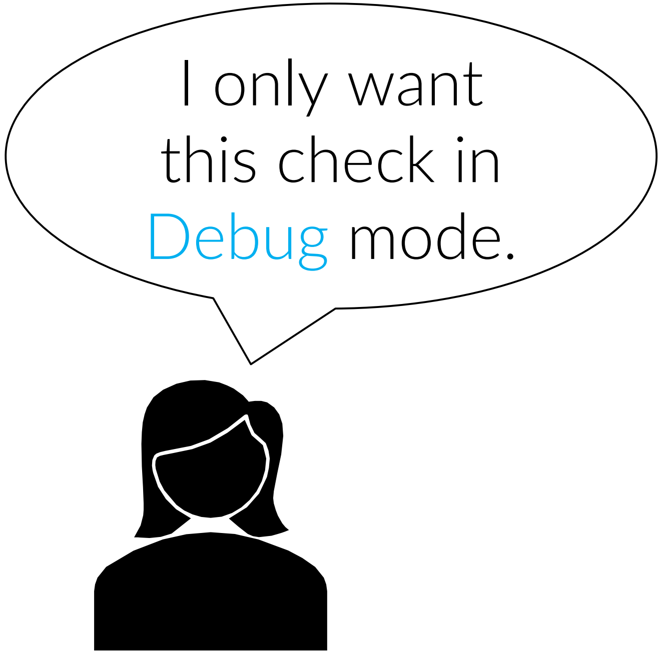


```
void pop() {
  storage[head--] = null;
}
```



```
void pop() {
  if (!empty()) {}
  storage[head--] = null;
}
```

Example of Feature Trace Recording



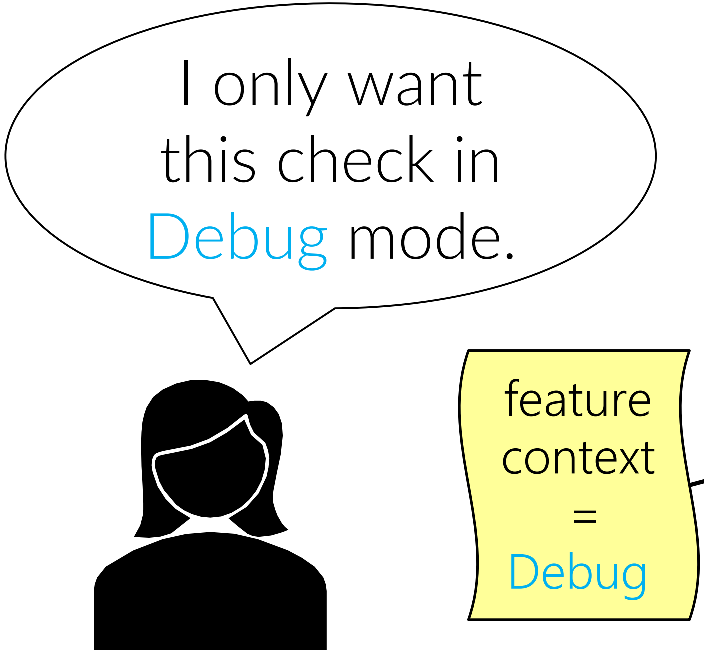
```
void pop() {
  storage[head--] = null;
}
```

⇩

```
void pop() {
  if (!empty()) {}
  storage[head--] = null;
}
```

Software Engineering
Programming Languages

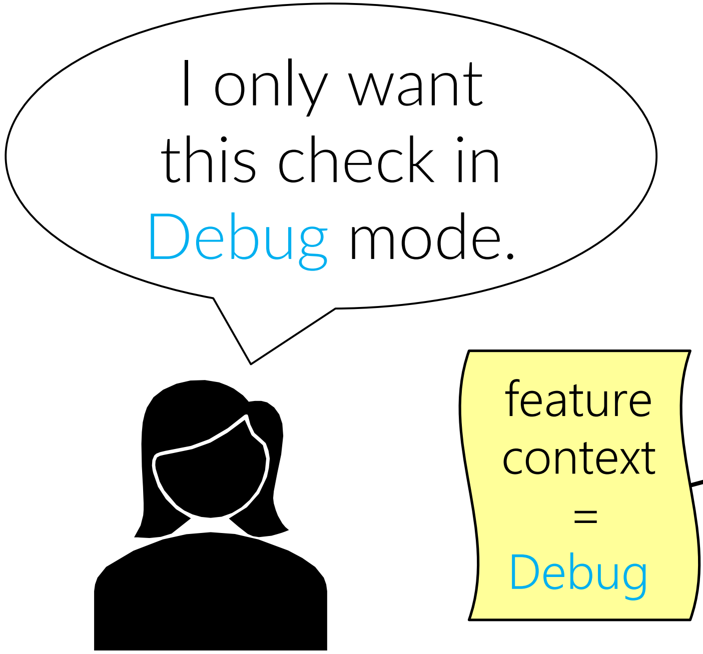
Example of Feature Trace Recording



```
void pop() {
  storage[head--] = null;
}
```

```
void pop() {
  if (!empty()) {}
  storage[head--] = null;
}
```

Example of Feature Trace Recording

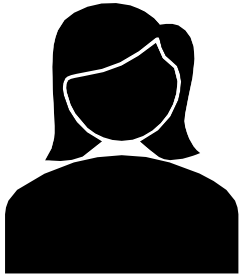


```
void pop() {
  storage[head--] = null;
}
```

```
void pop() {
  if (!empty()) {}
  storage[head--] = null;
}
```

Software Engineering
Programming Languages

Example of Feature Trace Recording

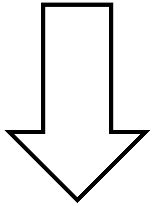


```
void pop() {
  if (!empty()) {}
  storage[head--] = null;
}
```

```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

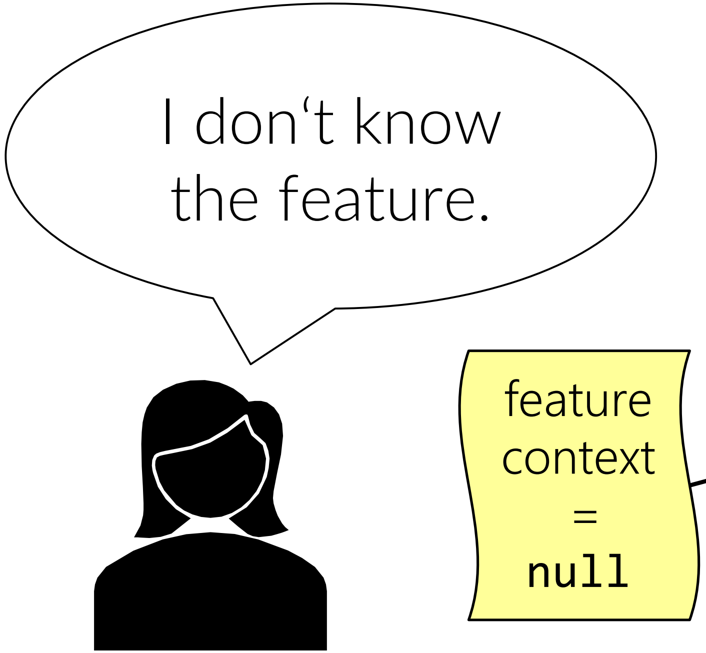
Example of Feature Trace Recording

```
void pop() {
  if (!empty()) {}
  storage[head--] = null;
}
```



```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

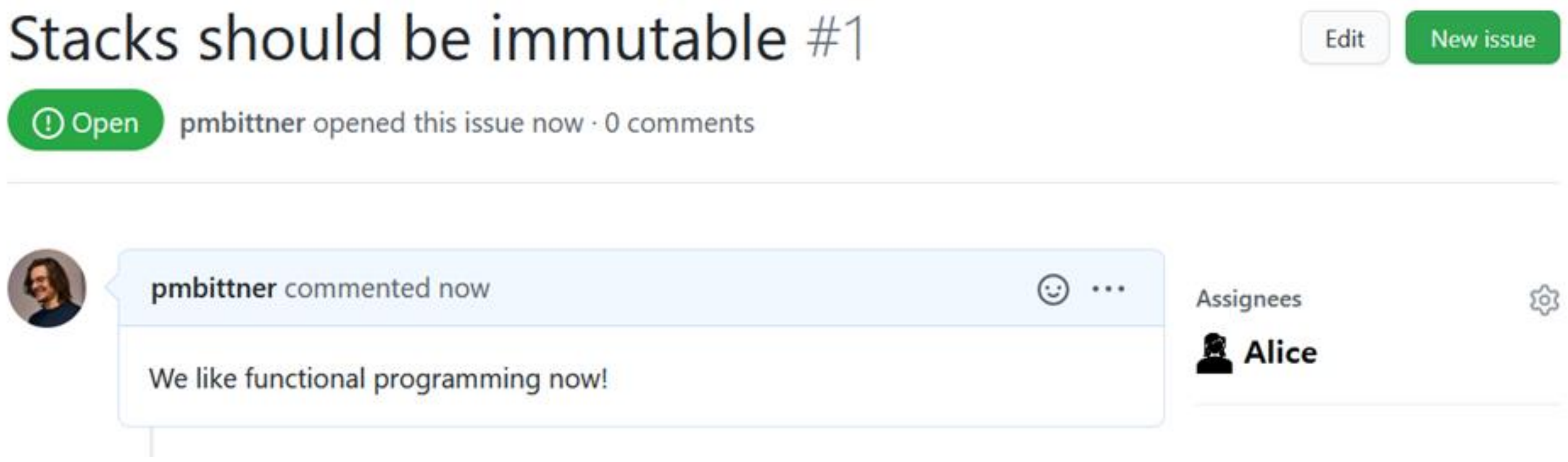
Example of Feature Trace Recording



```
void pop() {
   if (!empty()) {}
   storage[head--] = null;
}
```

```
void pop() {
   if (!empty()) {
      storage[head--] = null;
   }
}
```

Example of Feature Trace Recording – The Next Week


Stacks should be immutable #1

Edit

New issue

Open pmbittner opened this issue now · 0 comments

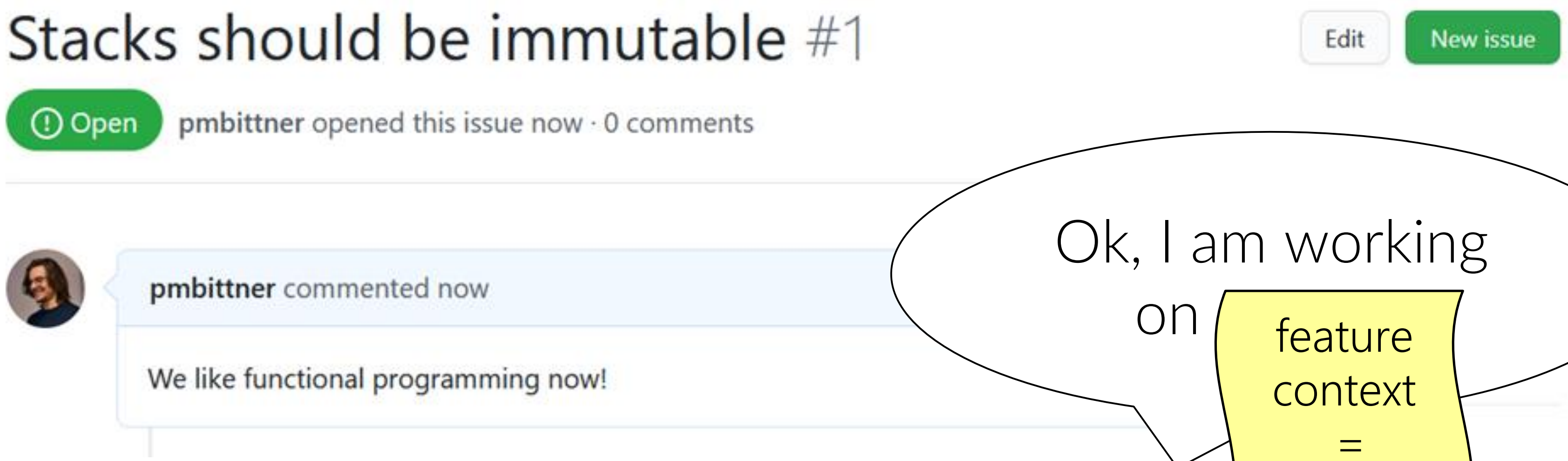
pmbittner commented now

We like functional programming now!

Assignees

Alice

Example of Feature Trace Recording – The Next Week

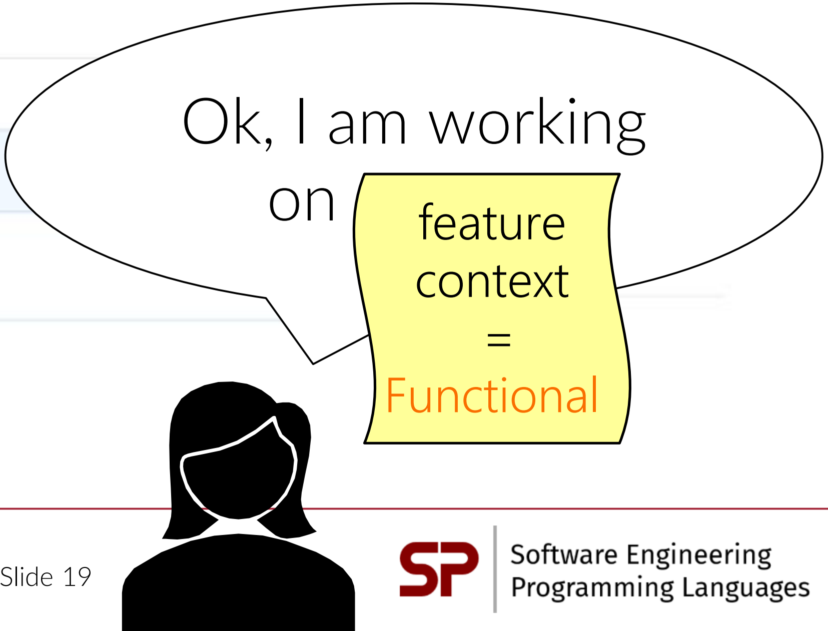


Stacks should be immutable #1

pmbittner opened this issue now · 0 comments

pmbittner commented now

We like functional programming now!



```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

$\xrightarrow{\text{delete}}$

```
void pop() {
  if (!empty()) {

  }
}
```

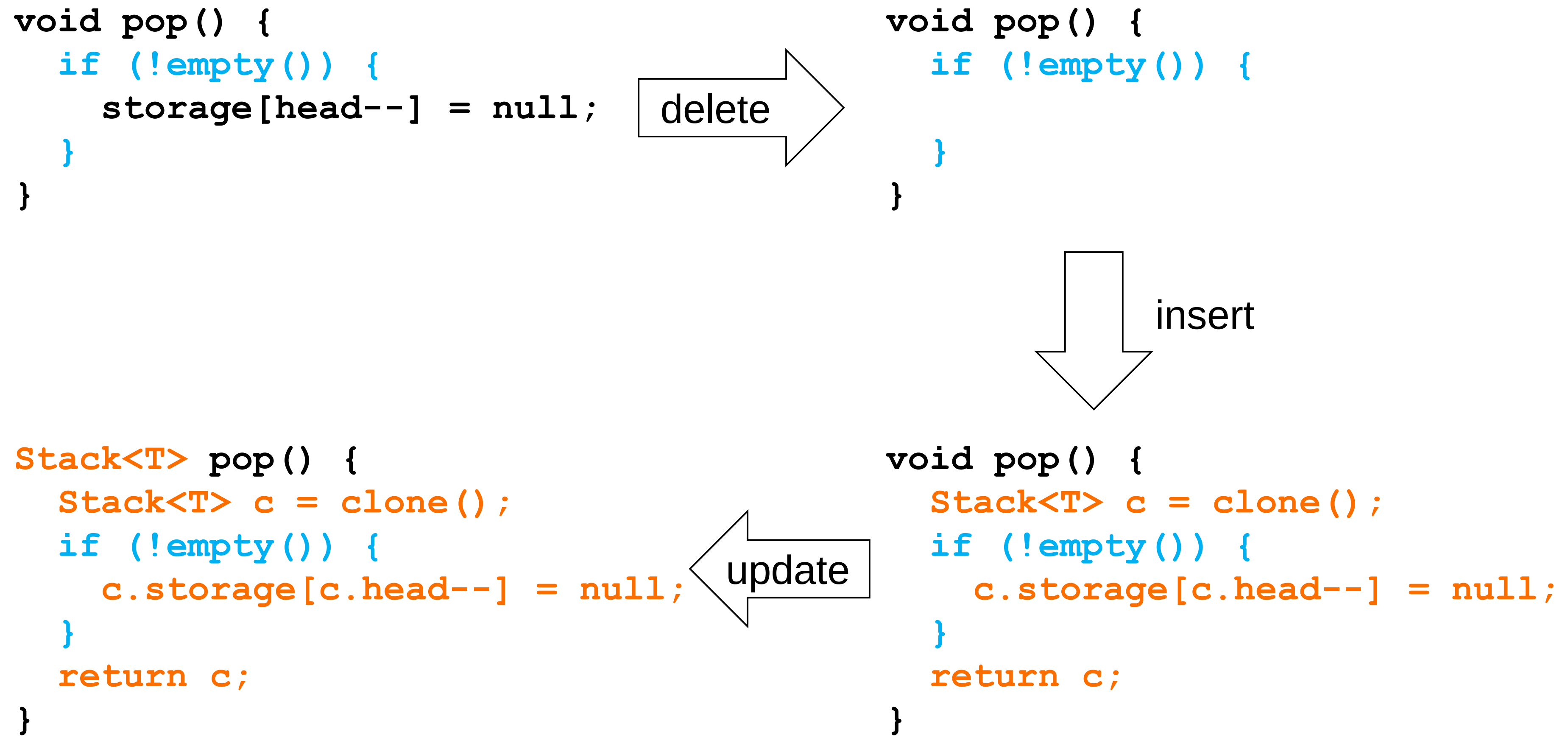
```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

delete

```
void pop() {
  if (!empty()) {

  }
}
```

insert

```
void pop() {
  Stack<T> c = clone();
  if (!empty()) {
    c.storage[c.head--] = null;
  }
  return c;
}
```



```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

```
void pop() {
  if (!empty()) {

  }
}
```

```
void pop() {
  Stack<T> c = clone();
  if (!empty()) {
    c.storage[c.head--] = null;
  }
  return c;
}
```

```
Stack<T> pop() {
  Stack<T> c = clone();
  if (!empty()) {
    c.storage[c.head--] = null;
  }
  return c;
}
```

```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

delete

```
void pop() {
  if (!empty()) {

  }
}
```

insert

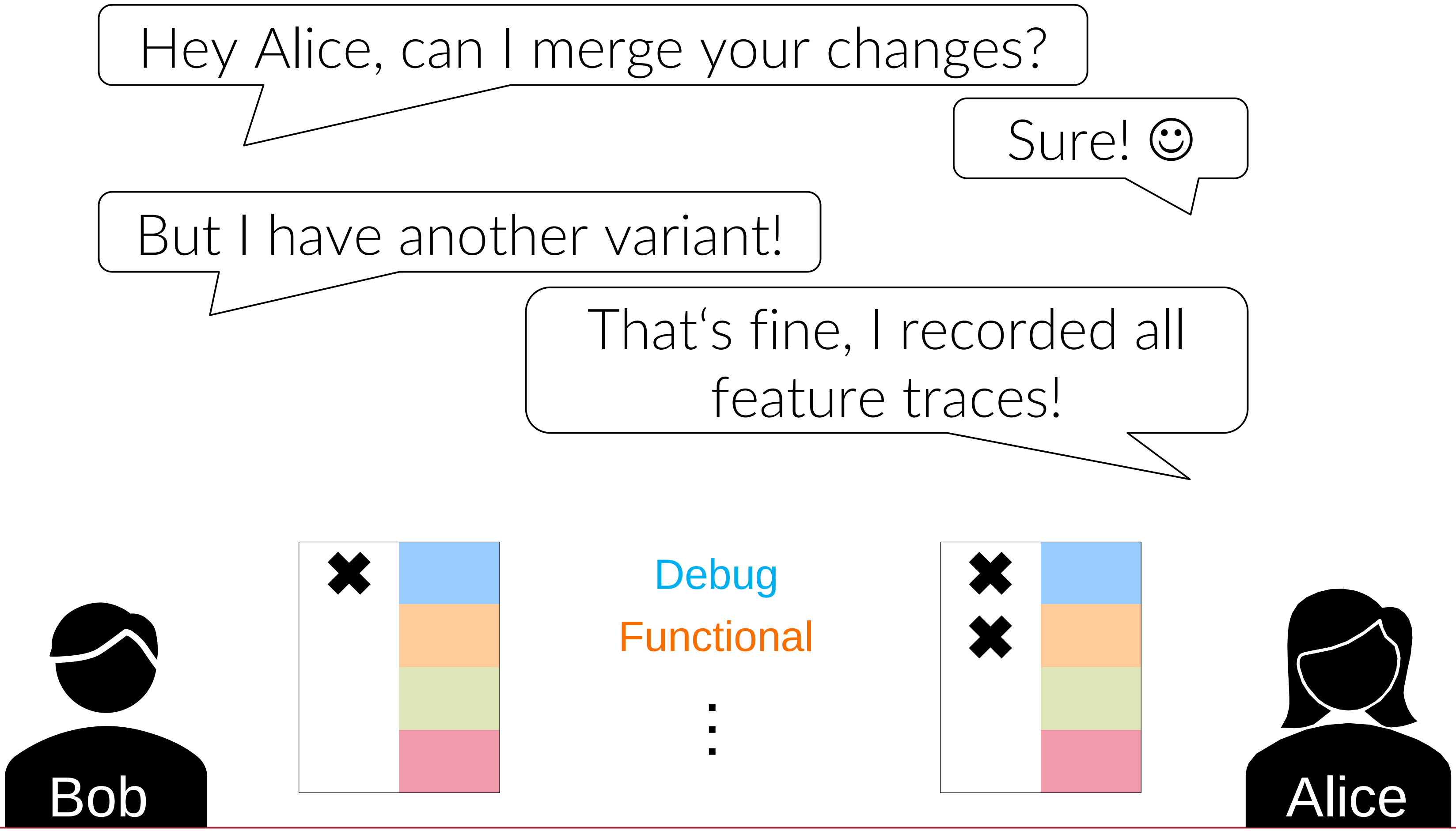
```
void pop() {
  Stack<T> c = clone();
  if (!empty()) {
    c.storage[c.head--] = null;
  }
  return c;
}
```

update

```
Stack<T> pop() {
  Stack<T> c = clone();
  if (!empty()) {
    c.storage[c.head--] = null;
  }
  return c;
}
```

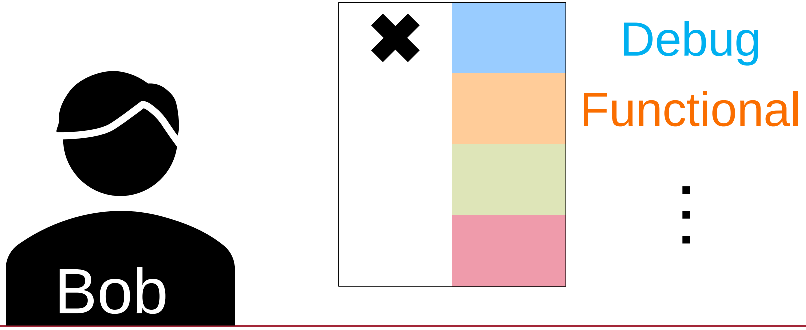
done with single

feature context = Functional


Hey Alice, can I merge your changes?
Sure! ☺
But I have another variant!
That‘s fine, I recorded all feature traces!
Debug
Functional
⋮
Bob
Alice

```
void pop() {/*...*/}
```

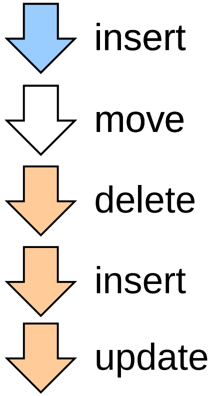
```
void pop() {/*...*/}
```



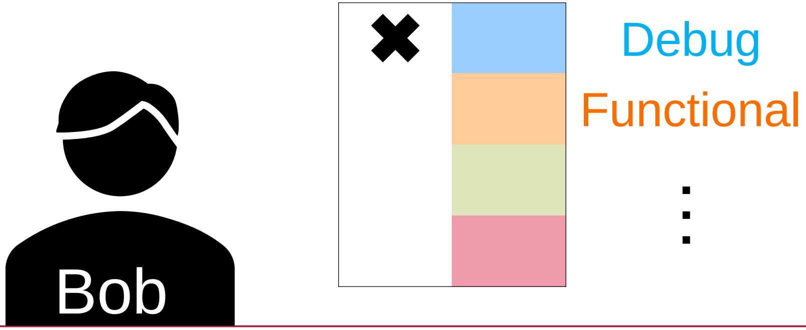
Software Engineering
Programming Languages

```
void pop() {/*...*/}
```

```
void pop() {/*...*/}
```

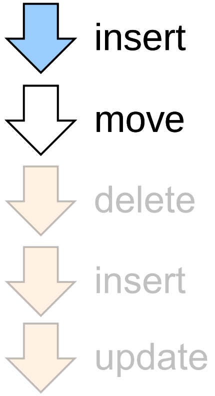


```
Stack<T> pop() {  /*...*/}
```

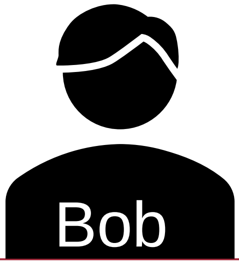


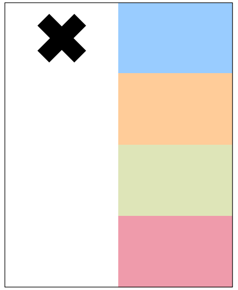
```
void pop() {/*...*/}
```

```
void pop() {/*...*/}
```



```
Stack<T> pop() {  /*...*/}
```





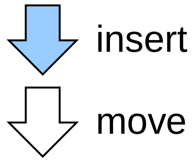
Debug

Functional

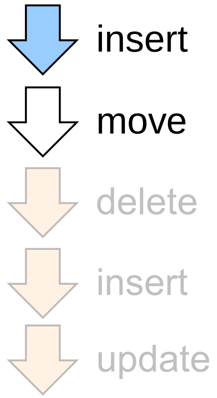
⋮

Software Engineering
Programming Languages

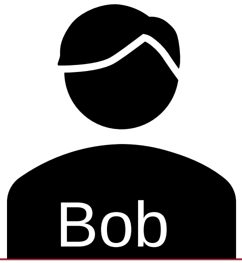
```
void pop() {/*...*/}
```

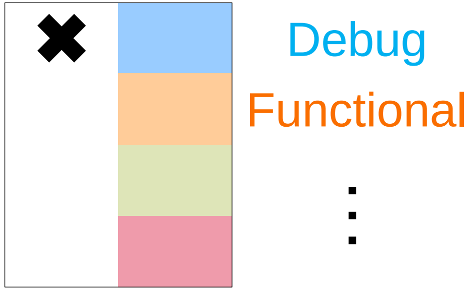


```
void pop() {/*...*/}
```



```
Stack<T> pop() {  /*...*/}
```





```
void pop() {/*...*/}
```

insert

move

```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

```
void pop() {/*...*/}
```

insert

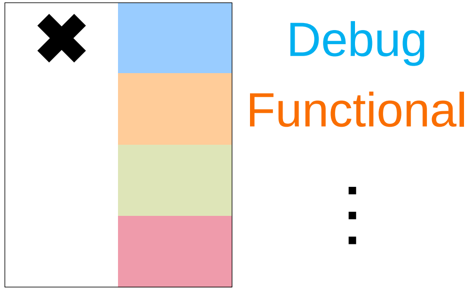
move

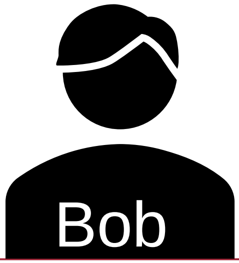
delete

insert

update

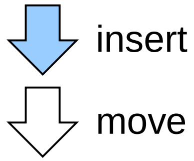
```
Stack<T> pop() {  /*...*/}
```



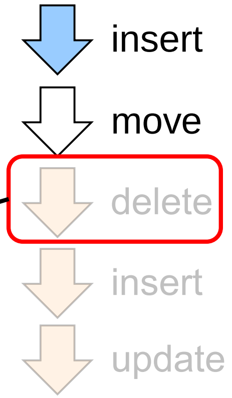


Software Engineering
Programming Languages

```
void pop() {/*...*/}
```

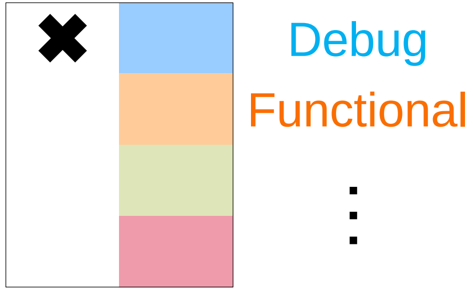


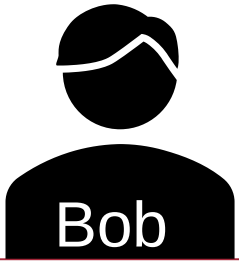
```
void pop() {/*...*/}
```

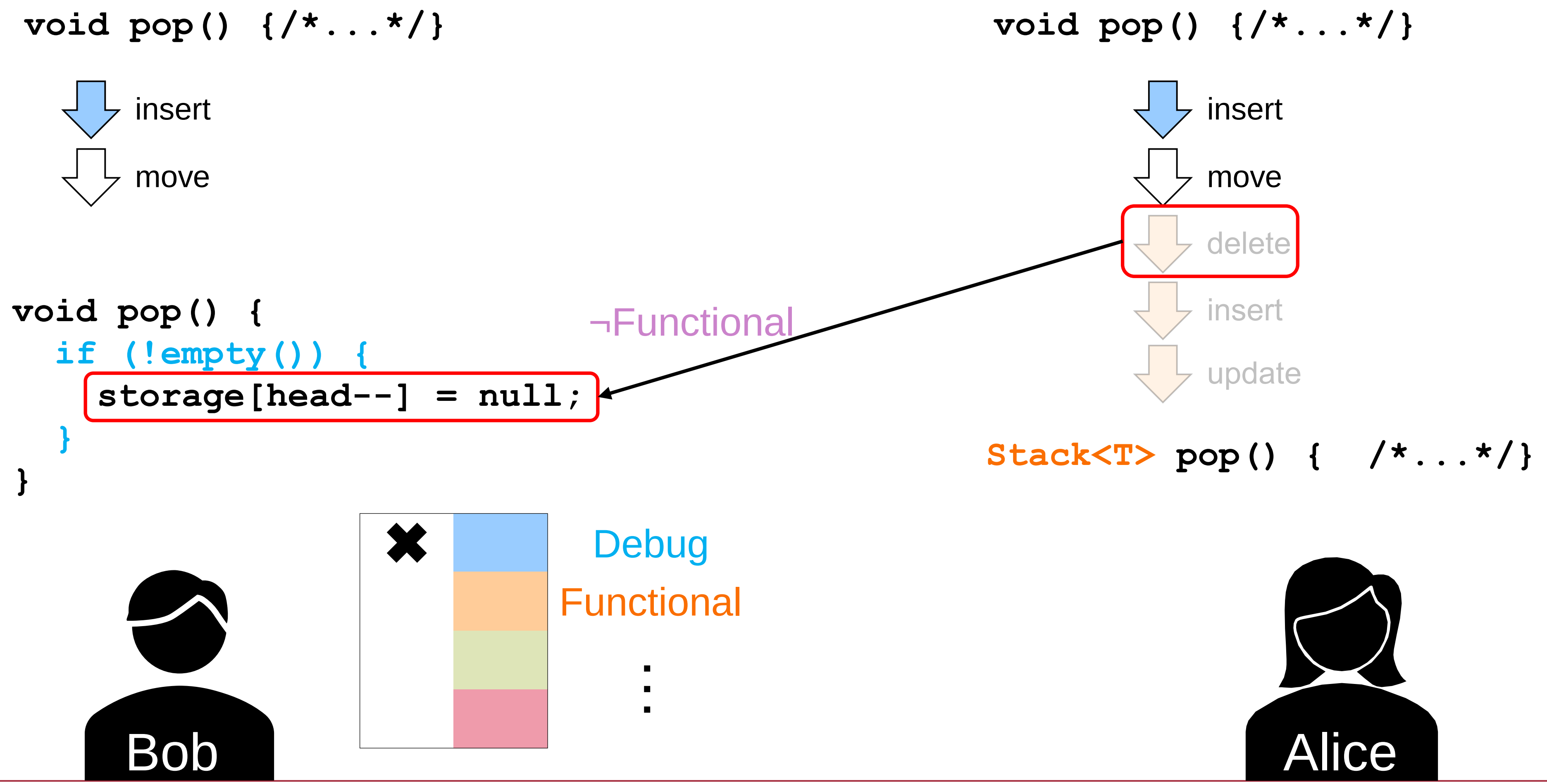


```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

```
Stack<T> pop() {  /*...*/}
```





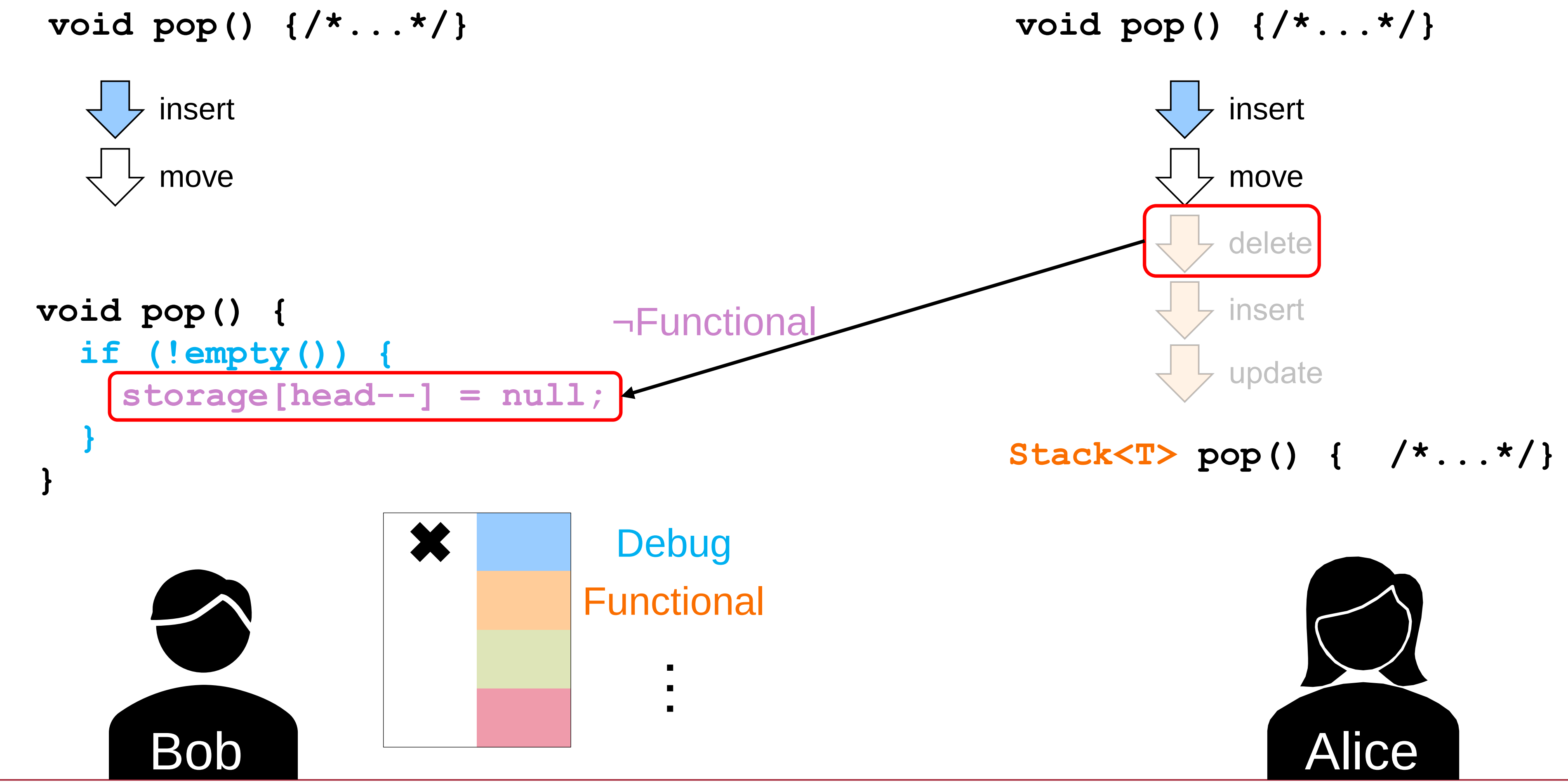


```
void pop() {/*...*/}
```

```
void pop() {/*...*/}
```

```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

```
Stack<T> pop() {  /*...*/}
```



```
void pop() {/*...*/}
```

```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

```
void pop() {/*...*/}
```

```
Stack<T> pop() {  /*...*/}
```

```
void pop() {/*...*/}
```

- insert
- move

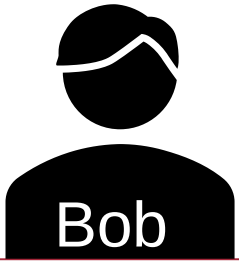
```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

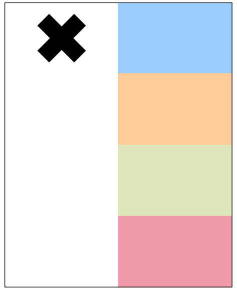
¬Functional

```
void pop() {/*...*/}
```

- insert
- move
- delete
- insert
- update

```
Stack<T> pop() {  /*...*/}
```





Debug

Functional

⋮

Software Engineering
Programming Languages

```
void pop() {/*...*/}
```

insert

move

```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

¬Functional

```
void pop() {/*...*/}
```

insert

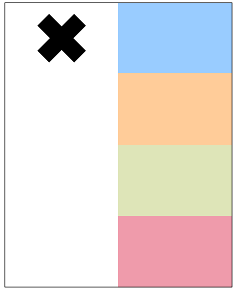
move

delete

insert

update

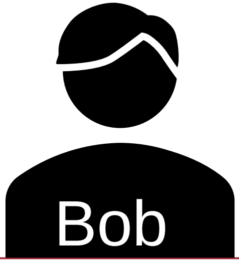
```
Stack<T> pop() {  /*...*/}
```

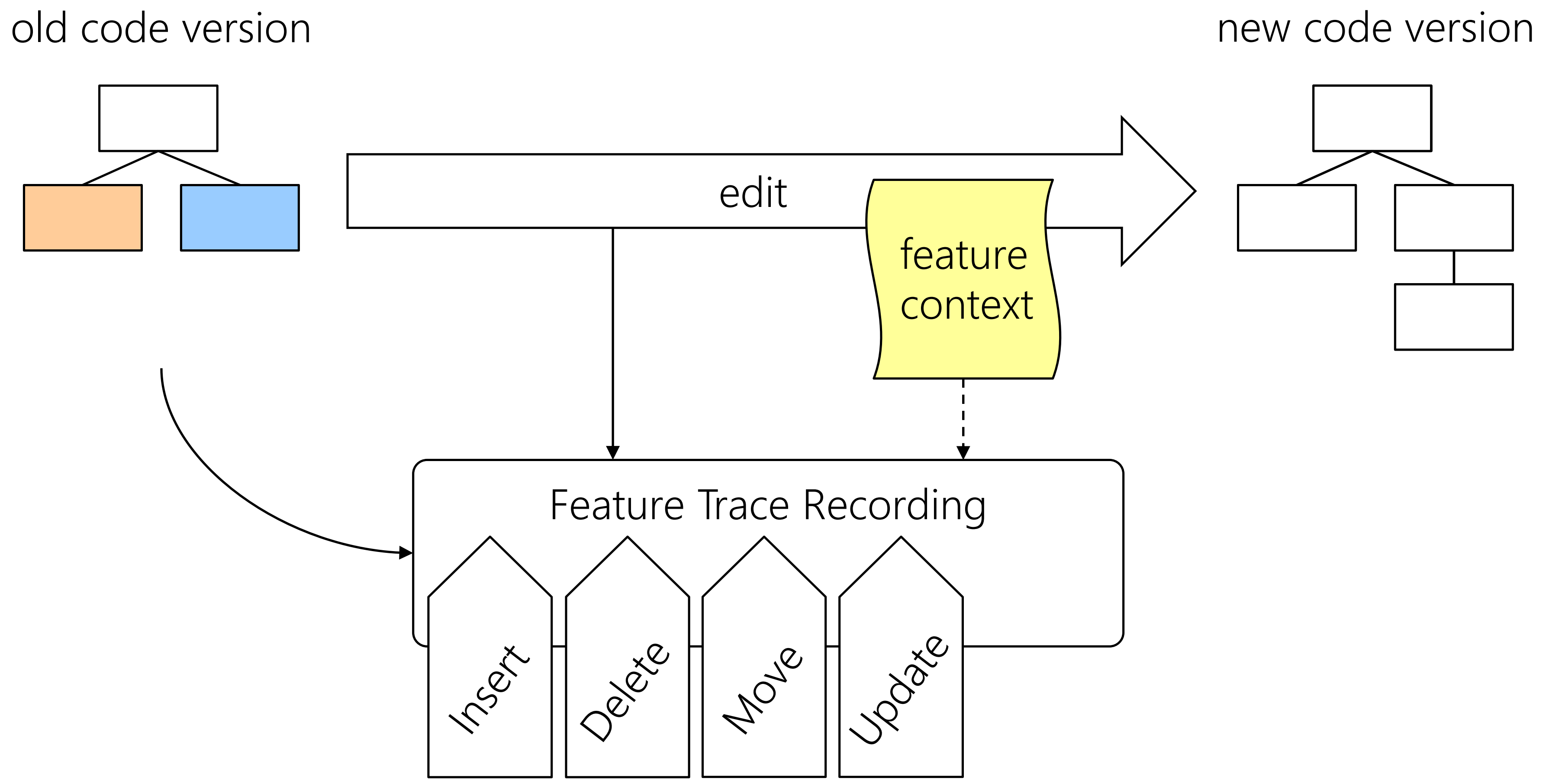


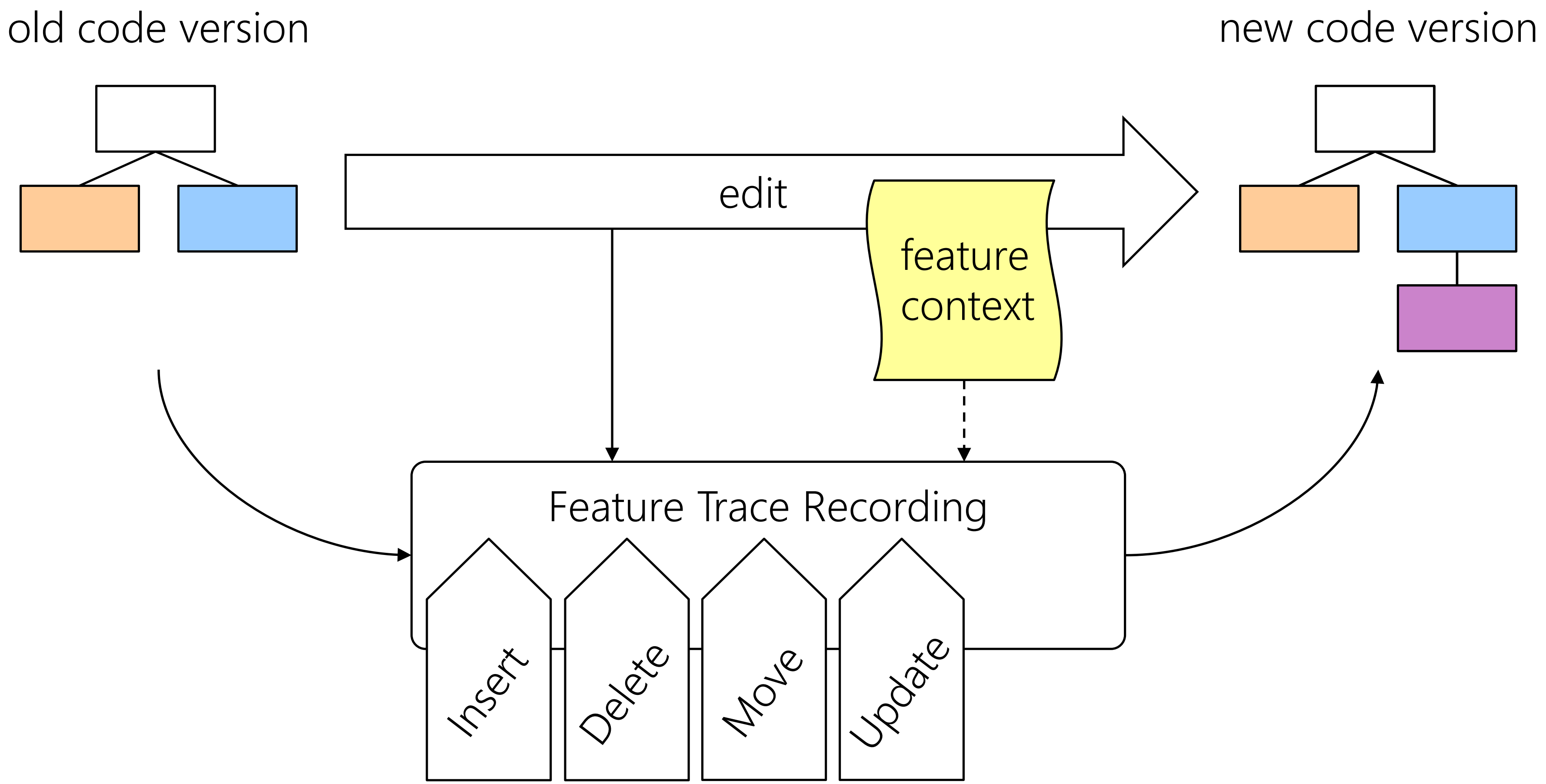
Debug

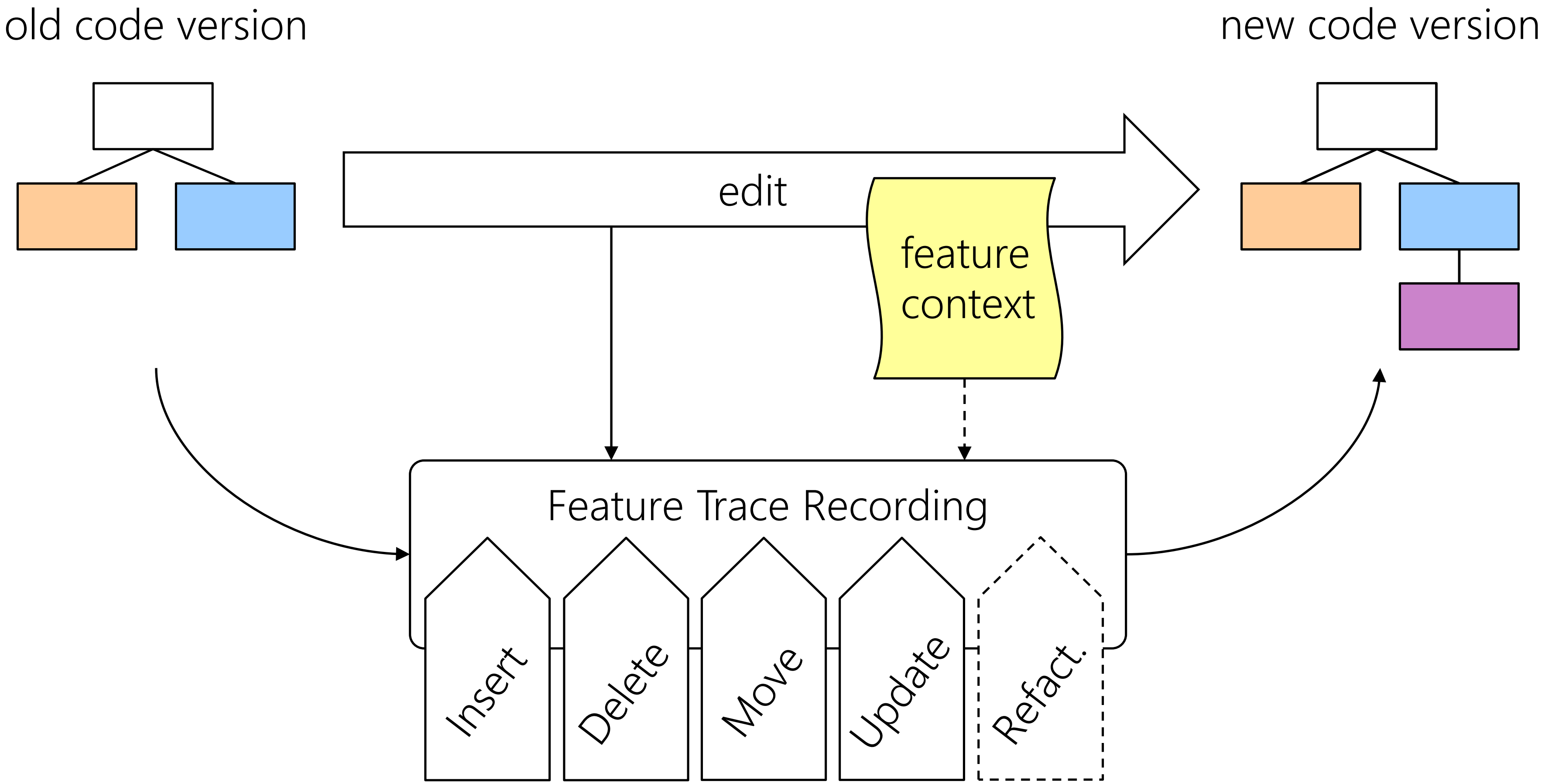
Functional

⋮



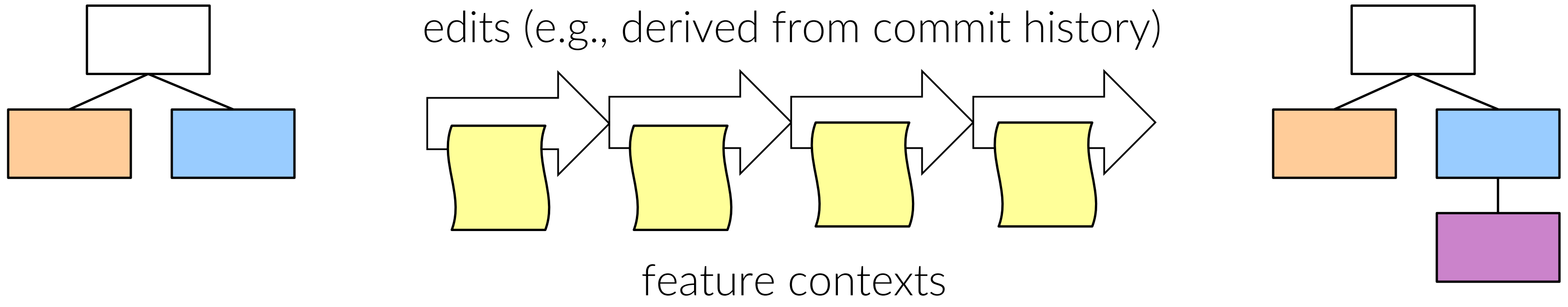

old code version
new code version
edit
feature context
Feature Trace Recording
Insert
Delete
Move
Update


old code version
new code version
edit
feature context
Feature Trace Recording
Insert
Delete
Move
Update

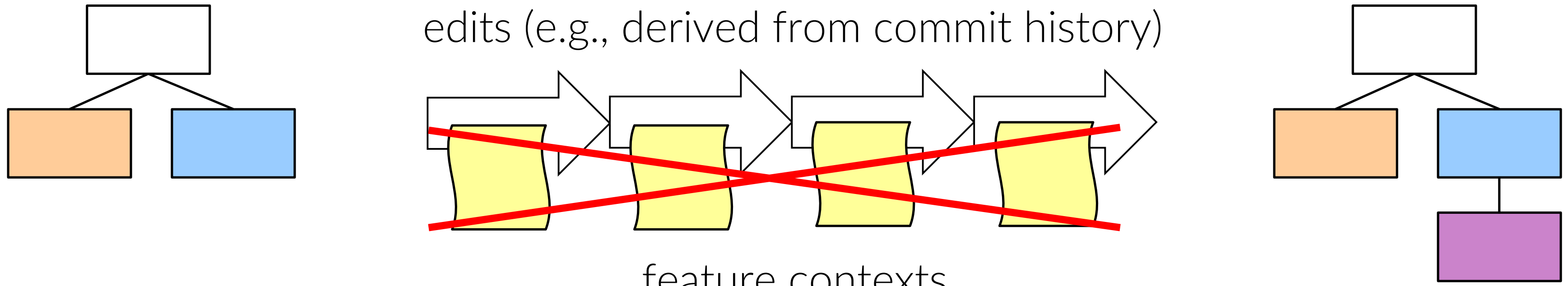


Software Engineering
Programming Languages

To evaluate feature trace recording we need

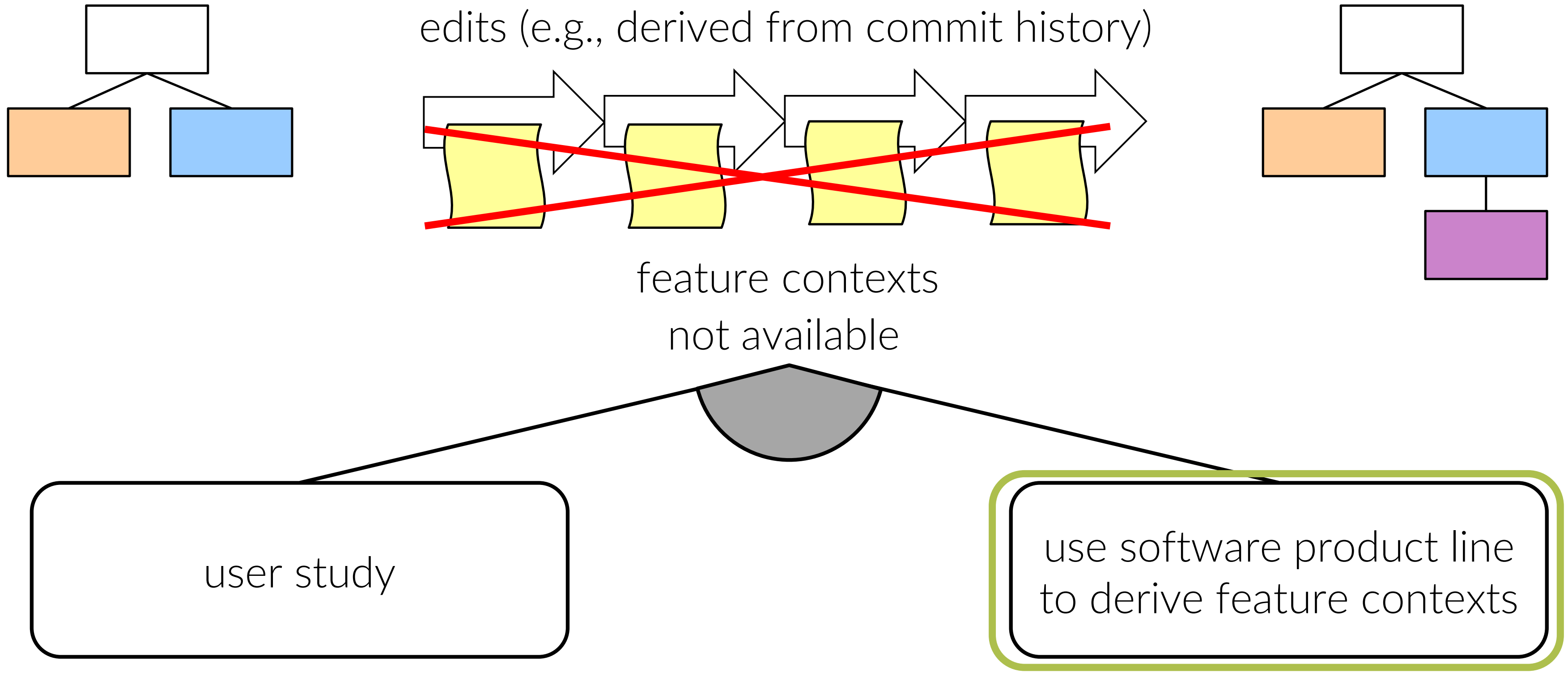


To evaluate feature trace recording we need

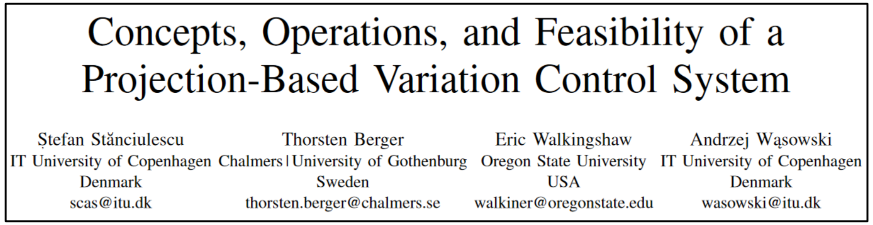


Software Engineering
Programming Languages

To evaluate feature trace recording we need

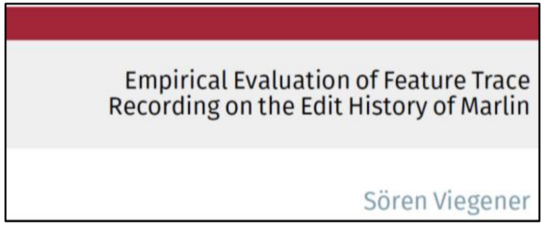

edits (e.g., derived from commit history)
feature contexts not available
user study
use software product line to derive feature contexts

Can we reproduce edits to SPLs as edits to variants?



Concepts, Operations, and Feasibility of a Projection-Based Variation Control System

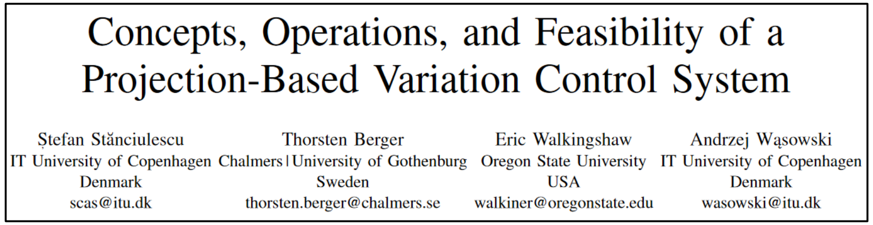
Ștefan Stănciulescu, IT University of Copenhagen, Denmark, scas@itu.dk
Thorsten Berger, Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se
Eric Walkingshaw, Oregon State University, USA, walkiner@oregonstate.edu
Andrzej Wąsowski, IT University of Copenhagen, Denmark, wasowski@itu.dk



Empirical Evaluation of Feature Trace Recording on the Edit History of Marlin

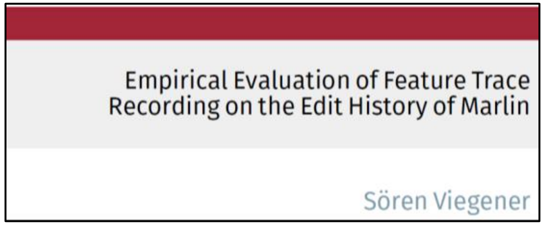
Sören Viegener

Can we reproduce edits to SPLs as edits to variants?



Concepts, Operations, and Feasibility of a Projection-Based Variation Control System

Ştefan Stănciulescu, IT University of Copenhagen, Denmark, scas@itu.dk
Thorsten Berger, Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se
Eric Walkingshaw, Oregon State University, USA, walkiner@oregonstate.edu
Andrzej Wąsowski, IT University of Copenhagen, Denmark, wasowski@itu.dk



Empirical Evaluation of Feature Trace Recording on the Edit History of Marlin

Sören Viegener

```
+ #if m
+     /* inserted code */
+ #endif
```

edit to SPL

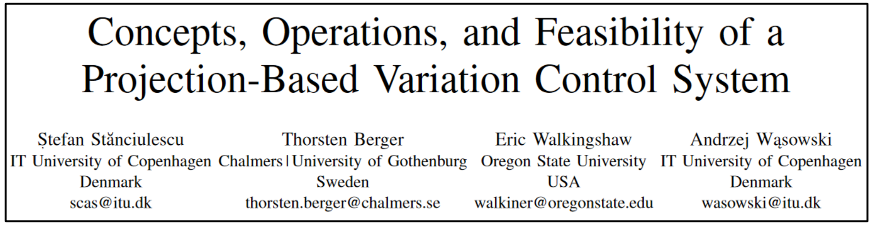
decompose

insert code into a variant implementing m (then merge)

edit to variants

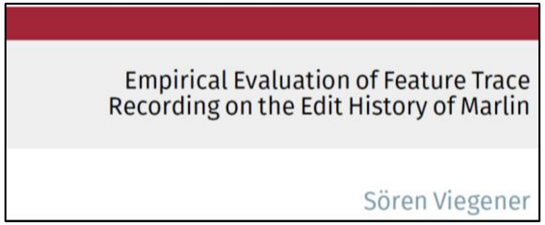
Software Engineering
Programming Languages

Can we reproduce edits to SPLs as edits to variants?



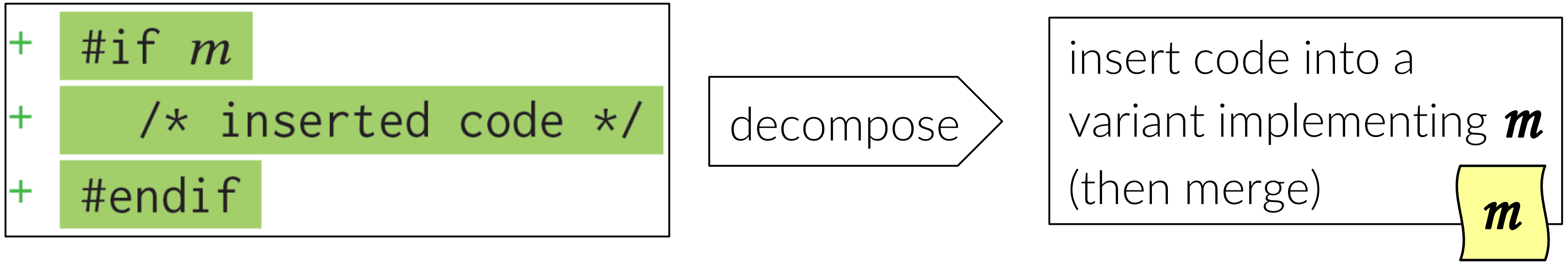
Concepts, Operations, and Feasibility of a Projection-Based Variation Control System

Ștefan Stănciulescu, IT University of Copenhagen, Denmark, scas@itu.dk
Thorsten Berger, Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se
Eric Walkingshaw, Oregon State University, USA, walkiner@oregonstate.edu
Andrzej Wąsowski, IT University of Copenhagen, Denmark, wasowski@itu.dk



Empirical Evaluation of Feature Trace Recording on the Edit History of Marlin

Sören Viegener



```
+ #if m
+    /* inserted code */
+ #endif
```

edit to SPL

edit to variants

Results

RQ1 – Can we reproduce all considered kinds of edits?

Yes

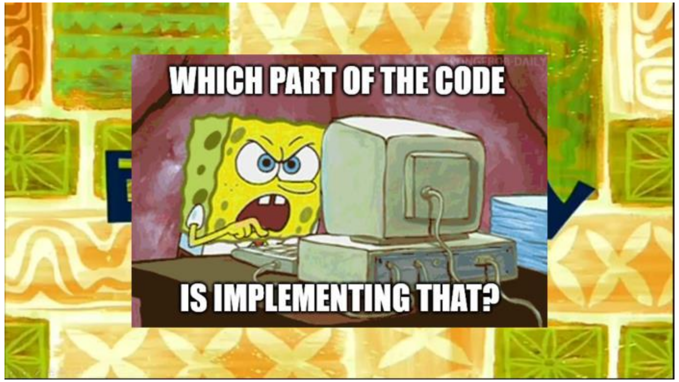
RQ2 – How many feature contexts are necessary?

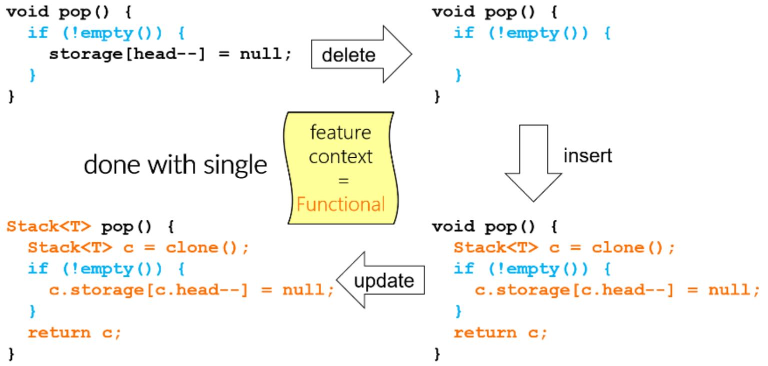
less or as many as when directly specifying mappings (worst case)

RQ3 – How complex are the feature contexts?

equal to target feature mapping (worst case)

Feature Trace Recording



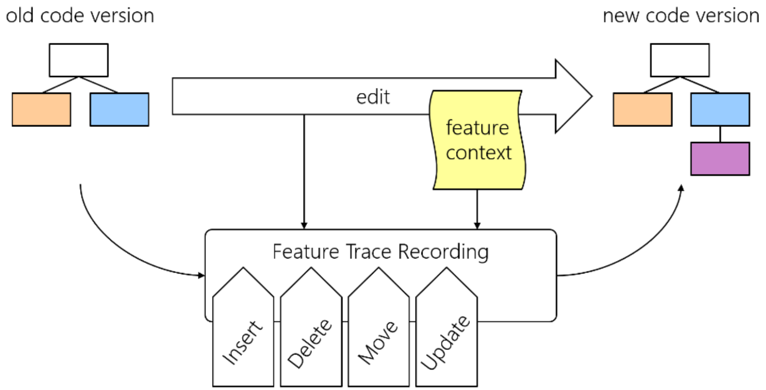


```
void pop() {
  if (!empty()) {
    storage[head--] = null;
  }
}
```

```
void pop() {
  if (!empty()) {

  }
}
```

```
void pop() {
  Stack<T> c = clone();
  if (!empty()) {
    c.storage[c.head--] = null;
  }
  return c;
}
```

```
Stack<T> pop() {
  Stack<T> c = clone();
  if (!empty()) {
    c.storage[c.head--] = null;
  }
  return c;
}
```



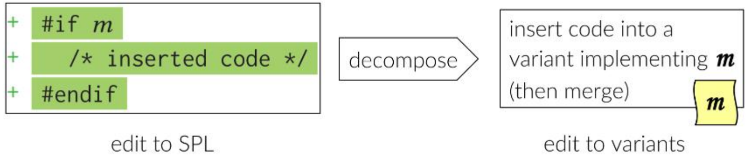
Can we reproduce edits to SPLs as edits to variants?

Concepts, Operations, and Feasibility of a Projection-Based Variation Control System

Ştefan Stănciulescu, IT University of Copenhagen, Denmark, scas@itu.dk
Thorsten Berger, Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se
Eric Walkingshaw, Oregon State University, USA, walkiner@oregonstate.edu
Andrzej Wąsowski, IT University of Copenhagen, Denmark, wasowski@itu.dk

Empirical Evaluation of Feature Trace Recording on the Edit History of Marlin

Sören Viegener



```
+ #if m
+   /* inserted code */
+ #endif
```

Software Engineering
Programming Languages