Simulator-based diff-time performance (regression) testing

Ivan Postolski, Victor Braberman, Diego Garbervetsky, Sebastián Uchitel

University of Buenos Aires
LaFHIS, ICC, CONICET, Argentina
ICSE 2019

Today's Performance Regression Testing

Developer <commits>

Source Code

<defines>

Performance Tests

QA Team

Today's Performance Regression Testing

Developer

Source Code

Performance Testing

<defines>

QA Team

Performance Tests

Today's Performance Regression Testing

Developer

- Time & Resource-consuming
- Complex environment
- High loads, Big inputs

Source Code

Performance Testing

<defines>

QA Team

Performance Tests

Today's Performance Regression Testing

Developer

Reported

Source Code

Performance Testing

Performance Regressions

<defines>

Tester

Performance Tests

Today's Performance Regression Testing

Developer

Reported

Source Code

Performance Testing

Performance Regressions

<defines>

Performance Tests

Tester

Goal

- Flag regressions earlier (right after a commit is introduced)

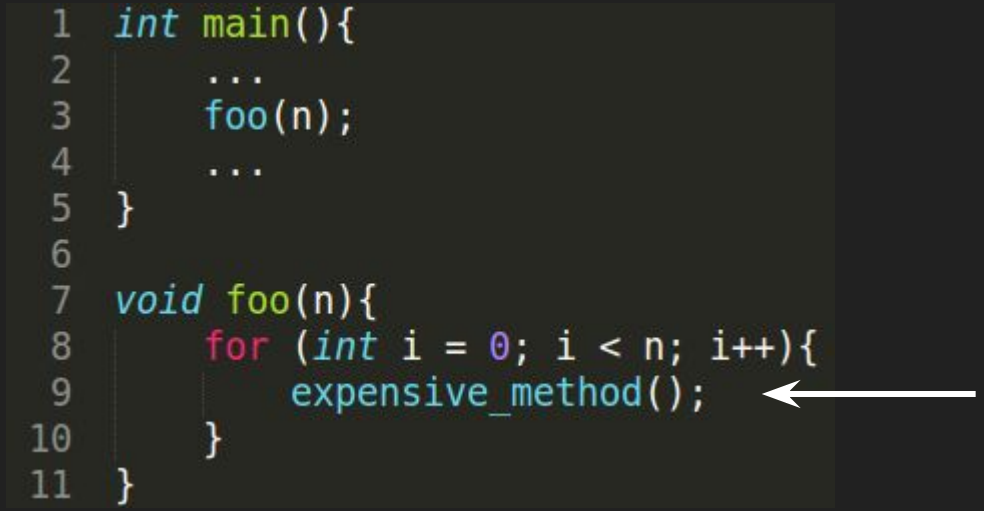
Goal

- Flag regressions earlier (right after a commit is introduced)
- In a more lightweight and faster fashion.

Focus on specific regressions

```
1  int main(){
2      ...
3      foo(n);
4      ...
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10     }
11 }
```

Focus on specific regressions



Focus on specific regressions

```
 1  int main(){
 2      ...
++ 3      foo(n*2);
 4      ...
 5  }
 6
 7  void foo(n){
 8      for (int i = 0; i < n; i++){
 9          expensive_method();
10      }
11  }
```

Focus on specific regressions

```
int main(){
++    if (cond()) n = n*n;
    foo(n);
    ...
}

void foo(n){
    for (int i = 0; i < n; i++){
        expensive_method();
    }
}
```

Focus on specific regressions

```
1  int main(){
2      ...
++ 3      for(...) foo(n);
4      ...
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10     }
11 }
```

P. Huang, X. Ma, D. Shen, and Y. Zhou, "*Performance regression testing target prioritization via performance risk analysis*," ICSE'14.

Our Approach

++

```
1  int main(){
2      ...
3      for(...) foo(n);
4      ...
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10     }
11 }
```

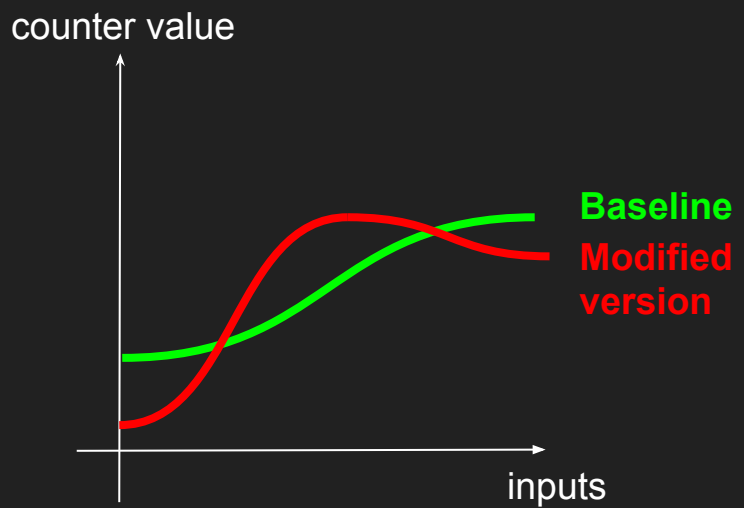
Our Approach

```
1  int main(){
2      ...
3      for(...) foo(n);
4      ...
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10         counter++;
11     }
12 }
```

++

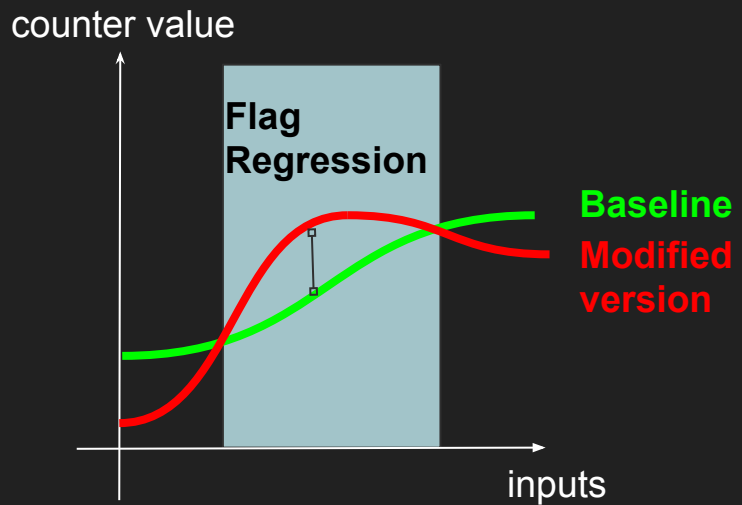
Our Approach

```
1  int main(){
2      ...
3      for(...) foo(n);
4      ...
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10         counter++;
11     }
12 }
```



Our Approach

```
1  int main(){
2      ...
++3    for(...) foo(n);
4      ...
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10         counter++;
11     }
12 }
```



Our Approach

```
1  int main(){
2      start_heavy_environment();
++3    for(...) foo(n);
4      other_heavy_operation();
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10         counter++;
11     }
12 }
```

Our Approach

```
1  int main(){
2      start_heavy_environment();
++3    for(...) foo(n);
4      other_heavy_operation();
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10         counter++;
11     }
12 }
```

Slice out statements and computations not related to counter value

Our Approach

```
1  int main(){
2      start_heavy_environment();
++ 3  for(...) foo(n);
4      other_heavy_operation();
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10         counter++;
11     }
12 }
```

Slice out statements and computations not related to counter value

Our Approach

```
1  int main(){
2      start_heavy_environment();
++ 3      for(...) foo(n);
4      other_heavy_operation();
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9          expensive_method();
10             counter++;
11     }
12 }
```

Slice out statements and computations not related to counter value

Our Approach

```
1  int main(){
2
++ 3      for(...) foo(n);
4
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9
10         counter++;
11     }
12 }
```

Slice out statements and computations not related to counter value

Our Approach

++

```
1  int main(){
2
3      for(...) foo(n);
4
5  }
6
7  void foo(n){
8      for (int i = 0; i < n; i++){
9
10         counter++;
11     }
12 }
```

Performance Simulator

Slice out statements and computations not related to counter value

Today's Performance Regression Testing

Developer

Reported

Source Code

Standard Performance Testing

Performance Regressions

<defines>

Performance Tests

Tester

Our approach

Developer <commits>

Source Code

Dynamic Slicing

<defines>

Tester

Performance Tests

Our approach

Developer <commits>

Source Code

Performance Simulator

Dynamic Slicing

<defines>

Performance Tests

Tester

Our approach

Developer

Performance Simulator

Faster Performance Testing

<defines>

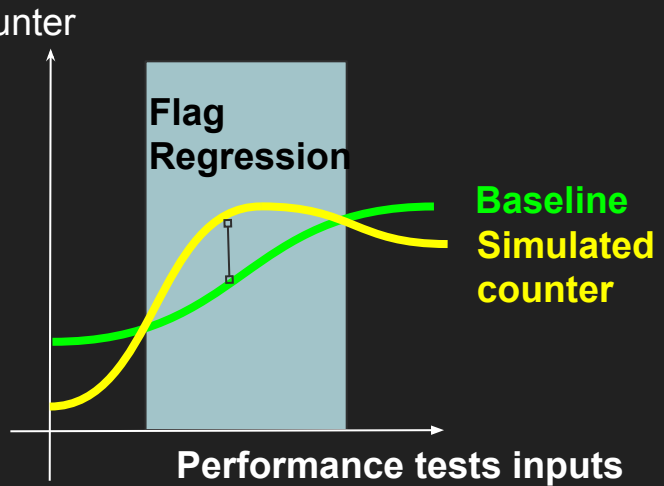
Tester

Performance Tests

Our approach

Developer

Performance Simulator



Faster Performance Testing

<defines>

Tester

Performance Tests

Our approach

Developer

Reported right away

Performance Simulator

Faster Performance Testing

Performance Regressions

<defines>

Tester

Performance Tests

Our approach

Okay!

Reported right away

Developer

Performance Simulator

Faster Performance Testing

Performance Regressions

<defines>

Tester

Performance Tests

What is the price?

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

What is the price?

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

Dynamic Slicing

What is the price?

true 5

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

Dynamic Slicing

What is the price?

true 5

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

Dynamic Slicing

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

What is the price?

true 5

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

Dynamic Slicing

true 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

What is the price?

true 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

100

Dynamic Slicing

==

true 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

100

What is the price?

true 9999

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

9999

Dynamic Slicing

==

true 9999

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

9999

What is the price?

false 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

Dynamic Slicing

false 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

What is the price?

false 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

1

Dynamic Slicing

false 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

0

≠

What is the price?

false 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

1

Dynamic Slicing

false 100

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

0

≠

What is the price?

false 9999

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

1

Dynamic Slicing

≠

false 9999

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

0

The bottomline

Performance Simulator

Made by

The bottomline

Performance Simulator

Made by

While fast ...

The bottomline

Performance Simulator

Made by

While fast ...

May or may not return the correct counter value

How can we tell when the simulator is correct?

Our idea

Source Code

```
int main(bool flag, int n) {

    if (flag){
        sleep(n);

        for(int i = 0; i < n; i++)
            counter++;

    } else {

        counter++;
    }

    return counter;
}
```

Performance Simulator

```
int main(bool flag, int n) {

    if (flag){

        for(int i = 0; i < n; i++)
            counter++;

    }



    return counter;
}
```

Our idea

Source Code

```
int main(bool flag, int n) {

    if (flag){
        sleep(n);

        for(int i = 0; i < n; i++)
            counter++;

    } else {

        counter++;
    }

    return counter;
}
```

Performance Simulator

```
int main(bool flag, int n) {

    if (flag){

        for(int i = 0; i < n; i++)
            counter++;

    }



    return counter;
}
```

Our idea

Source Code

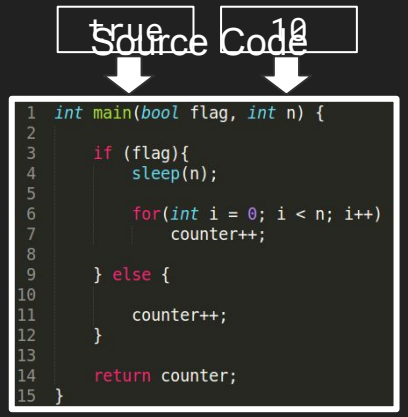
```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

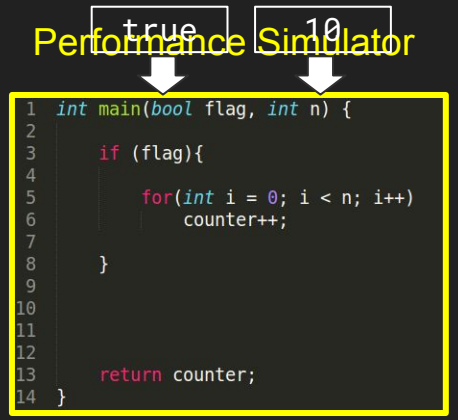
Performance Simulator

```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

Our idea

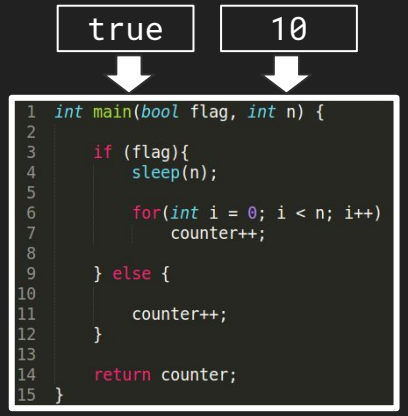

true
10
Certificate Program

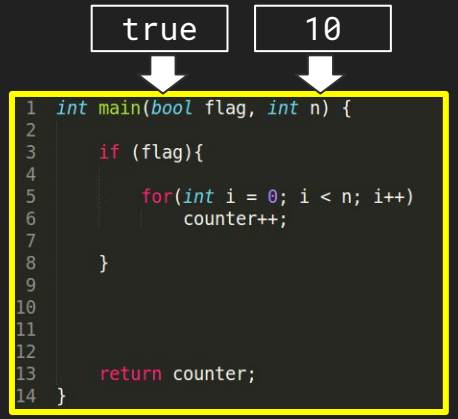

true
10
Source Code
1 int main(bool flag, int n) {
2
3 if (flag){
4 sleep(n);
5
6 for(int i = 0; i < n; i++)
7 counter++;
8
9 } else {
10
11 counter++;
12 }
13
14 return counter;
15 }


true
10
Performance Simulator
1 int main(bool flag, int n) {
2
3 if (flag){
4
5 for(int i = 0; i < n; i++)
6 counter++;
7
8 }
9
10
11
12
13 return counter;
14 }

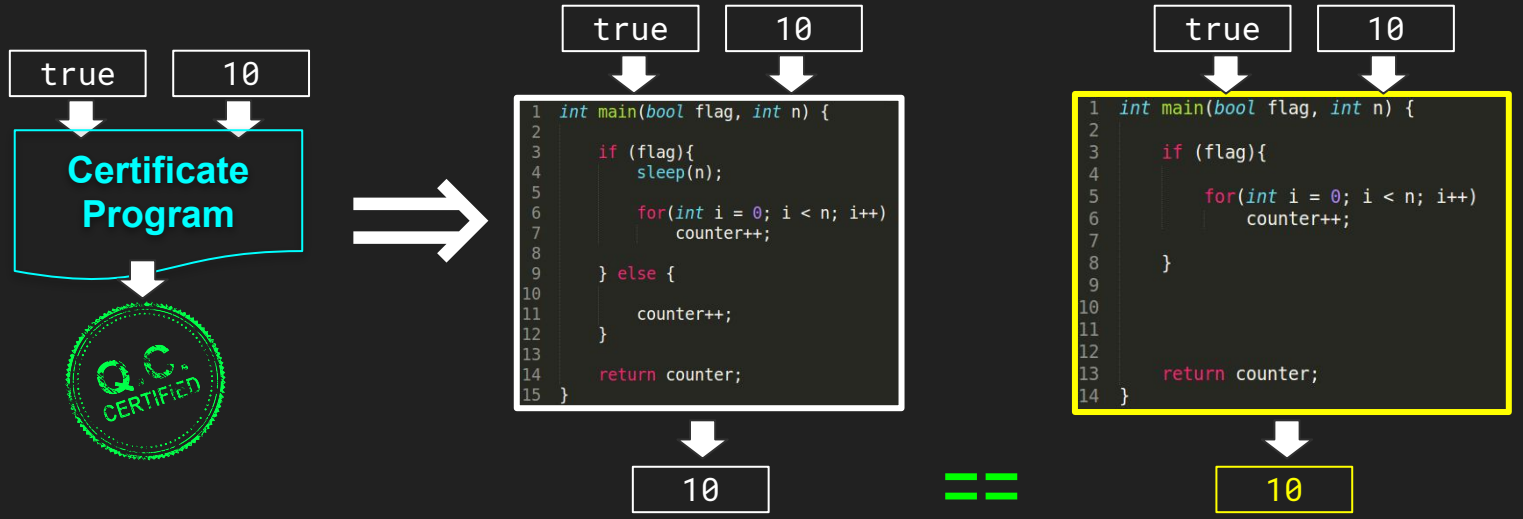
Our idea


true
10
Certificate Program
Q.C. CERTIFIED


true
10
int main(bool flag, int n) {
if (flag){
sleep(n);
for(int i = 0; i < n; i++)
counter++;
} else {
counter++;
}
return counter;
}

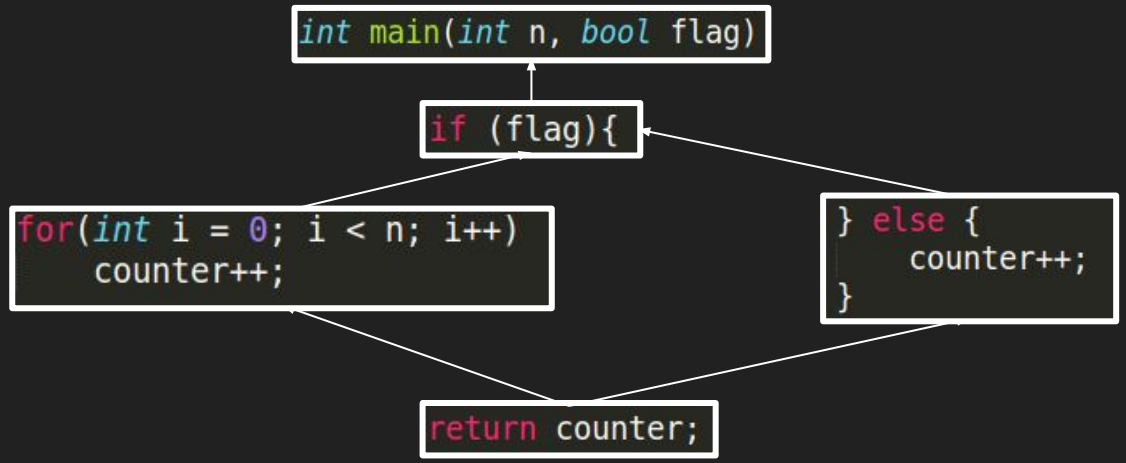

true
10
int main(bool flag, int n) {
if (flag){
for(int i = 0; i < n; i++)
counter++;
}
return counter;
}

Our idea


true
10
Certificate Program
Q.C. CERTIFIED
true
10
1 int main(bool flag, int n) {
2
3 if (flag){
4 sleep(n);
5
6 for(int i = 0; i < n; i++)
7 counter++;
8
9 } else {
10
11 counter++;
12 }
13
14 return counter;
15 }
10
==
true
10
1 int main(bool flag, int n) {
2
3 if (flag){
4
5 for(int i = 0; i < n; i++)
6 counter++;
7
8 }
9
10
11
12
13 return counter;
14 }
10

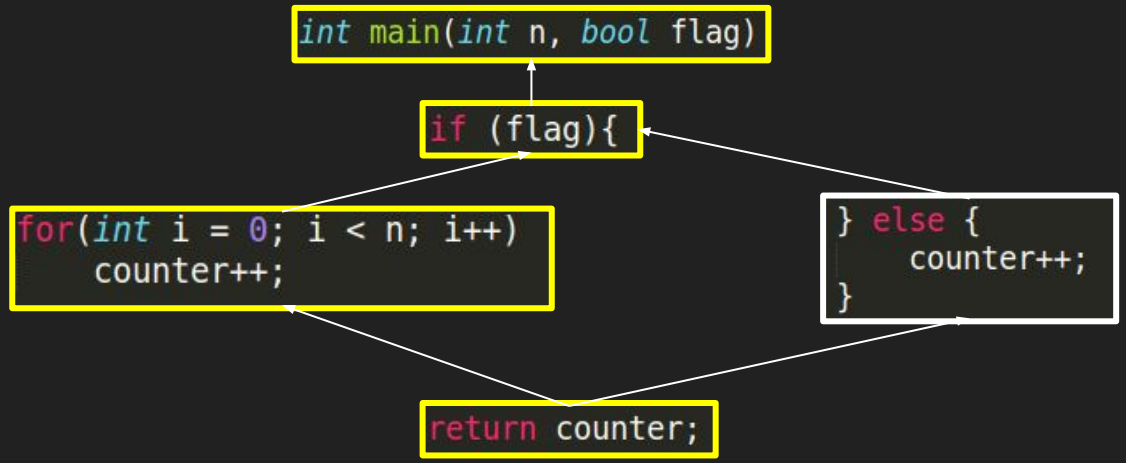
How to build a Certificate Program

Static Dependency Graph



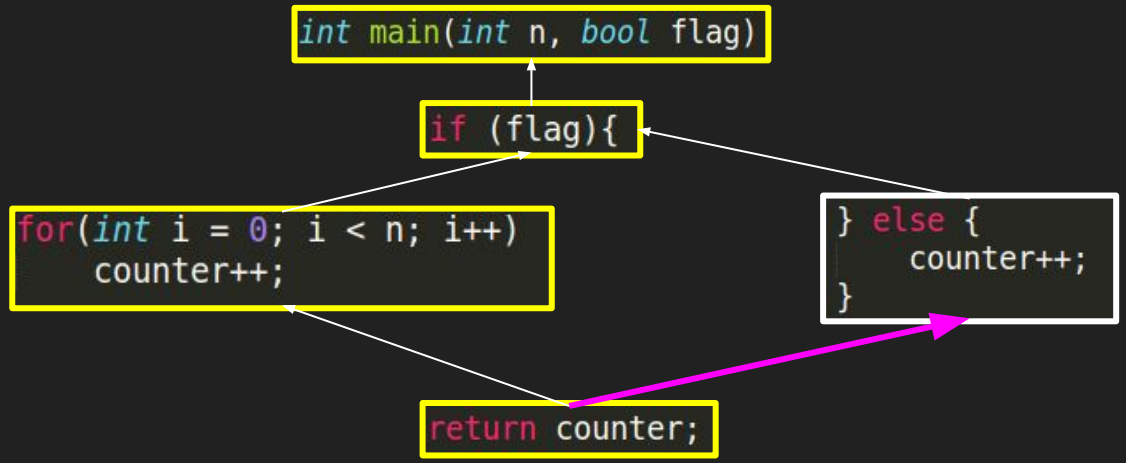
How to build a Certificate Program

Dynamic Slicing Statements



How to build a Certificate Program

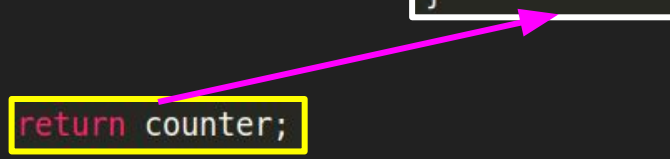
Find its **Frontiers**



How to build a Certificate Program

```
} else {
    counter++;
}
```

```
return counter;
```



```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

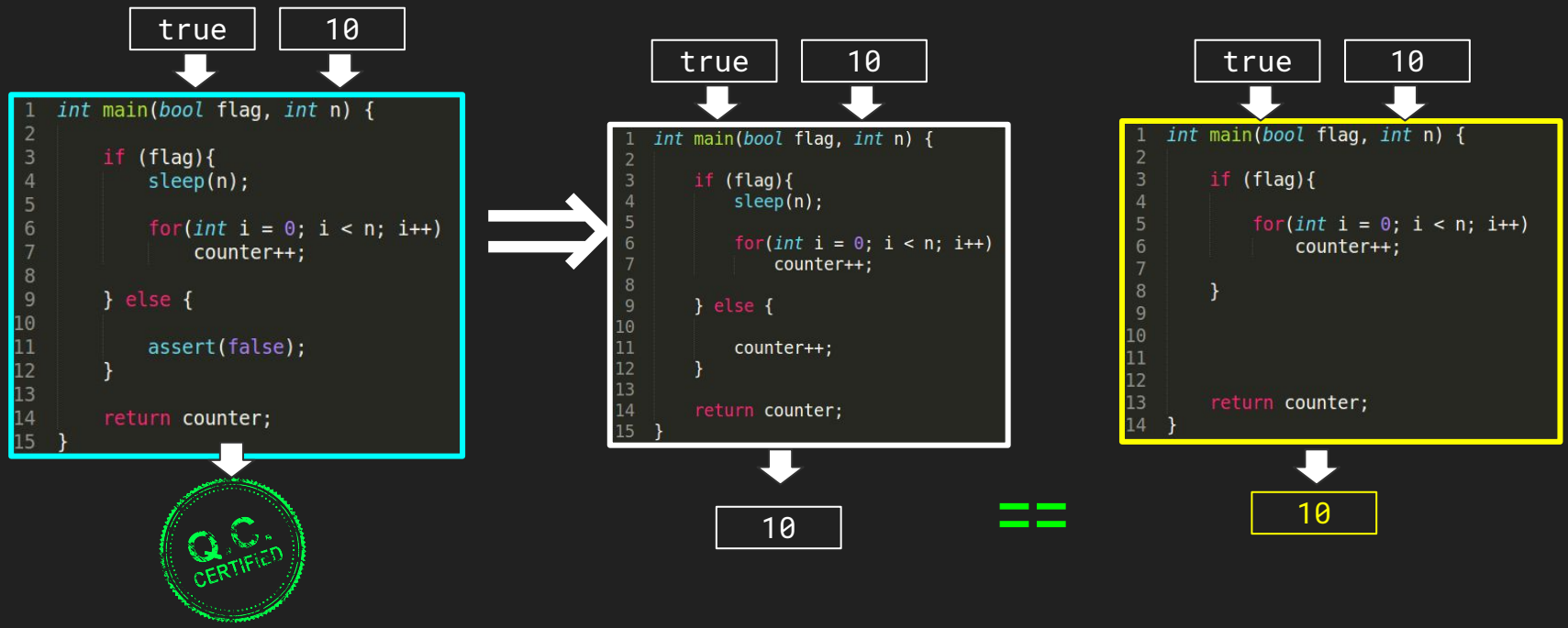
true

10

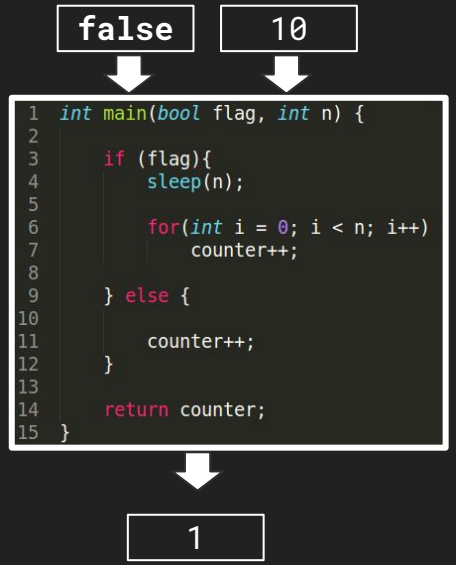
```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         assert(false);
12     }
13
14     return counter;
15 }
```


```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         counter++;
12     }
13
14     return counter;
15 }
```

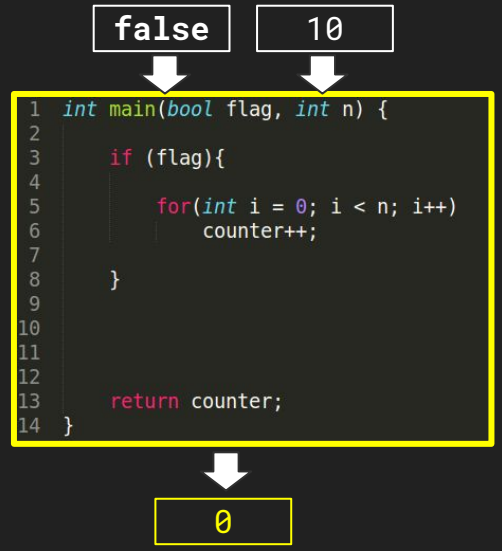
```
1  int main(bool flag, int n) {
2
3      if (flag){
4
5          for(int i = 0; i < n; i++)
6              counter++;
7
8      }
9
10
11
12
13     return counter;
14 }
```

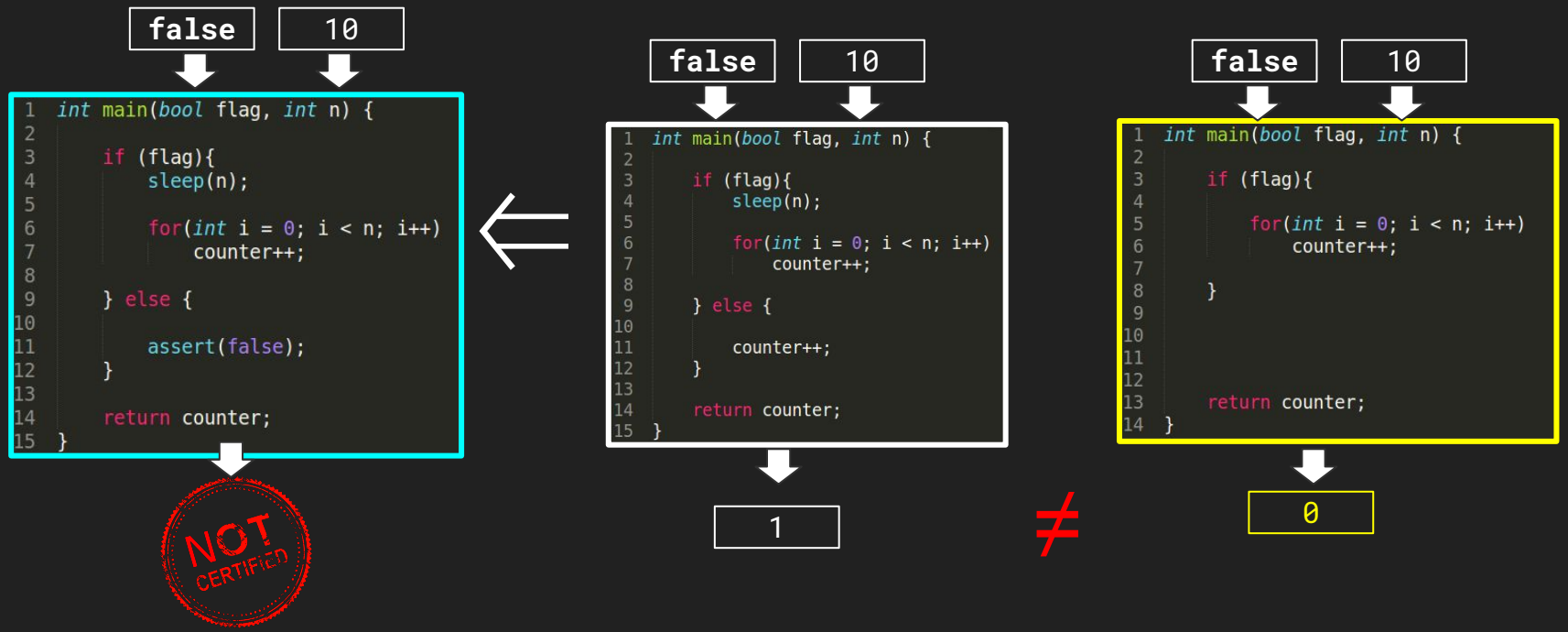

true
10
1 int main(bool flag, int n) {
2
3 if (flag){
4 sleep(n);
5
6 for(int i = 0; i < n; i++)
7 counter++;
8
9 } else {
10
11 assert(false);
12 }
13
14 return counter;
15 }
Q.C. CERTIFIED
true
10
1 int main(bool flag, int n) {
2
3 if (flag){
4 sleep(n);
5
6 for(int i = 0; i < n; i++)
7 counter++;
8
9 } else {
10
11 counter++;
12 }
13
14 return counter;
15 }
10
==
true
10
1 int main(bool flag, int n) {
2
3 if (flag){
4
5 for(int i = 0; i < n; i++)
6 counter++;
7
8 }
9
10
11
12
13 return counter;
14 }
10

```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         assert(false);
12     }
13
14     return counter;
15 }
```



≠




false
10
int main(bool flag, int n) {
if (flag){
sleep(n);
for(int i = 0; i < n; i++)
counter++;
} else {
assert(false);
}
return counter;
}
NOT CERTIFIED
false
10
int main(bool flag, int n) {
if (flag){
sleep(n);
for(int i = 0; i < n; i++)
counter++;
} else {
counter++;
}
return counter;
}
1
≠
false
10
int main(bool flag, int n) {
if (flag){
for(int i = 0; i < n; i++)
counter++;
}
return counter;
}
0

true

999999999

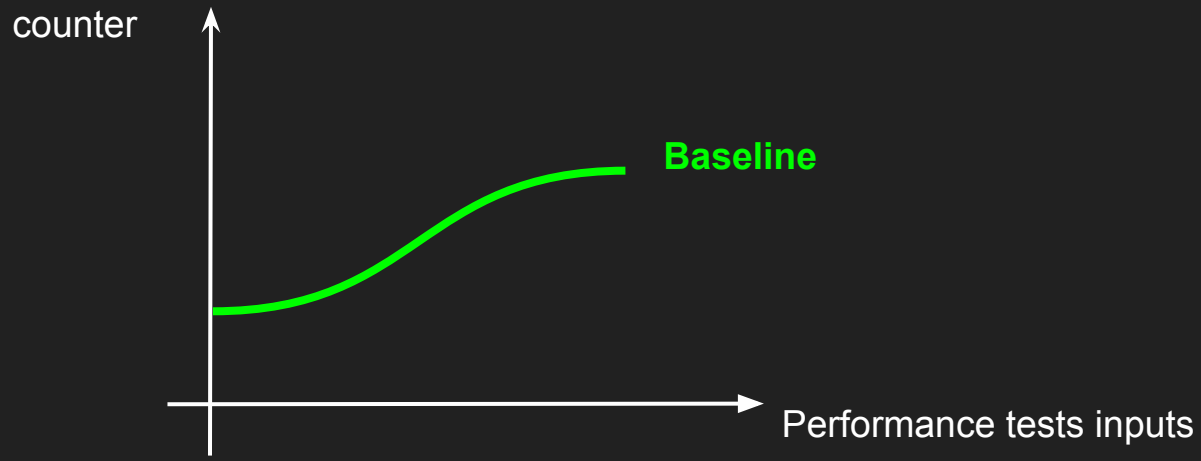
```
1  int main(bool flag, int n) {
2
3      if (flag){
4          sleep(n);
5
6          for(int i = 0; i < n; i++)
7              counter++;
8
9      } else {
10
11         assert(false);
12     }
13
14     return counter;
15 }
```

How to build a **faster** Certificate Program

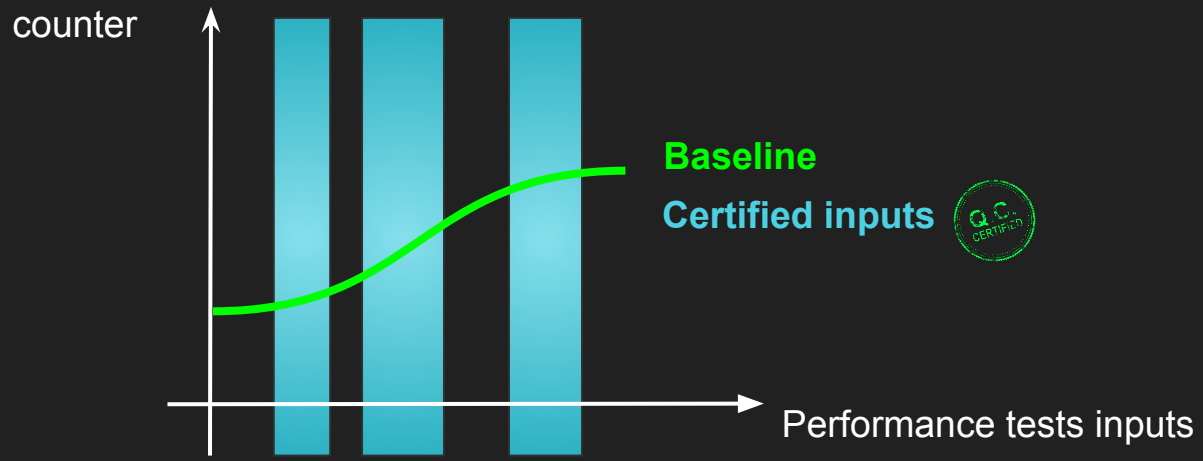
```
1  int main(bool flag, int n) {
2
3      //{if flag == true:
4      //     then the program will not fail}
5      if (flag) return;
6
7      if (flag){
8          sleep(n);
9          for(int i = 0; i < n; i++)
10             counter++;
11
12     } else {
13         assert(false);
14     }
15
16     return counter;
17 }
```

Accelerated by “Failure-directed program trimming,” K. Ferles, V. Wustholz, M. Christakis, and I. Dillig, in FSE’17.

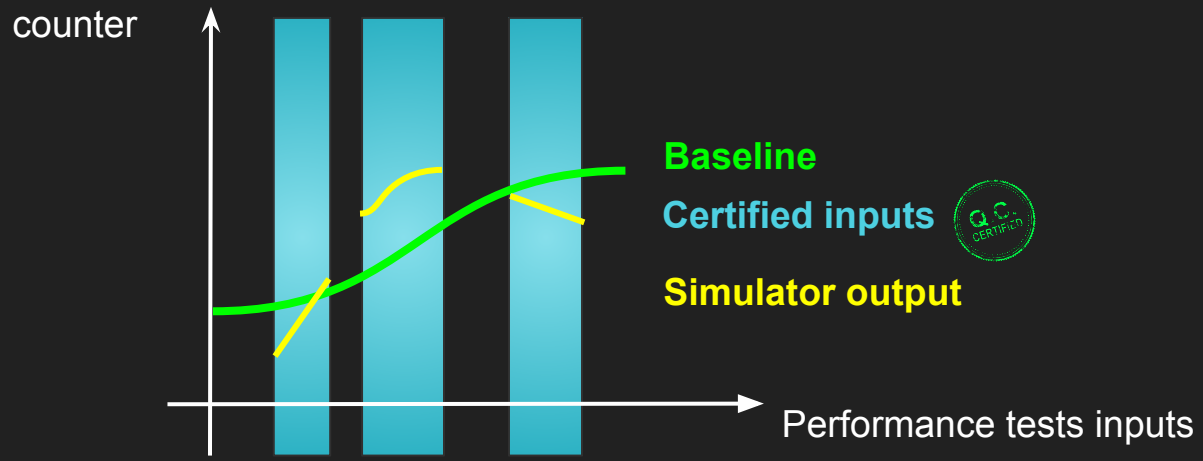
Performance testing with certificate



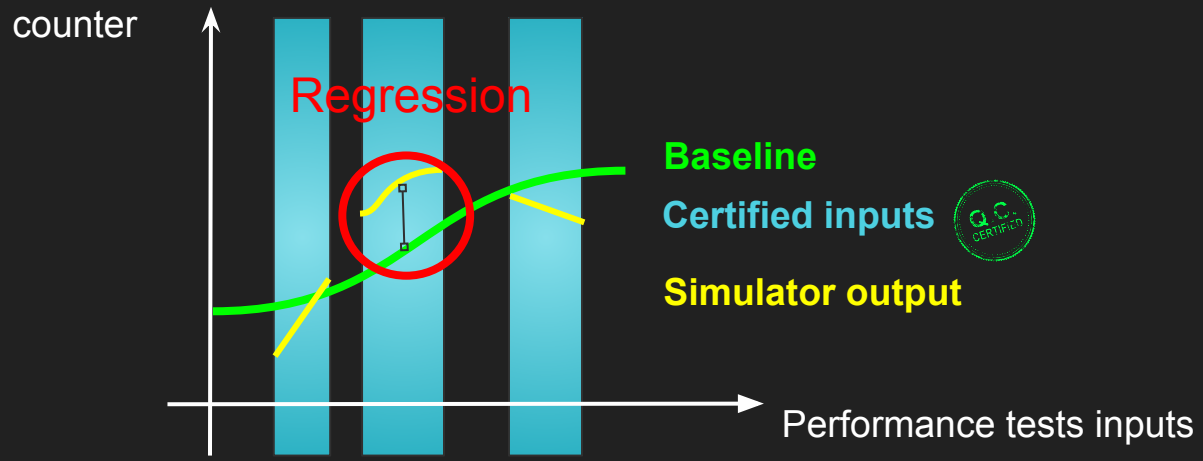
Performance testing with certificate



Performance testing with certificate



Performance testing with certificate



Proof of concept

Regressions → Simulator and Certificate → Results

MySQL (2.2 Mloc)
#46011

Olden BH (2 Kloc)
#artificial

Proof of concept

Regressions → Simulator and Certificate → Results

MySQL (2.2 Mloc)
#46011
operation= filesort()

Olden BH (2 Kloc)
#artificial
operation=logging()

Proof of concept

Regressions	Simulator and Certificate	Results
MySQL (2.2 Mloc) #46011 operation= filesort()	Sliced 3 lines, no frontiers were found statically. All perf. inputs were certified.	

Olden BH (2 Kloc)
#artificial
operation=logging()		

Proof of concept

Regressions	Simulator and Certificate	Results
MySQL (2.2 Mloc) #46011 operation= filesort()	Sliced 3 lines, no frontiers were found statically. All perf. inputs were certified.	

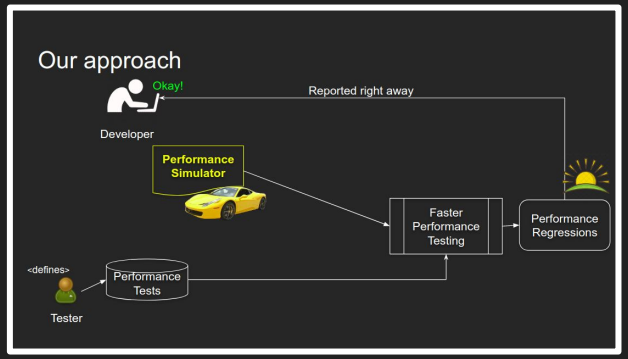
Olden BH (2 Kloc)
#artificial
operation=logging()
	Sliced 90% of the program, several frontiers were found.
	50% perf. inputs certified.	

Proof of concept

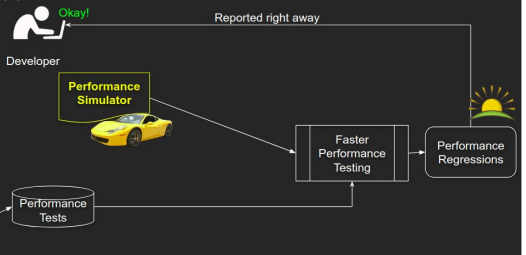
Regressions	Simulator and Certificate	Results
MySQL (2.2 Mloc) #46011 operation= filesort()	Sliced 3 lines, no frontiers were found statically. All perf. inputs were certified.	Flagged, ~10x Speedup Perf. tests / (SB+CB+C+S) 15m → 1.5m

Olden BH (2 Kloc)
#artificial
operation=logging()
	Sliced 90% of the program, several frontiers were found.
	50% perf. inputs certified.
		Flagged, ~20x Speedup
		Perf. tests / (SB+CB+C+S)
		19m → 1m

Our approach



Our approach



The bottomline

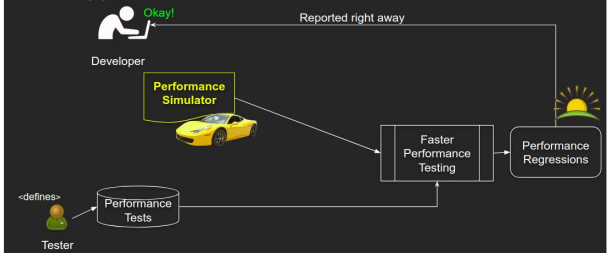
Performance Simulator

Made by

While fast ...

May or may not return the correct counter value

Our approach



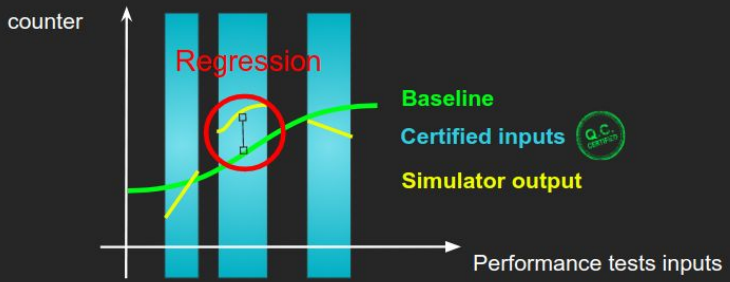
The bottomline

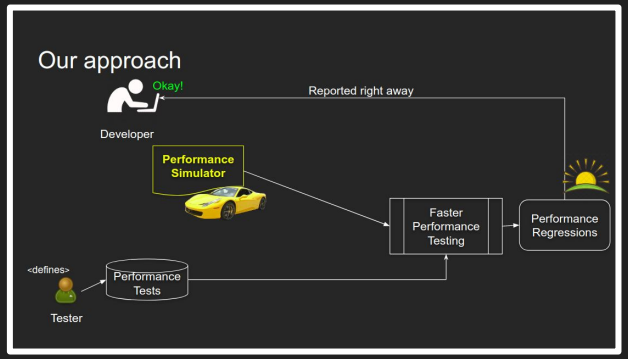
Performance Simulator

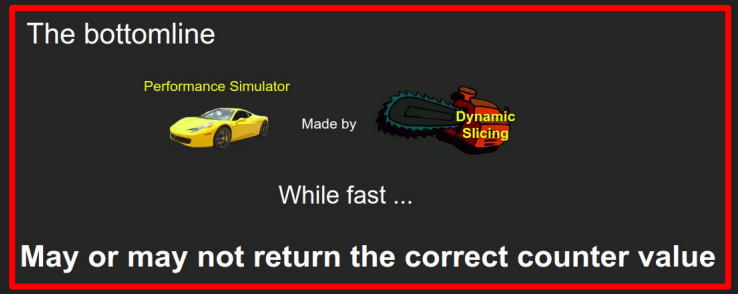
Made by

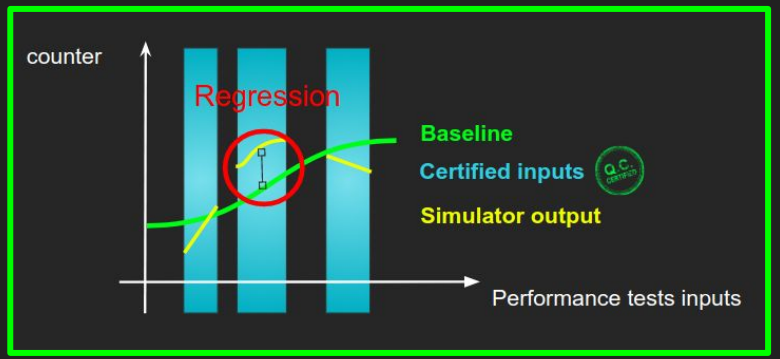
While fast ...

May or may not return the correct counter value







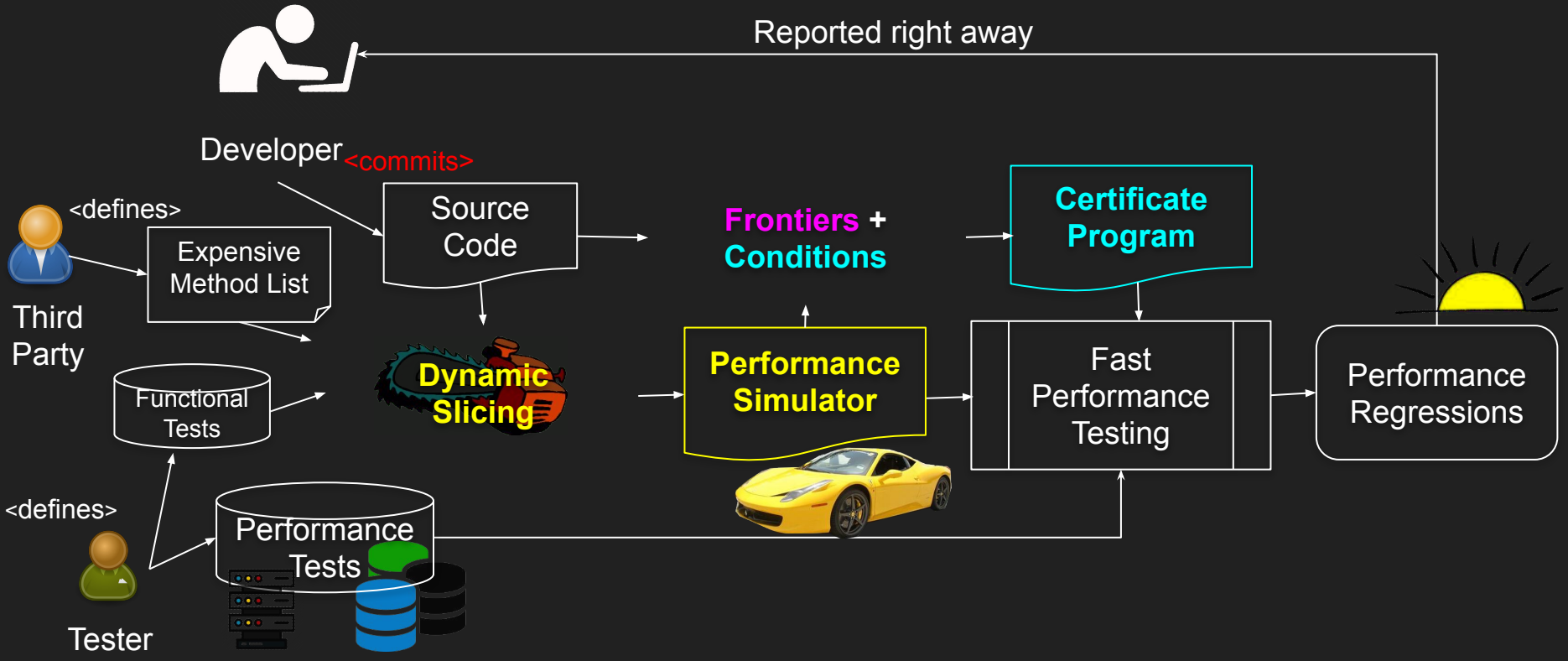


Pre-print & Offline-Questions

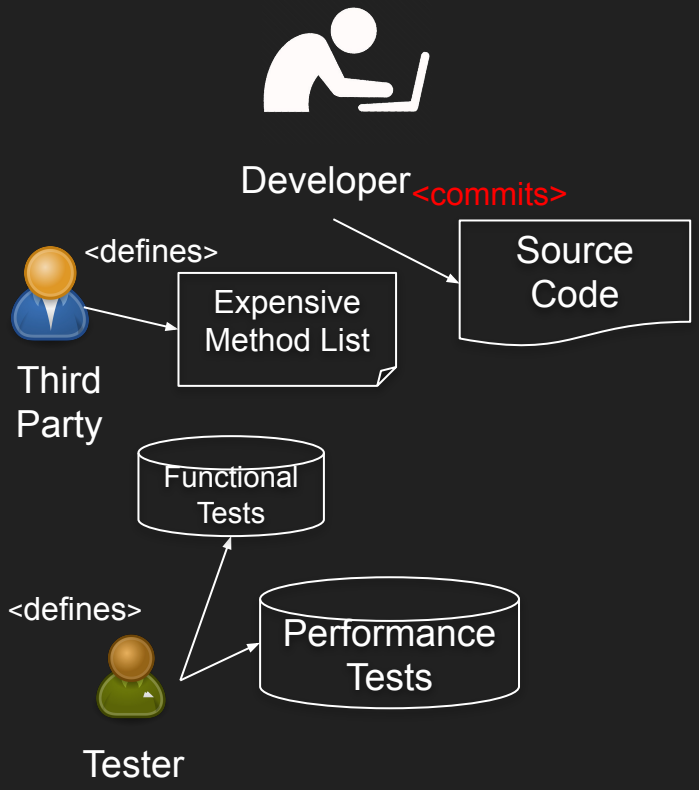
Appendix. Emerging Results (Full table)

Application	Simulator Building (a)	Certificate Building (b)	Simulation Execution (Avg.) (c)	Certificate Execution (Avg.) (d)	Standard Perf. Test Execution (Avg.) (e)	Variable Gain Factor $(\frac{e}{c+d})$	Total Gain Factor $(\frac{e}{a+b+c+d})$
Olden BH	25s	1.2s	0.07s	33.8s	18m 45s	33.21x	18.72x
MySQL-5.1.73	41.4s	55s	1.15s	0s	15m 42s	819.13x	9.65x

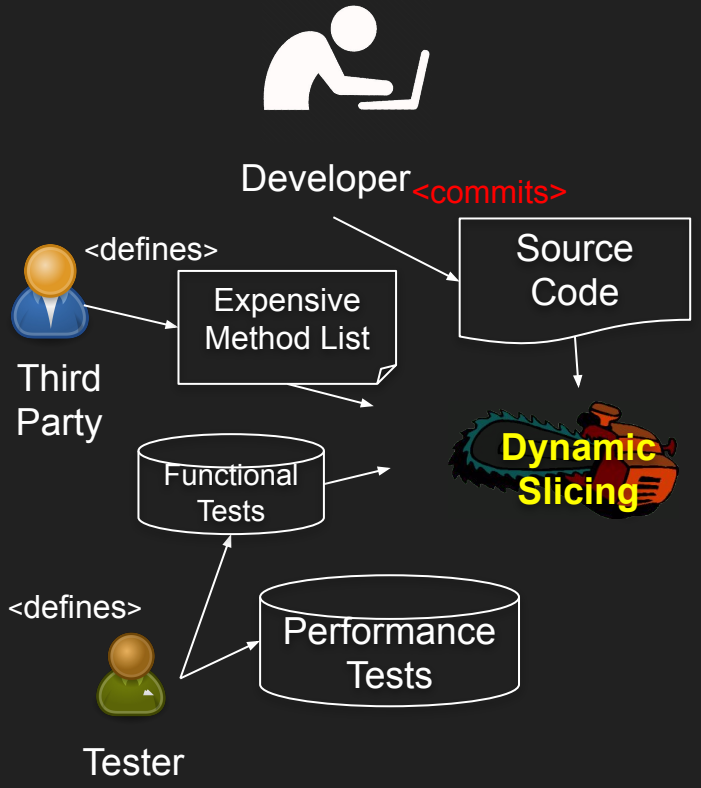
Our approach



Our approach



Our approach



Our approach

Developer <commits>

<defines>

Expensive Method List

Third Party

Source Code

Functional Tests

Dynamic Slicing

Performance Simulator

<defines>

Performance Tests

Tester

Our approach

Developer <commits>

Source Code

<defines>

Expensive Method List

Third Party

Functional Tests

Dynamic Slicing

Performance Simulator

Faster Performance Testing

<defines>

Performance Tests

Tester

Our approach

Developer <commits>

<defines>

Expensive Method List

Source Code

Third Party

Functional Tests

Dynamic Slicing

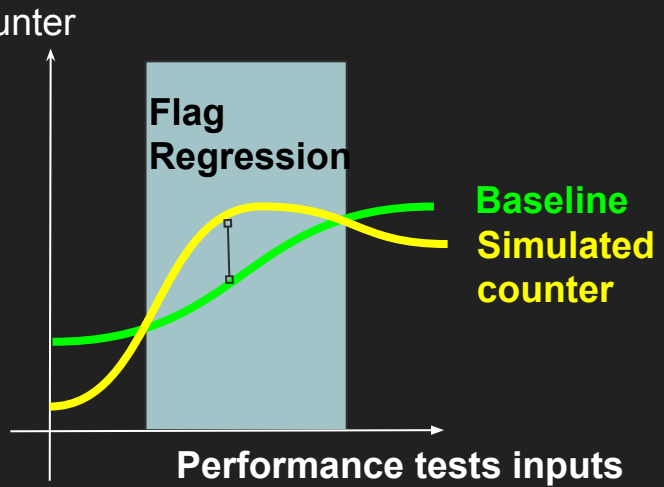
Performance Simulator

Faster Performance Testing

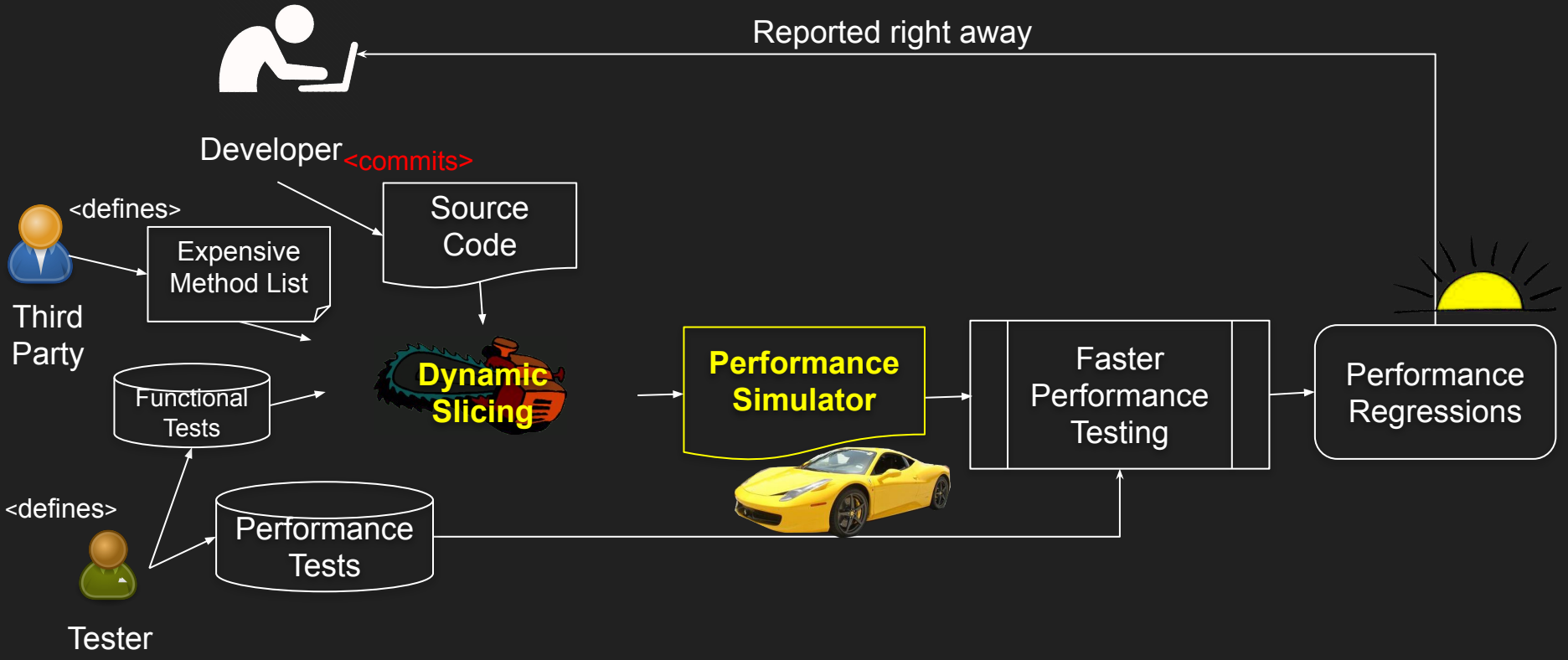
<defines>

Performance Tests

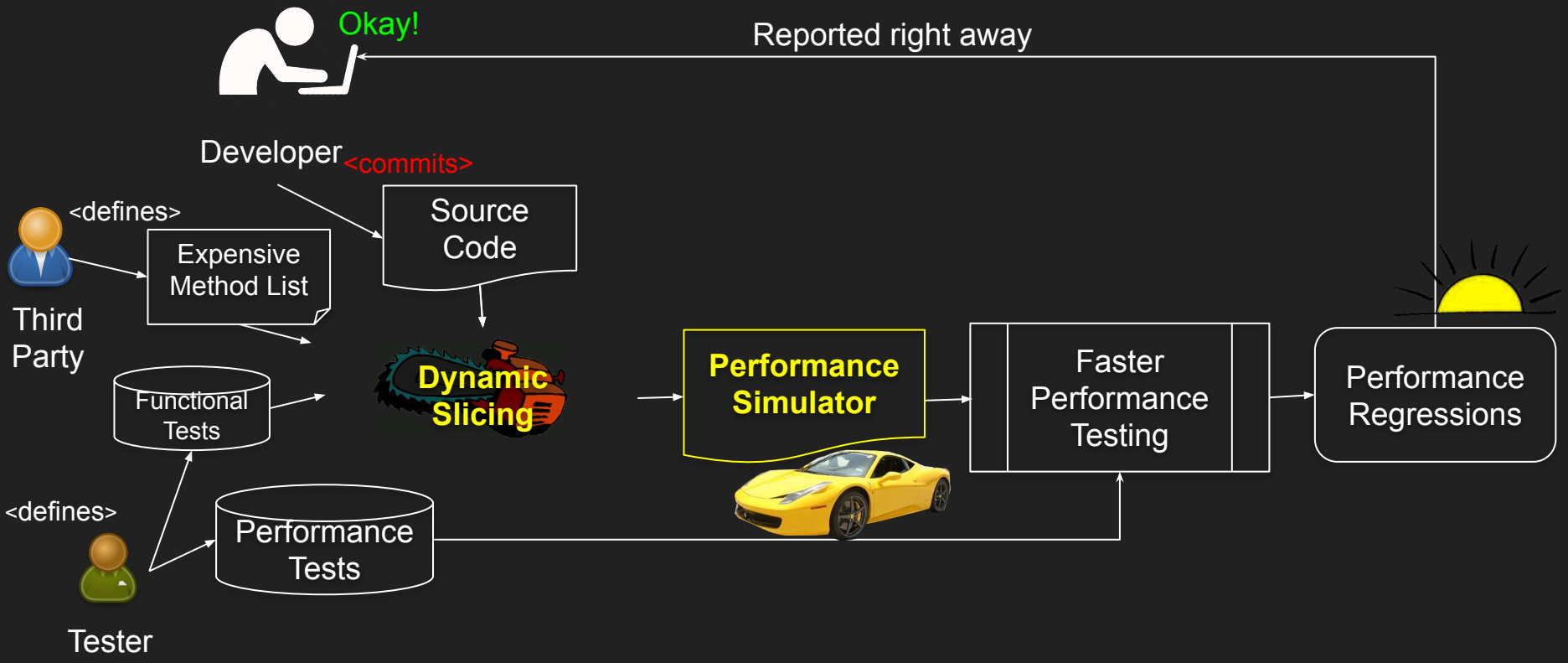
Tester



Our approach



Our approach



Our approach