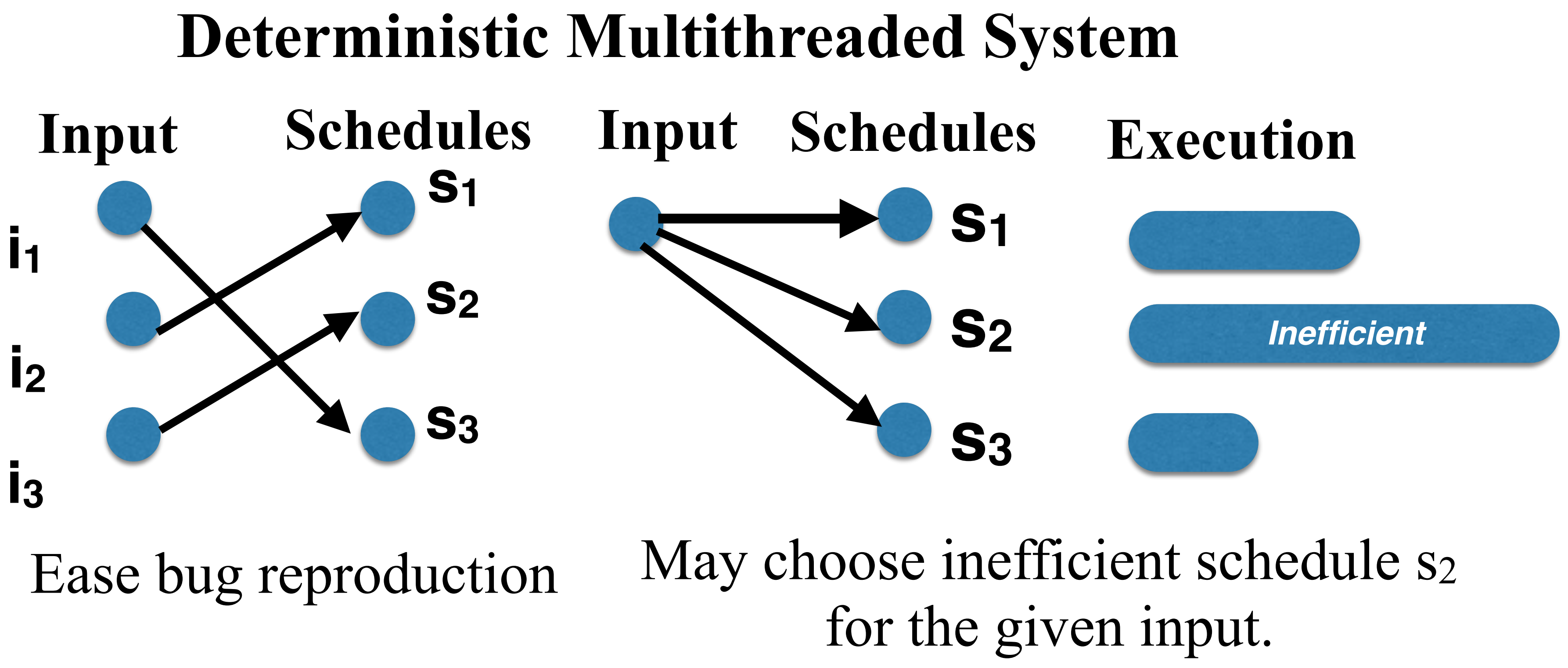
Techniques to Improve the Reliability of Software Applications

Monika Dhok
Advisor : Murali Krishna Ramanathan
Department of Computer Science and Automation

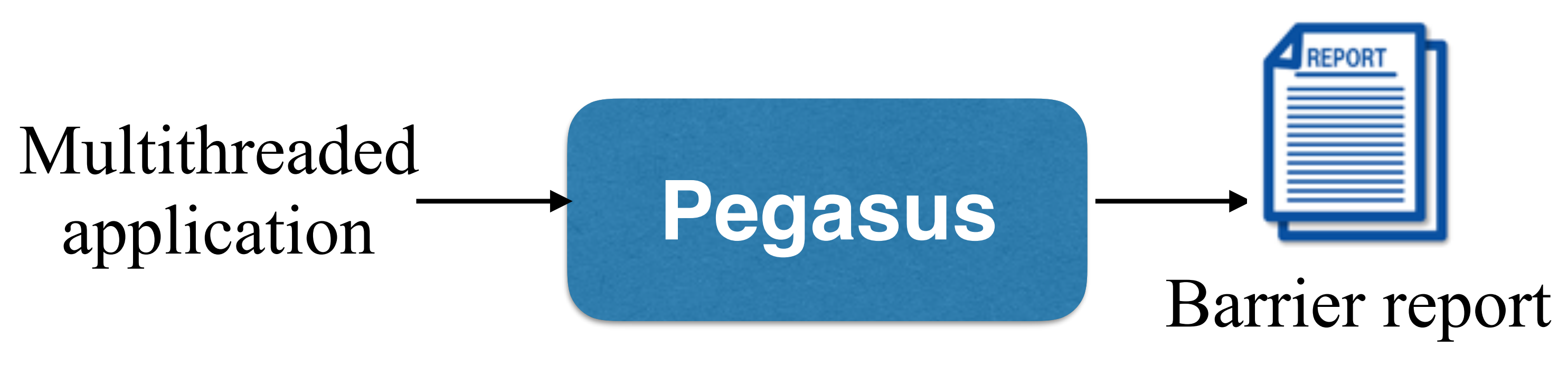
Developed dynamic analysis techniques to address following research problems in the domain of performance analysis, automated test generation, and their intersection.

Automated Barrier Inference for Performance Improvement [ISSTA'15]

joint work with R. Mudduluru, Murali Krishna Ramanathan



How to force deterministic runtime to use most efficient schedule, s_3?

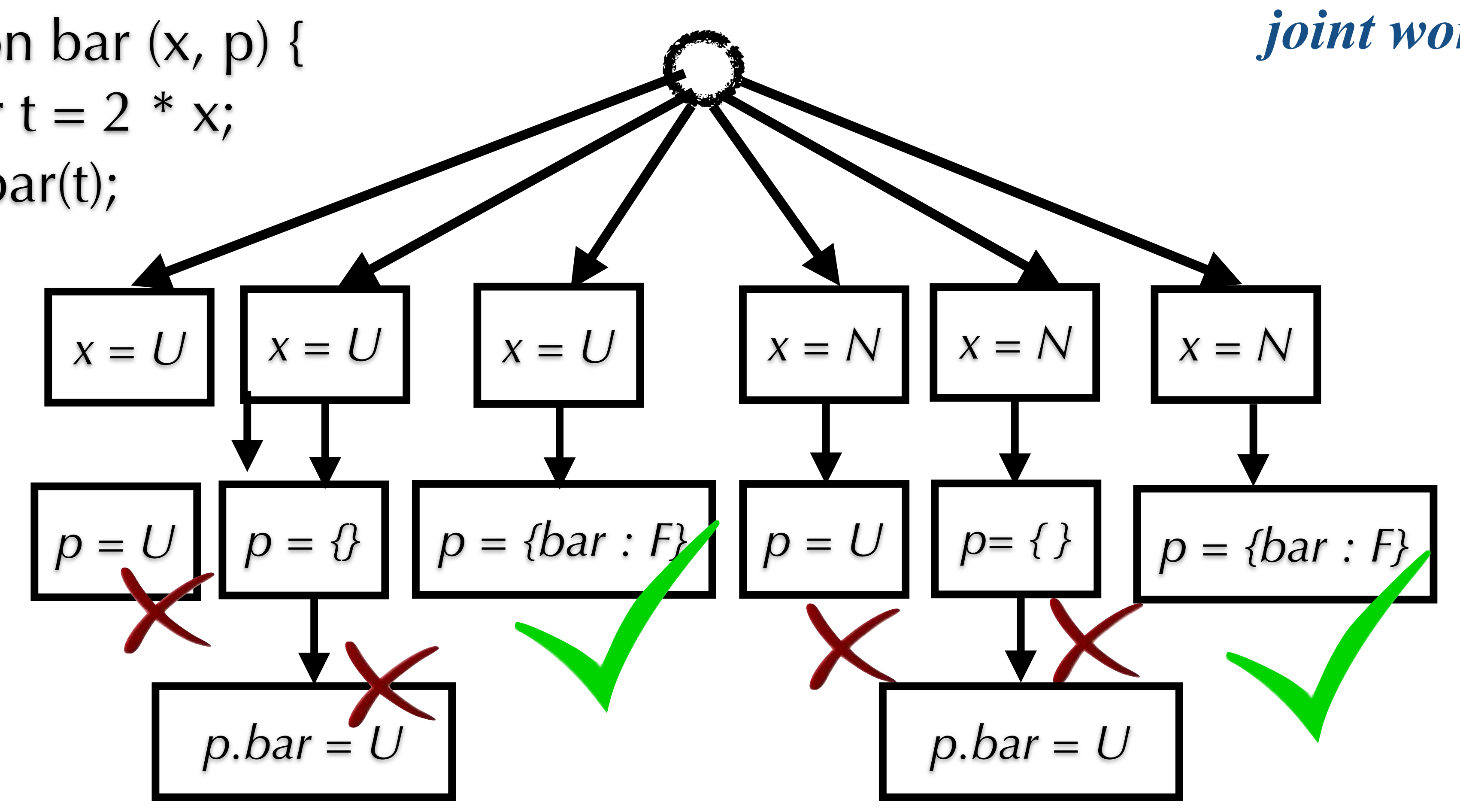


Proposed an approach to insert barriers in the program for performance improvement. Observed improvement ranging from 38-88%.

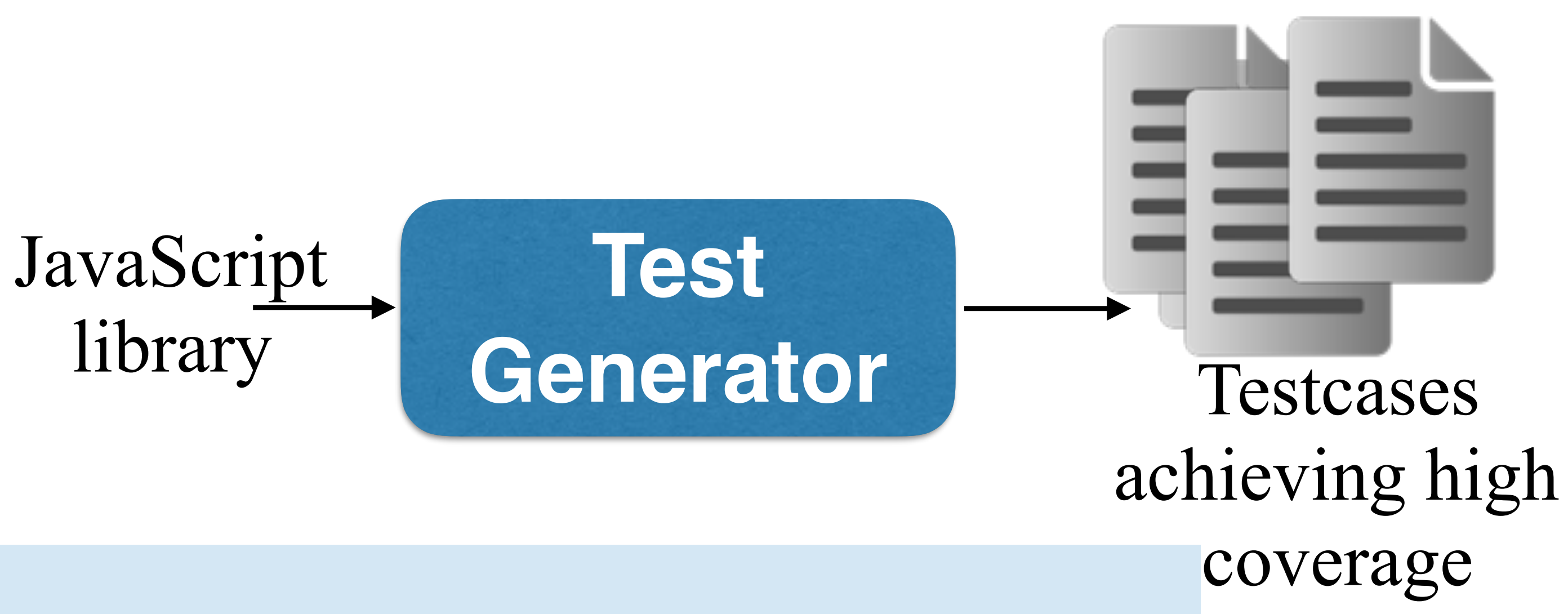
Type-aware Concolic Testing of JavaScript programs [ICSE'16]

joint work with Murali Krishna Ramanathan, Nishant Sinha

```
L1 : function bar (x, p) {
L2 :       var t = 2 * x;
L3 :       p.bar(t);
L4 : }
```



Do we need to explore all these paths to achieve high coverage?



Type-aware concolic testing reduced the generation of redundant inputs. Generated less than 5% of the inputs as compared to conventional approaches.

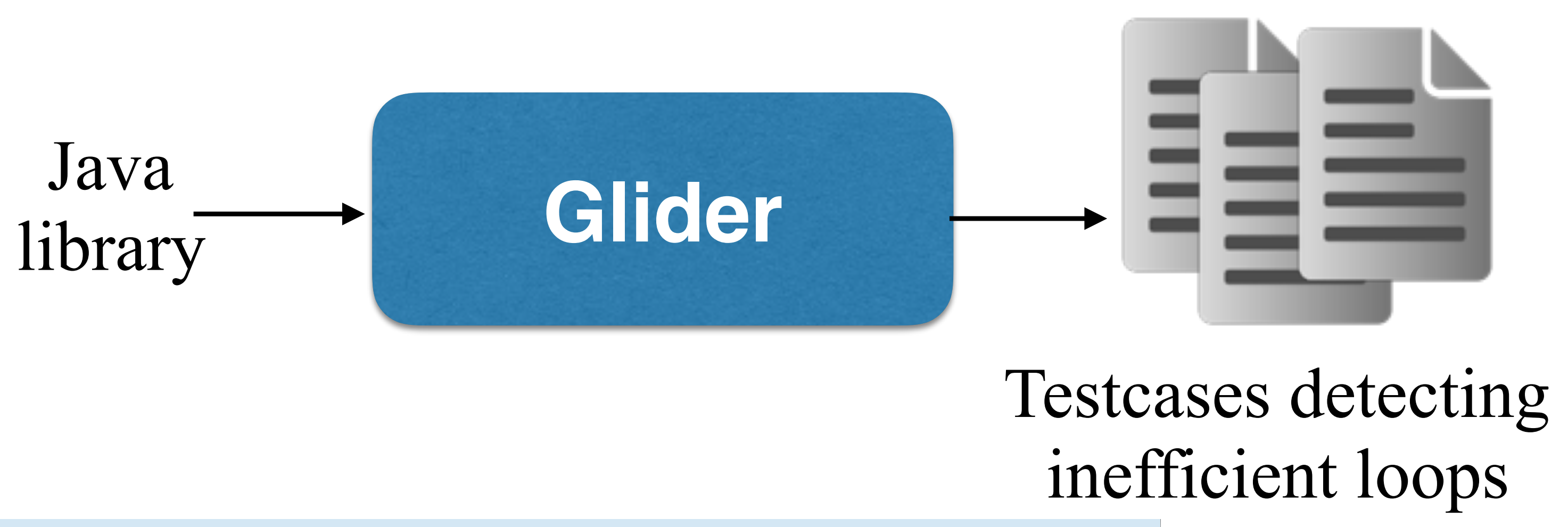
Directed Test Generation to Detect Loop Inefficiencies [FSE'16]

joint work with Murali Krishna Ramanathan

Redundant traversal bugs : program iterates over a data structure repeatedly without any intermediate changes

```
boolean foo(Collection c1, Collection c2){
  for(Object e : c1)
    if(c2.contains(e))
      return true
 return false
}
```

How to expose such performance bugs, and prove their presence?



Proposed an approach to generate tests detecting loop inefficiencies. Detected 46 bugs across 7 Java libraries including 34 previously unknown.

Techniques to improve the reliability of software applications

Monika Dhok
Advisor : Murali Krishna Ramanathan

Program Analysis

Program Analysis

Studying behaviours of the computer program regarding certain property.

Program Analysis

Studying behaviours of the computer program regarding certain property.

Problems investigated

Program Analysis

Studying behaviours of the computer program regarding certain property.

Problems investigated

Automated Barrier Inference for Performance Improvement [ISSTA'15]

Proposed an approach to automatically insert barriers in the program for performance improvement. Observed improvement ranging from 38-88%

Program Analysis

Studying behaviours of the computer program regarding certain property.

Problems investigated

Automated Barrier Inference for Performance Improvement [ISSTA'15]

Proposed an approach to automatically insert barriers in the program for performance improvement. Observed improvement ranging from 38-88%

Type-aware Concolic Testing of JavaScript programs [ICSE'16]

Type-aware concolic testing reduced the generation of redundant inputs. Generated upto 5% inputs as compared to the conventional approaches.

Program Analysis

Studying behaviours of the computer program regarding certain property.

Problems investigated

Automated Barrier Inference for Performance Improvement [ISSTA'15]

Proposed an approach to automatically insert barriers in the program for performance improvement. Observed improvement ranging from 38-88%

Type-aware Concolic Testing of JavaScript programs [ICSE'16]

Type-aware concolic testing reduced the generation of redundant inputs. Generated upto 5% inputs as compared to the conventional approaches.

Directed Test Generation to Detect Loop Inefficiencies [FSE'16]

Proposed an approach to generate tests detecting loop inefficiencies. Detected 46 bugs across 7 Java libraries including 34 previously unknown.

Software efficiency is very important

Software efficiency is very important

- Performance issues are hard to detect during testing

Software efficiency is very important

- Performance issues are hard to detect during testing
- These issues are found even in well tested commercial softwares

Software efficiency is very important

- Performance issues are hard to detect during testing
- These issues are found even in well tested commercial softwares
- Degrade application responsiveness and user experience

Performance bugs are critical

Performance bugs are critical

- Implementation mistakes that cause inefficiency

Performance bugs are critical

- Implementation mistakes that cause inefficiency
- Difficult to catch them during compiler optimizations

Performance bugs are critical

- Implementation mistakes that cause inefficiency
- Difficult to catch them during compiler optimizations
- Fixing them can result in large speedups, thereby improving efficiency

Redundant traversal bugs

When program iterates over a data structure repeatedly without any intermediate modifications

Redundant traversal bugs

When program iterates over a data structure repeatedly without any intermediate modifications

```
boolean foo(Collection c1, Collection c2){
  for(Object e : c1)
     if(c2.contains(e))
        return true
 return false
}
```

Redundant traversal bugs

When program iterates over a data structure repeatedly without any intermediate modifications

```
boolean foo(Collection c1, Collection c2){
  for(Object e : c1)
     if(c2.contains(e))
        return true
 return false
}
```

Complexity :
O(size(c1) * size(c2))

Redundant traversal bugs

When program iterates over a data structure repeatedly without any intermediate modifications

```
boolean foo(Collection c1, Collection c2){
  for(Object e : c1)
    if(c2.contains(e))
       return true
  return false
}
```

Complexity :
O(size(c1) * size(c2))

```
boolean foo(Collection c1, Collection c2){
  HashSet c3 = new HashSet(c2)
  for(Object e : c1)
    if(c3.contains(e))
       return true
  return false
}
```

Redundant traversal bugs

When program iterates over a data structure repeatedly without any intermediate modifications

```
boolean foo(Collection c1, Collection c2){
  for(Object e : c1)
    if(c2.contains(e))
      return true
  return false
}
```

Complexity :
$O(size(c1) * size(c2))$

```
boolean foo(Collection c1, Collection c2){
  HashSet c3 = new HashSet(c2)
  for(Object e : c1)
    if(c3.contains(e))
      return true
  return false
}
```

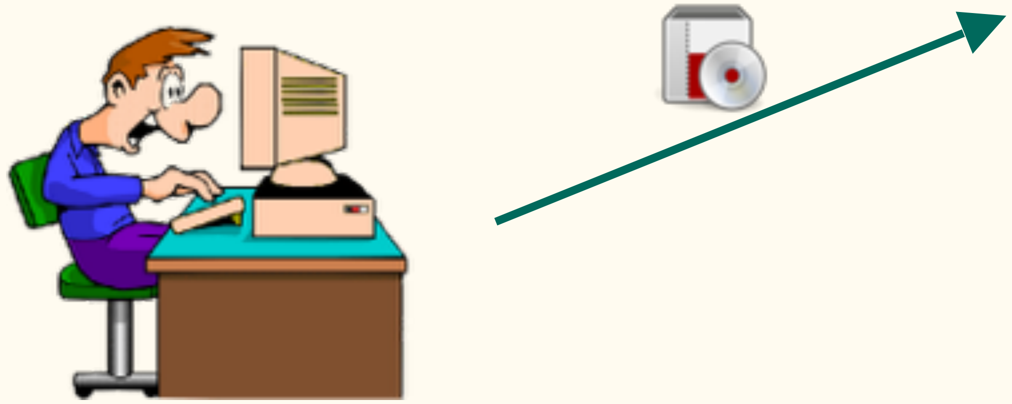
Complexity :
$O(size(c1)) + \epsilon$

Performance tests are written by developers

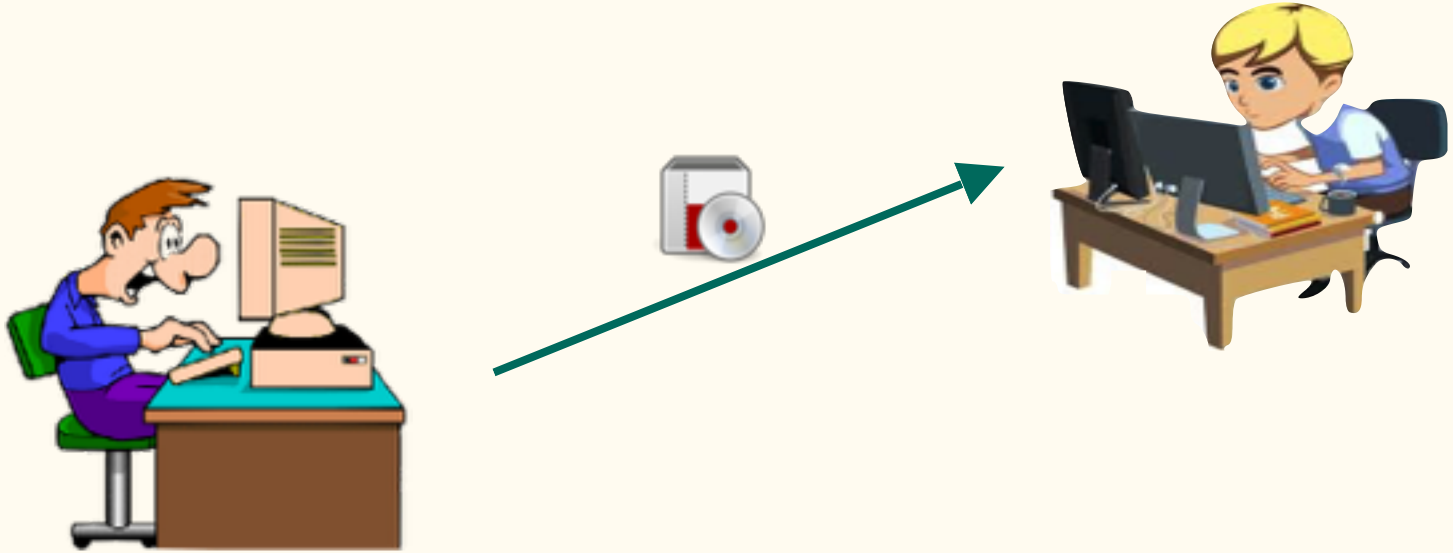
Performance tests are written by developers

Performance tests are written by developers

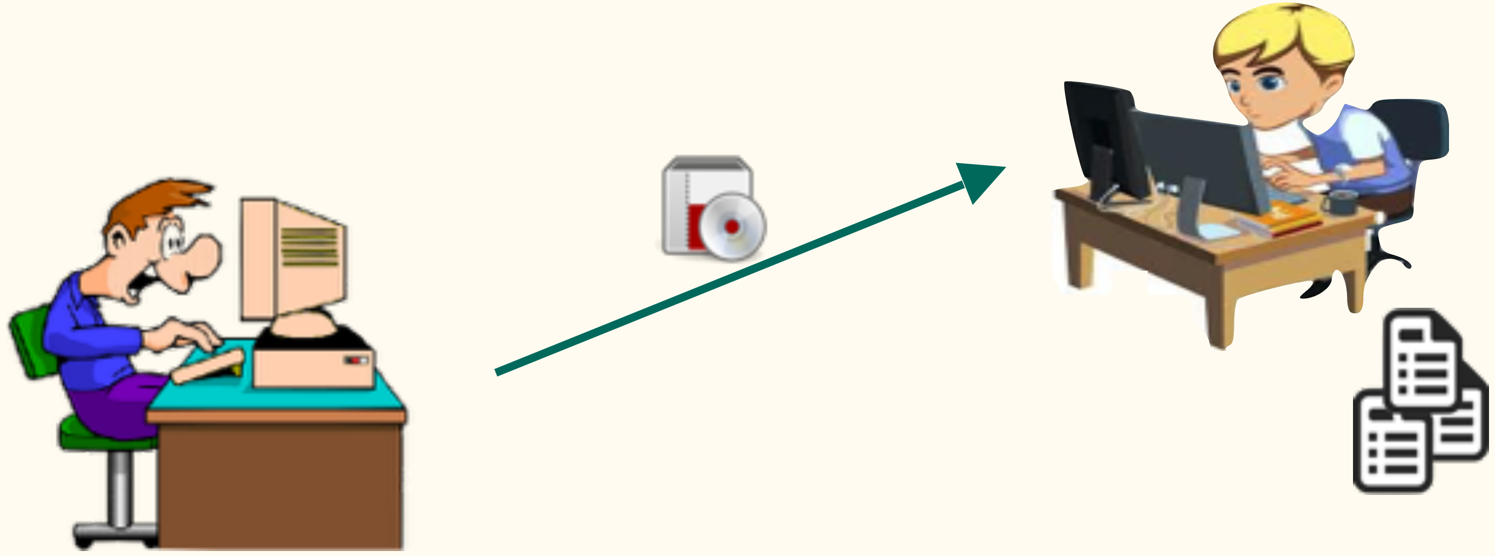
Performance tests are written by developers



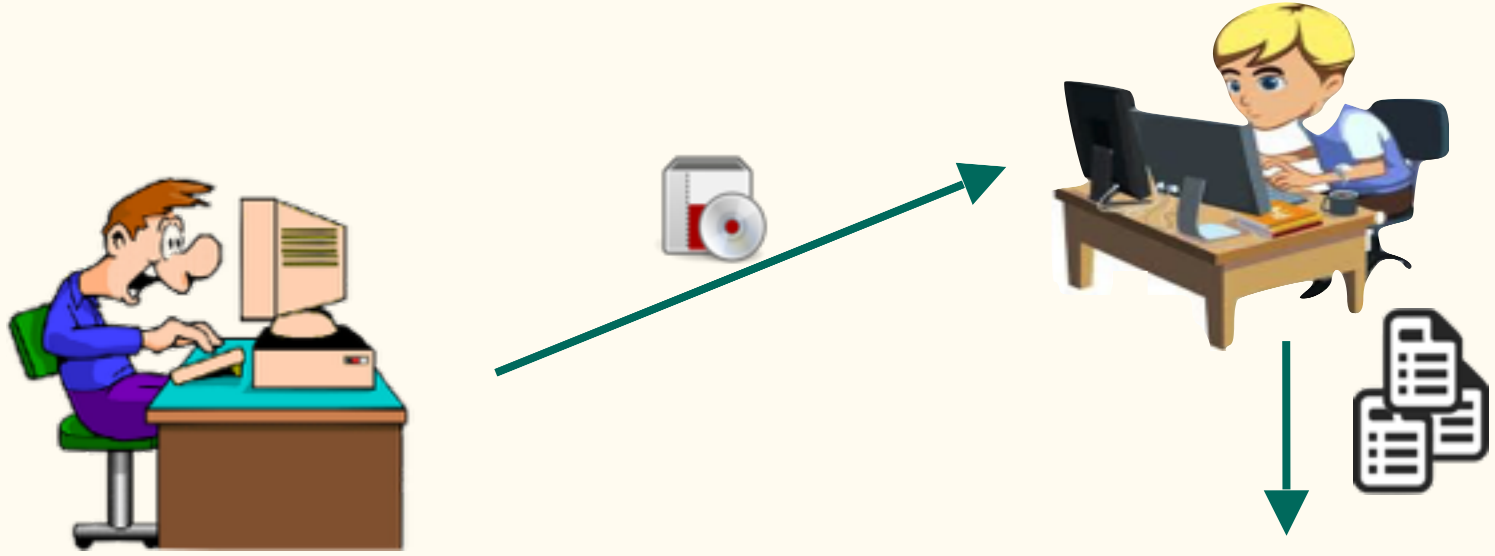
Performance tests are written by developers



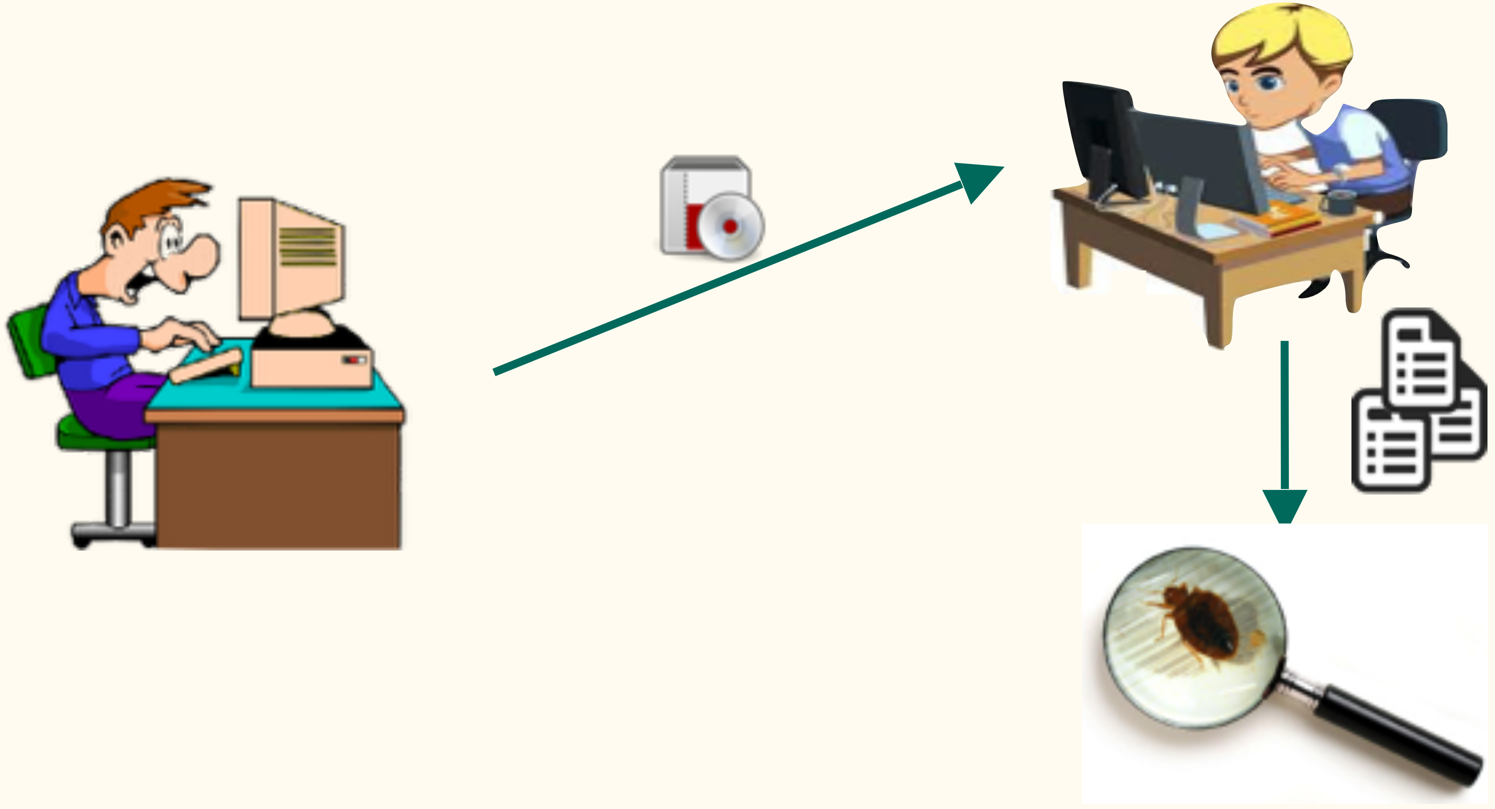
Performance tests are written by developers



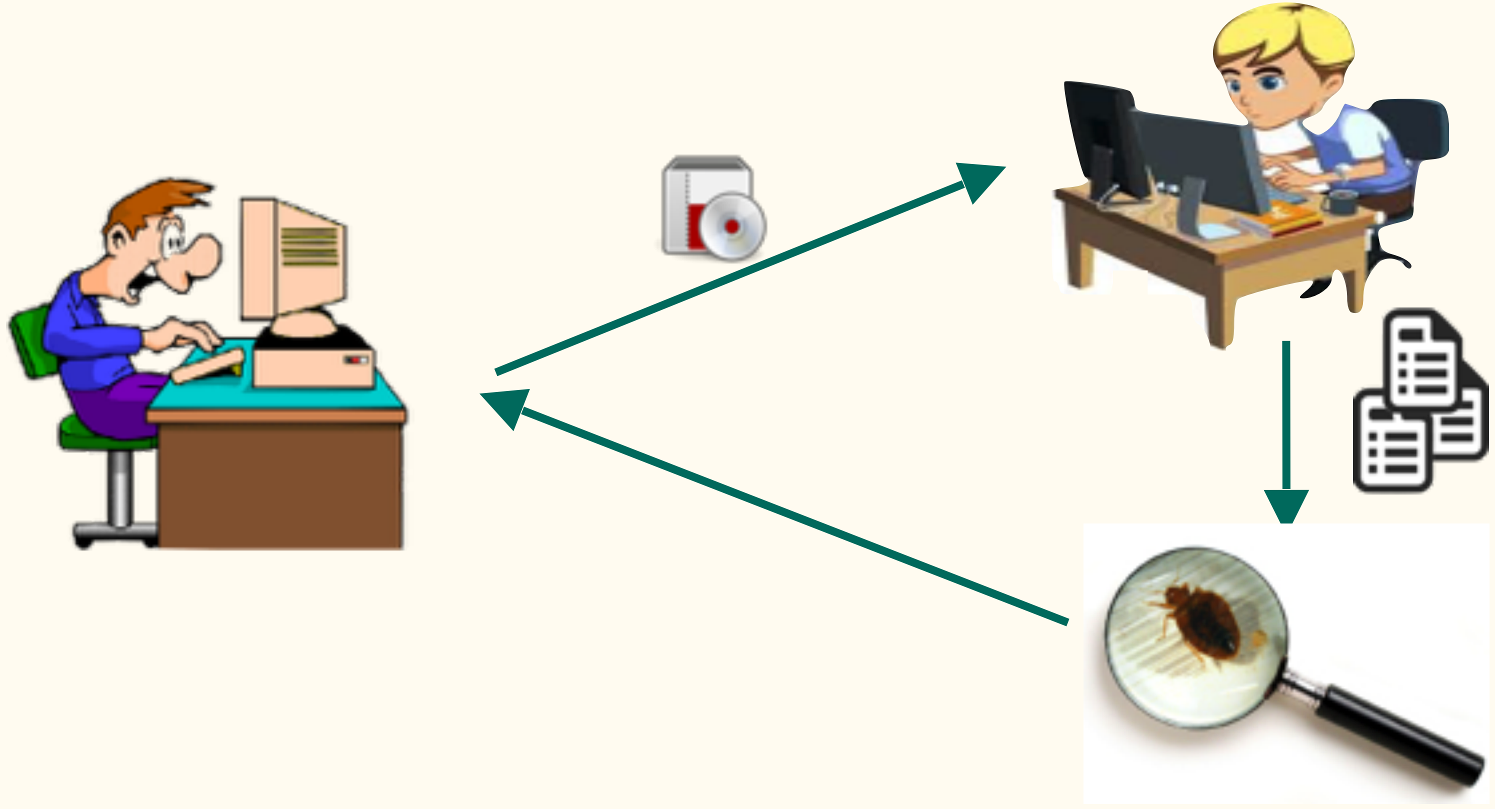
Performance tests are written by developers



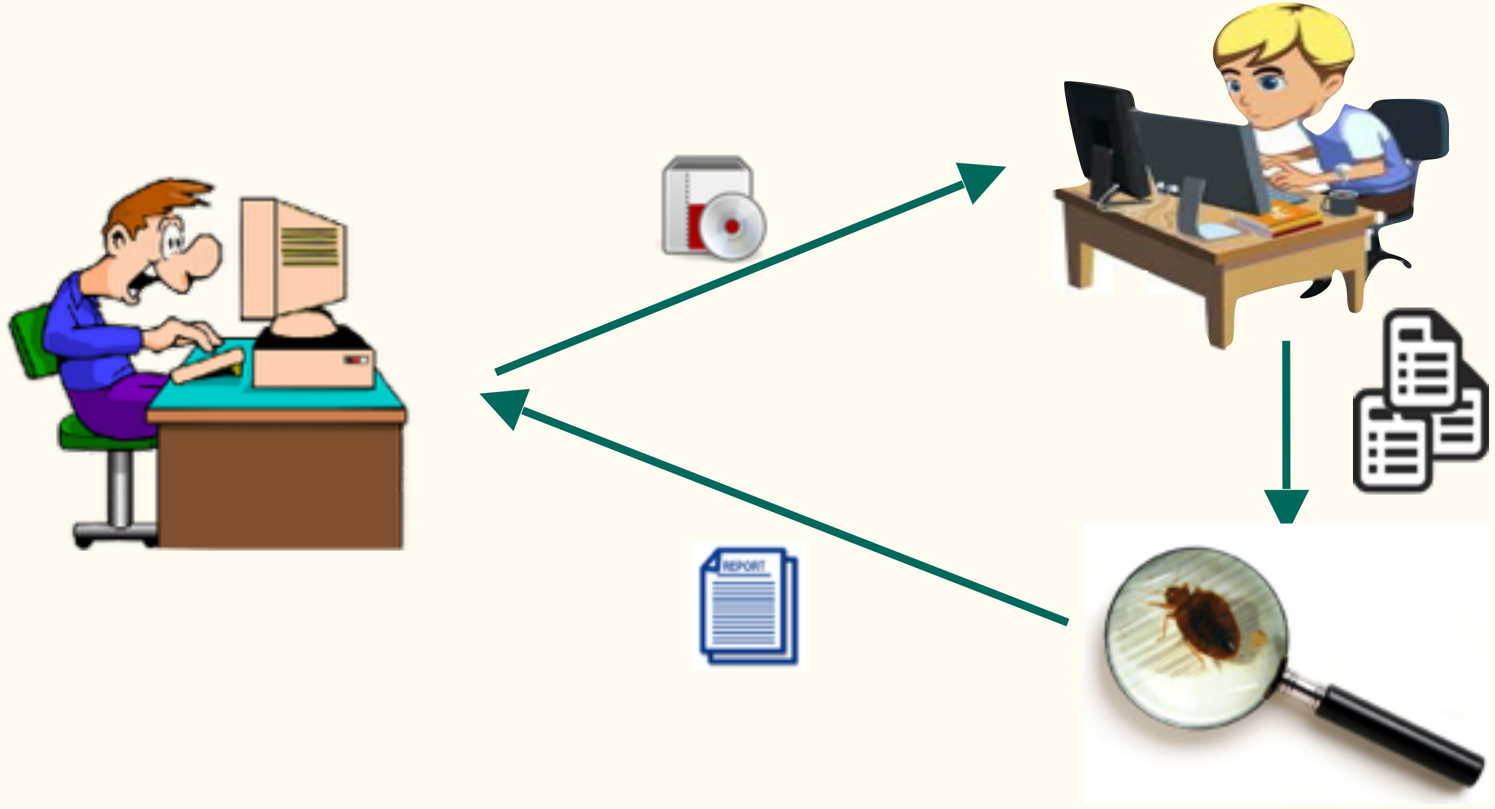
Performance tests are written by developers



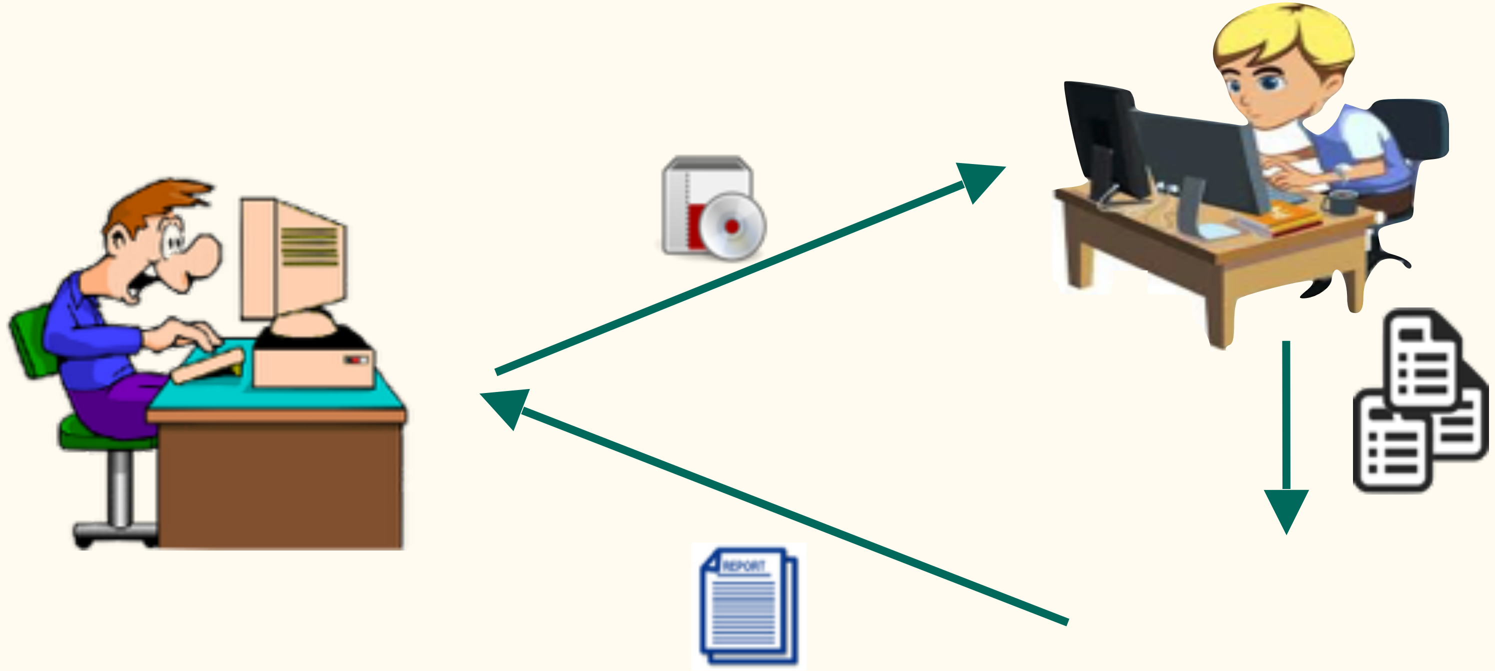
Performance tests are written by developers



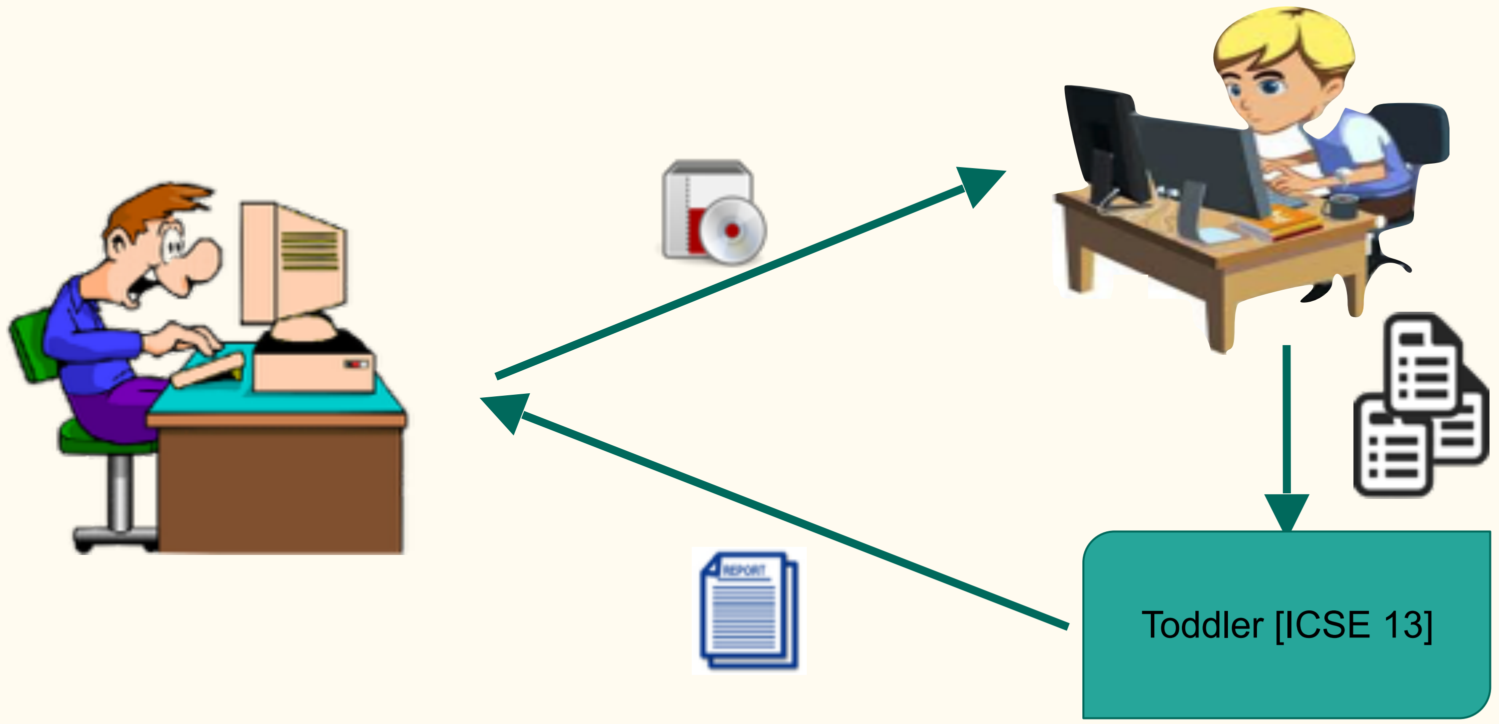
Performance tests are written by developers



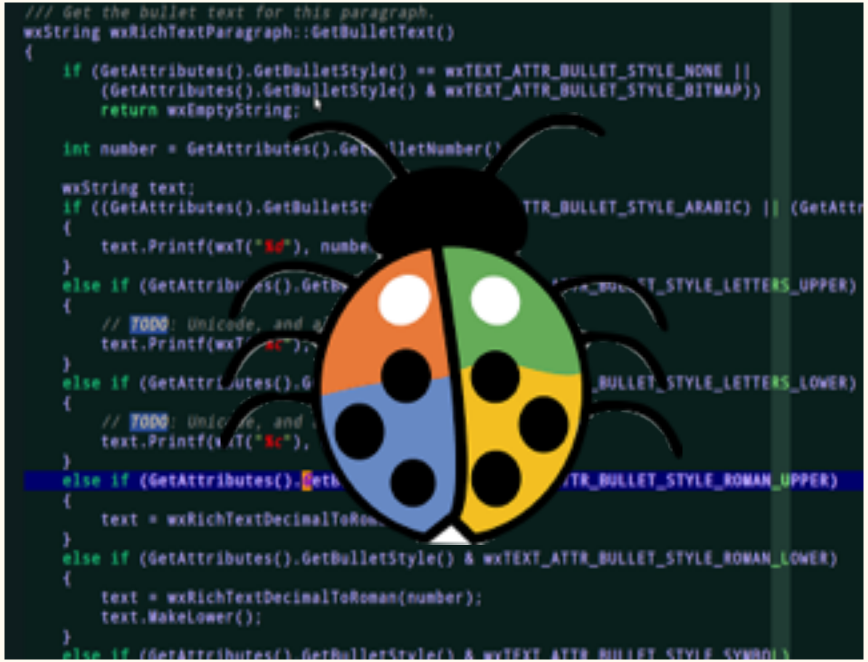
Performance tests are written by developers



Performance tests are written by developers

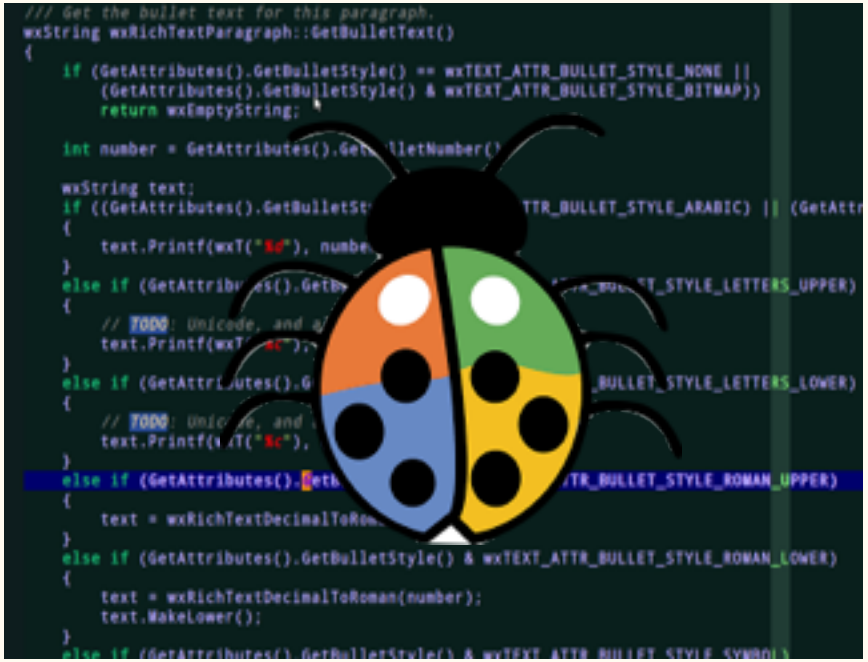


Static analysis techniques alone are not effective



```
/// Get the bullet text for this paragraph.
wxString wxRichTextParagraph::GetBulletText()
{
    if (GetAttributes().GetBulletStyle() == wxTEXT_ATTR_BULLET_STYLE_NONE ||
        (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_BITMAP))
        return wxEmptyString;

    int number = GetAttributes().Get[illegible]lletNumber()

    wxString text;
    if ((GetAttributes().GetBulletSt[illegible]TTR_BULLET_STYLE_ARABIC) || (GetAttr
    {
        text.Printf(wxT("%d"), numbe[illegible]
    }
    else if (GetAttrib[illegible]es().GetB[illegible]_BULLET_STYLE_LETTERS_UPPER)
    {
        // TODO: Unicode, and a[illegible]
        text.Printf(wxT[illegible]%c");
    }
    else if (GetAttri[illegible]utes().G[illegible]BULLET_STYLE_LETTERS_LOWER)
    {
        // TODO: Unic[illegible]e, and [illegible]
        text.Printf(w[illegible]T("%c"),
    }
    else if (GetAttributes().Getb[illegible]TR_BULLET_STYLE_ROMAN_UPPER)
    {
        text = wxRichTextDecimalToRom[illegible]
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_ROMAN_LOWER)
    {
        text = wxRichTextDecimalToRoman(number);
        text.MakeLower();
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_SYMBOL)
```

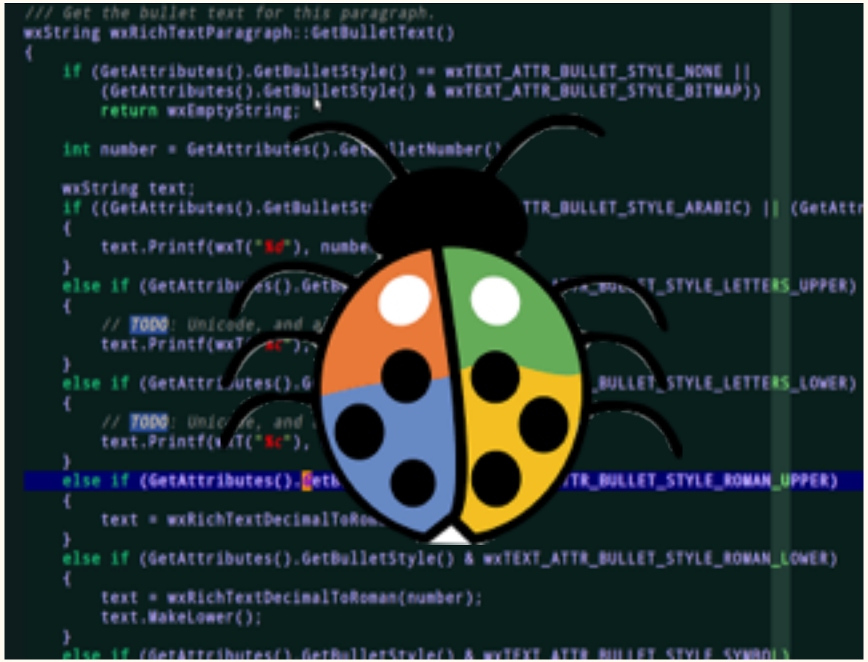
Static analysis techniques alone are not effective



```
/// Get the bullet text for this paragraph.
wxString wxRichTextParagraph::GetBulletText()
{
    if (GetAttributes().GetBulletStyle() == wxTEXT_ATTR_BULLET_STYLE_NONE ||
        (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_BITMAP))
        return wxEmptyString;

    int number = GetAttributes().Get  lletNumber()

    wxString text;
    if ((GetAttributes().GetBulletSt                    TTR_BULLET_STYLE_ARABIC) || (GetAttr
    {
        text.Printf(wxT("%d"), numbe
    }
    else if (GetAttribu  es().GetB                        T_STYLE_LETTERS_UPPER)
    {
        // TODO: Unicode, and a
        text.Printf(wxT  %c");
    }
    else if (GetAttri utes().G                          BULLET_STYLE_LETTERS_LOWER)
    {
        // TODO: Unic  e, and
        text.Printf( xT("%c"),
    }
    else if (GetAttributes().Getb                        TR_BULLET_STYLE_ROMAN_UPPER)
    {
        text = wxRichTextDecimalToRom
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_ROMAN_LOWER)
    {
        text = wxRichTextDecimalToRoman(number);
        text.MakeLower();
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_SYMBOL)
```

Challenges :

Static analysis techniques alone are not effective

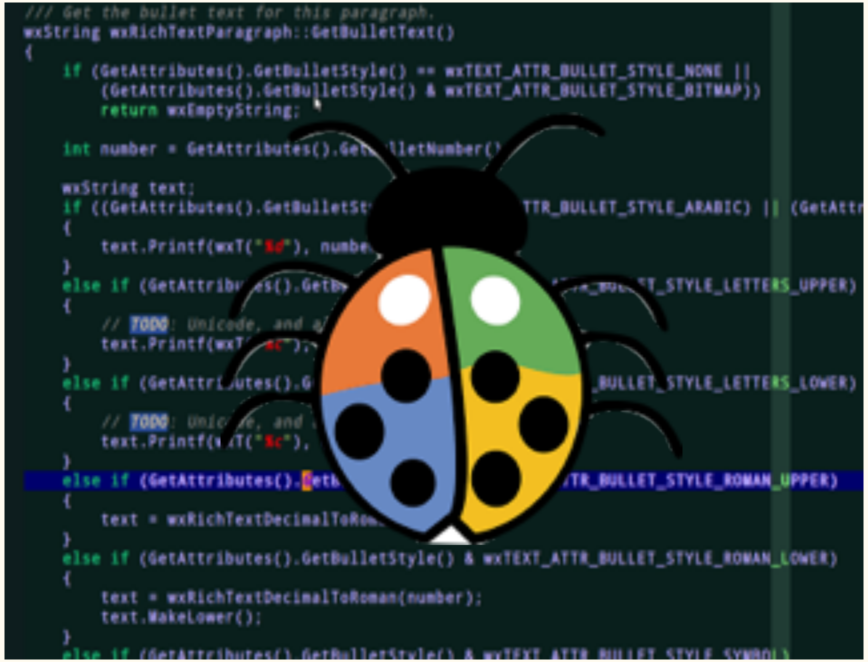


```
/// Get the bullet text for this paragraph.
wxString wxRichTextParagraph::GetBulletText()
{
    if (GetAttributes().GetBulletStyle() == wxTEXT_ATTR_BULLET_STYLE_NONE ||
        (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_BITMAP))
        return wxEmptyString;

    int number = GetAttributes().Ge   lletNumber()

    wxString text;
    if ((GetAttributes().GetBulletSt          TTR_BULLET_STYLE_ARABIC) || (GetAttr
    {
        text.Printf(wxT("%d"), numbe
    }
    else if (GetAttrib  es().GetB              T_STYLE_LETTERS_UPPER)
    {
        // TODO: Unicode, and a
        text.Printf(wxT  %c");
    }
    else if (GetAttri  utes().G                 BULLET_STYLE_LETTERS_LOWER)
    {
        // TODO: Unic  e, and
        text.Printf(  T("%c"),
    }
    else if (GetAttributes().Getb               TR_BULLET_STYLE_ROMAN_UPPER)
    {
        text = wxRichTextDecimalToRom
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_ROMAN_LOWER)
    {
        text = wxRichTextDecimalToRoman(number);
        text.MakeLower();
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_SYMBOL)
```

Challenges :

- How to confirm the validity of the bug?

Static analysis techniques alone are not effective

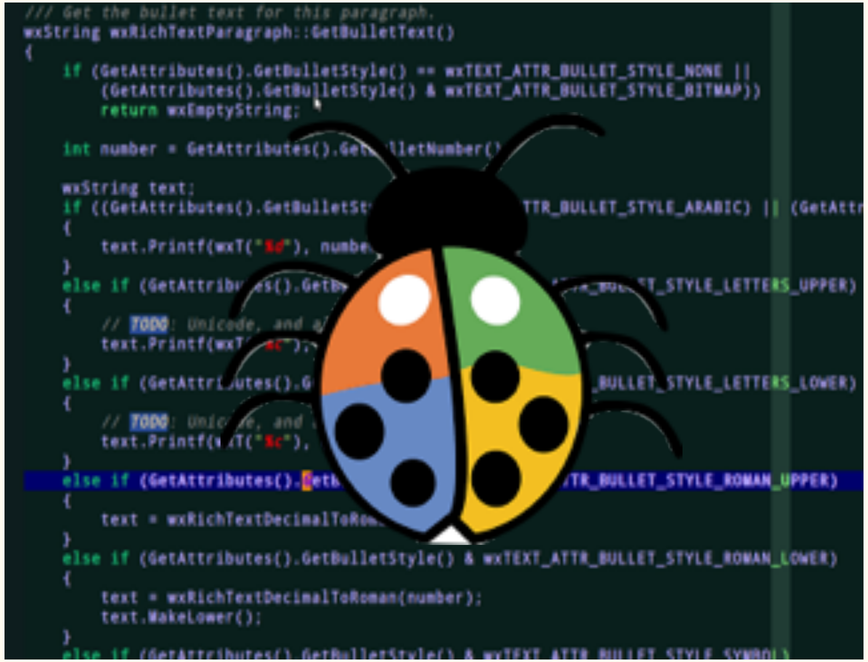


```
/// Get the bullet text for this paragraph.
wxString wxRichTextParagraph::GetBulletText()
{
    if (GetAttributes().GetBulletStyle() == wxTEXT_ATTR_BULLET_STYLE_NONE ||
        (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_BITMAP))
        return wxEmptyString;

    int number = GetAttributes().Ge[illegible]lletNumber()

    wxString text;
    if ((GetAttributes().GetBulletSt[illegible]TTR_BULLET_STYLE_ARABIC) || (GetAttr
    {
        text.Printf(wxT("%d"), numbe[illegible]
    }
    else if (GetAttrib[illegible]es().GetB[illegible]T_STYLE_LETTERS_UPPER)
    {
        // TODO: Unicode, and a[illegible]
        text.Printf(wxT[illegible]%c");[illegible]
    }
    else if (GetAttri[illegible]utes().G[illegible]BULLET_STYLE_LETTERS_LOWER)
    {
        // TODO: Unic[illegible]e, and [illegible]
        text.Printf(w[illegible]T("%c"),[illegible]
    }
    else if (GetAttributes().Getb[illegible]TR_BULLET_STYLE_ROMAN_UPPER)
    {
        text = wxRichTextDecimalToRom[illegible]
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_ROMAN_LOWER)
    {
        text = wxRichTextDecimalToRoman(number);
        text.MakeLower();
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_SYMBOL)
```

Challenges :

- How to confirm the validity of the bug?
- How to expose the root cause?

Static analysis techniques alone are not effective



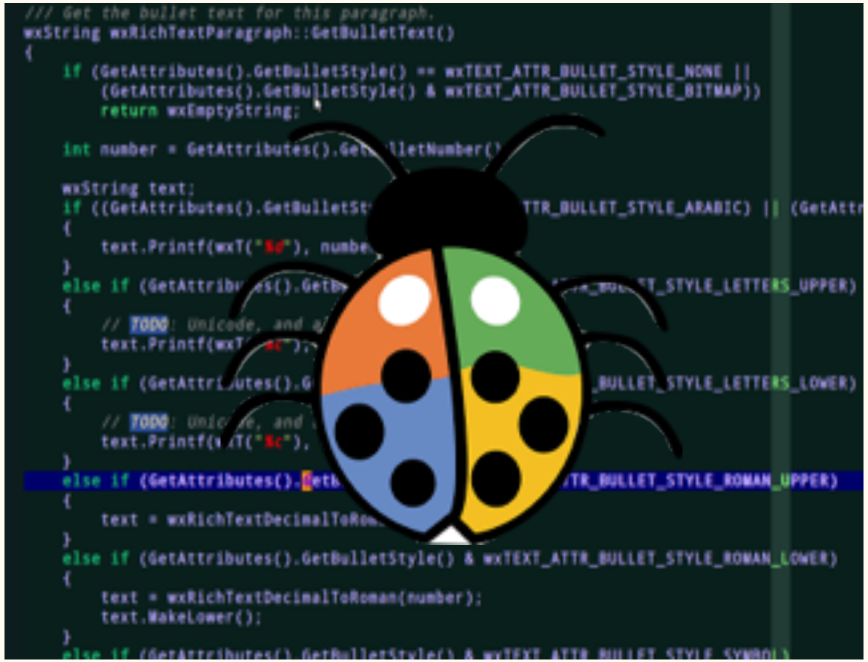
```
/// Get the bullet text for this paragraph.
wxString wxRichTextParagraph::GetBulletText()
{
    if (GetAttributes().GetBulletStyle() == wxTEXT_ATTR_BULLET_STYLE_NONE ||
        (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_BITMAP))
        return wxEmptyString;

    int number = GetAttributes().Get[illegible]lletNumber()

    wxString text;
    if ((GetAttributes().GetBulletSt[illegible]TTR_BULLET_STYLE_ARABIC) || (GetAttr
    {
        text.Printf(wxT("%d"), numbe[illegible]
    }
    else if (GetAttribu[illegible]es().GetB[illegible]T_STYLE_LETTERS_UPPER)
    {
        // TODO: Unicode, and a[illegible]
        text.Printf(wxT[illegible]%c");[illegible]
    }
    else if (GetAttri[illegible]utes().G[illegible]_BULLET_STYLE_LETTERS_LOWER)
    {
        // TODO: Unic[illegible]e, and [illegible]
        text.Printf(w[illegible]T("%c"),[illegible]
    }
    else if (GetAttributes().Getb[illegible]TR_BULLET_STYLE_ROMAN_UPPER)
    {
        text = wxRichTextDecimalToRom[illegible]
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_ROMAN_LOWER)
    {
        text = wxRichTextDecimalToRoman(number);
        text.MakeLower();
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_SYMBOL)
```

Challenges :

- How to confirm the validity of the bug?

- How to expose the root cause?
    - Execution trace can be helpful

Static analysis techniques alone are not effective



```
/// Get the bullet text for this paragraph.
wxString wxRichTextParagraph::GetBulletText()
{
    if (GetAttributes().GetBulletStyle() == wxTEXT_ATTR_BULLET_STYLE_NONE ||
        (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_BITMAP))
        return wxEmptyString;

    int number = GetAttributes().Get[illegible]lletNumber()

    wxString text;
    if ((GetAttributes().GetBulletSt[illegible]TTR_BULLET_STYLE_ARABIC) || (GetAttr
    {
        text.Printf(wxT("%d"), numbe[illegible]
    }
    else if (GetAttrib[illegible]es().GetB[illegible]T_STYLE_LETTERS_UPPER)
    {
        // TODO: Unicode, and a[illegible]
        text.Printf(wxT[illegible]%c");[illegible]
    }
    else if (GetAttri[illegible]utes().G[illegible]BULLET_STYLE_LETTERS_LOWER)
    {
        // TODO: Unic[illegible]e, and [illegible]
        text.Printf([illegible]xT("%c"), [illegible]
    }
    else if (GetAttributes().GetB[illegible]TR_BULLET_STYLE_ROMAN_UPPER)
    {
        text = wxRichTextDecimalToRom[illegible]
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_ROMAN_LOWER)
    {
        text = wxRichTextDecimalToRoman(number);
        text.MakeLower();
    }
    else if (GetAttributes().GetBulletStyle() & wxTEXT_ATTR_BULLET_STYLE_SYMBOL)
```

Challenges :

- How to confirm the validity of the bug?
- How to expose the root cause?
  - Execution trace can be helpful
- How to detect that the performance bug is fixed?

Challenges involved in writing performance tests

Challenges involved in writing performance tests

Virtual call resolution

Generating tests for all possible resolutions of method invocation is not scalable

Challenges involved in writing performance tests

Virtual call resolution

Generating tests for all possible resolutions of method invocation is not scalable

Generating appropriate context

Realization of the defect can be dependent on certain conditions that affect the reachability of the inefficient loop

Challenges involved in writing performance tests

Virtual call resolution

Generating tests for all possible resolutions of method invocation is not scalable

Generating appropriate context

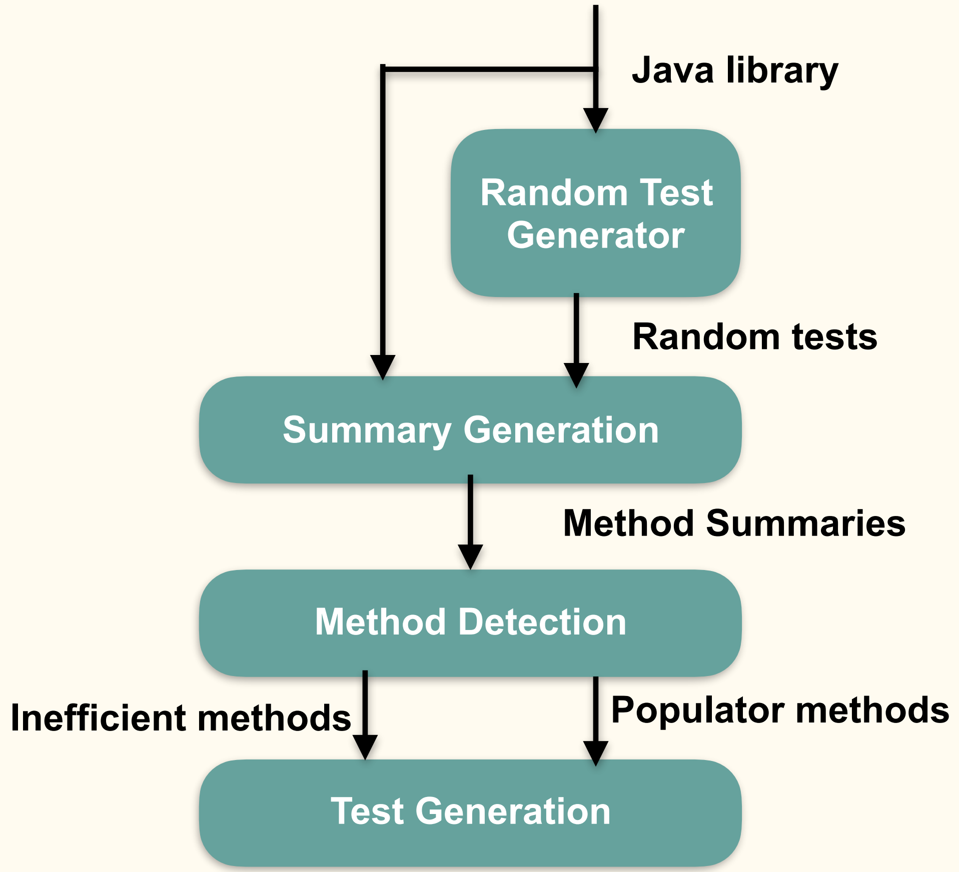
Realization of the defect can be dependent on certain conditions that affect the reachability of the inefficient loop

Arrangement of elements

Problem can only occur when data structure has large elements arranged in particular fashion

Glider

We propose a novel and scalable approach to automatically generate tests for exposing loop inefficiencies

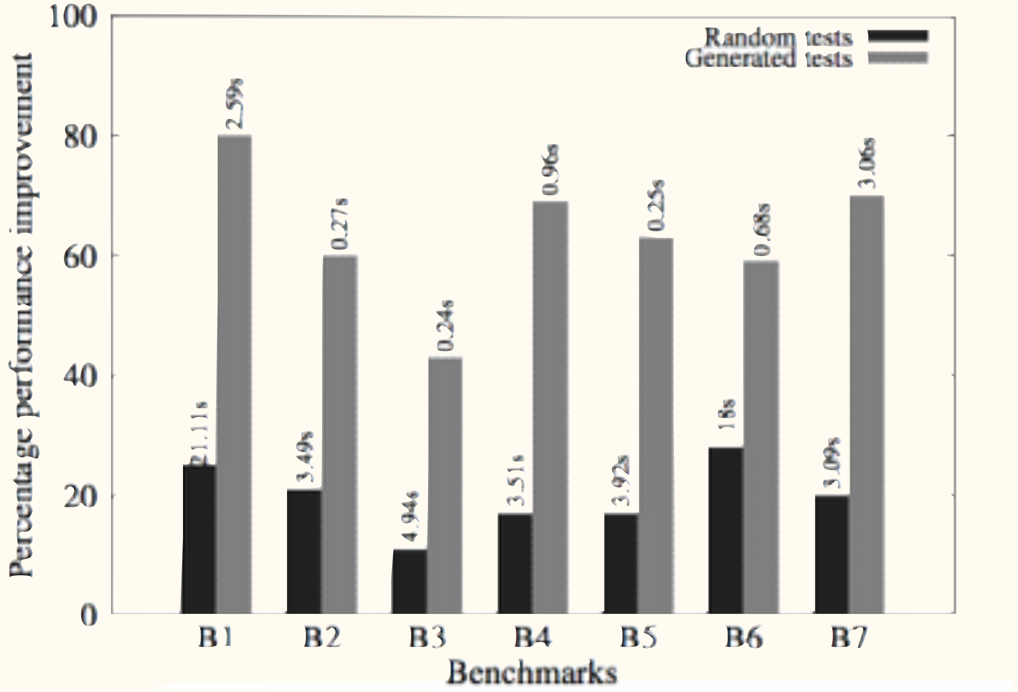


RQ1 : Effectiveness of test generation

Benchmark	ID	# Generated tests	# Bugs	# New bugs	# False positives	Analysis time (min)
Apache collections	B1	80	16	9	1	45
PDFBox	B2	30	6	6	0	12
Groovy	B3	20	5	4	0	7
Guava	B4	50	9	10	1	28
JFreeChart	B5	15	3	1	4	24
Ant	B6	24	6	3	1	15
Lucent	B7	5	1	1	0	16

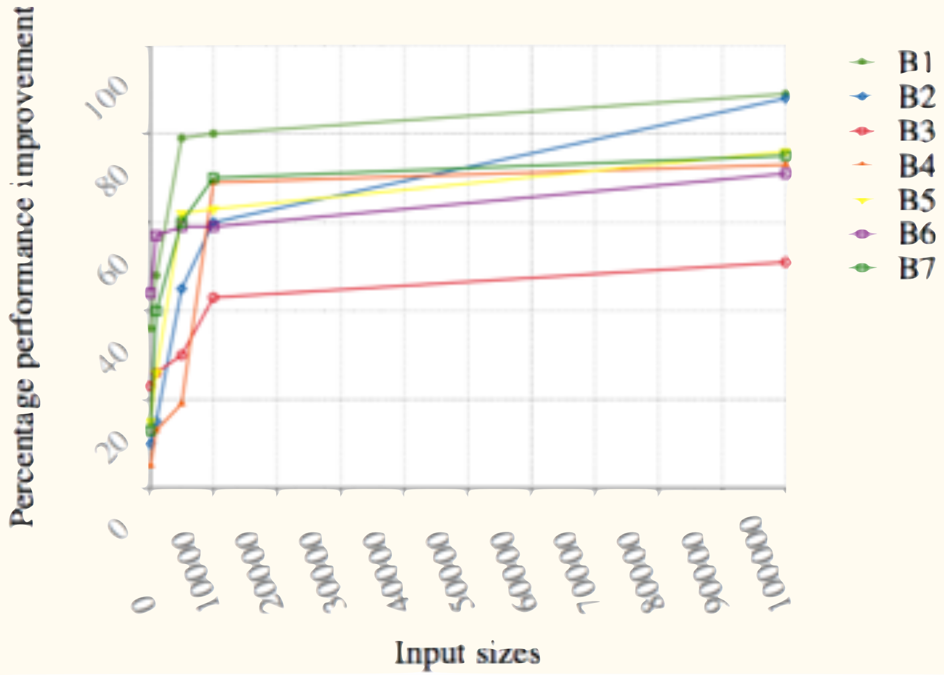
Our approach is able to generate useful tests using random tests

RQ2 : Comparison with randomly generated tests



Generated tests are more suitable to expose the magnitude of performance problem

RQ3 : Size of collection objects



Collection objects with 10K elements will enable detection of performance issues

Results

Results

We have **implemented our approach on SOOT** bytecode framework and evaluated it on number of libraries

Results

We have **implemented our approach on SOOT** bytecode framework and evaluated it on number of libraries

Our approach detected **46 bugs across 7 java libraries** including 34 previously unknown bugs.

Results

We have **implemented our approach on SOOT** bytecode framework and evaluated it on number of libraries

Our approach detected **46 bugs across 7 java libraries** including 34 previously unknown bugs.

Tests generated using our approach **significantly outperform the randomly generated tests**.