

Automatically Documenting Unit Test Cases

Boyang Li, Christopher Vendome, Mario Linares-Vasquez, Denys Poshyvanyk, and Nicholas A. Kraft*

College of William and Mary, Williamsburg, VA, USA

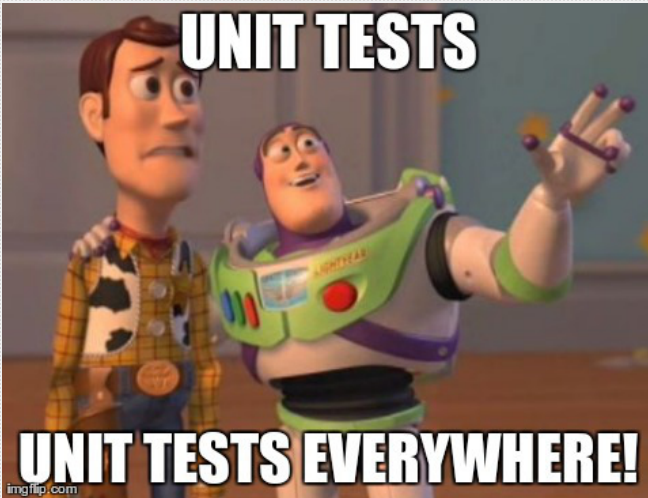
*ABB Corporate Research Center, Raleigh, NC, USA



ICST2016

Unit Test Case Maintenance

Maintaining unit test cases requires comprehension of the unit test code. Comments in unit test cases are added with the purpose of making the test cases easier to understand.



Challenges:

- Unit test cases may lack comments.
- Source code changes could lead to inconsistencies in unit test case comments.

Our Contributions



A survey of both open-source and industrial developers



A mining-based study on a large dataset of C# projects



An approach to automatically generate natural language descriptions to document the purpose of unit test cases

} 3 RQs

} 5 RQs

Research Questions

RQ1. To what extent do unit test cases contain comments?

RQ2. To what extent do developers update unit test case comments?

RQ3. To what extent do developers have difficulty understanding unit test cases?

Methodology



A survey



A mining-based analysis

Methodology



A survey

with



212 developers



A mining-based analysis

Methodology



A survey

with



212 developers



A mining-based analysis

on



53,735 unit tests

Methodology



A survey

with



212 developers



A mining-based analysis

on



53,735 unit tests

Methodology



A survey

with



212 developers



A mining-based analysis

on



53,735 unit tests

Methodology



A survey

with



212 developers



Results

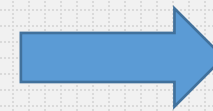


A mining-based analysis

on



53,735 unit tests



Results

Data Collection

All C# projects **GitHub**

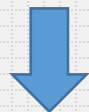


2,209 projects



4,115 developers

565 developers **ABB**



212 completed

Data Collection

All C# projects



2,209 projects

GitHub

 ≥ 1  ≥ 1  ≥ 1



1,414 projects

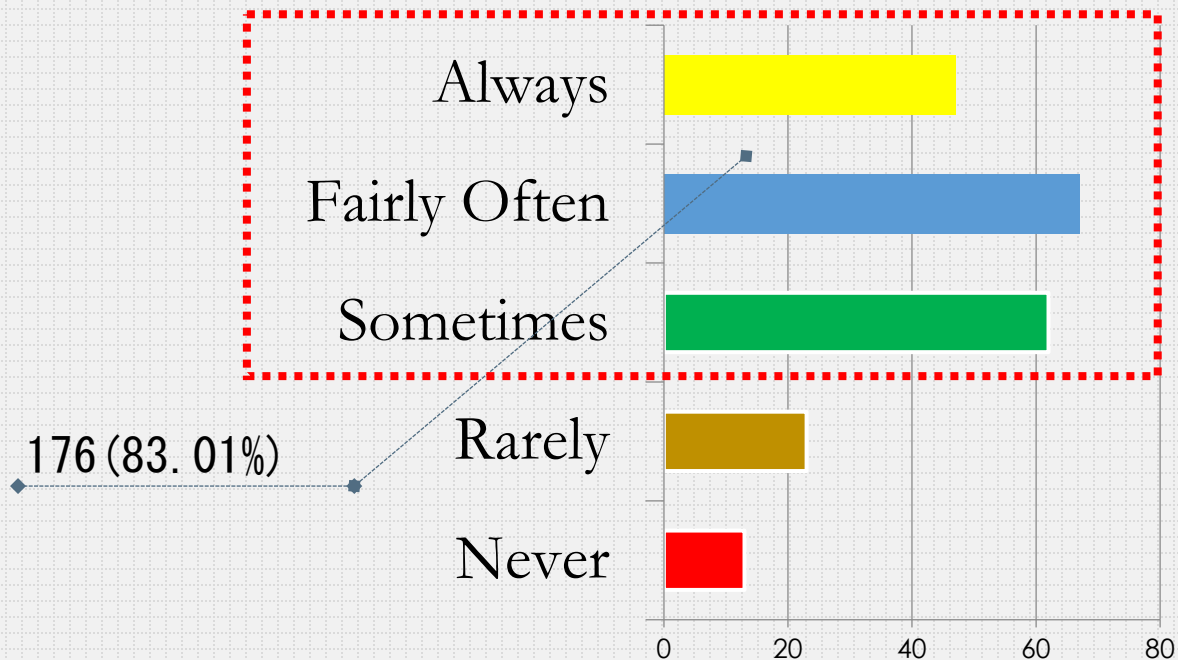
Randomly selected



53,735 unit test cases

RQ1. To what extent do unit test cases contain comments?

1. How often do you write unit test cases for your project(s)?

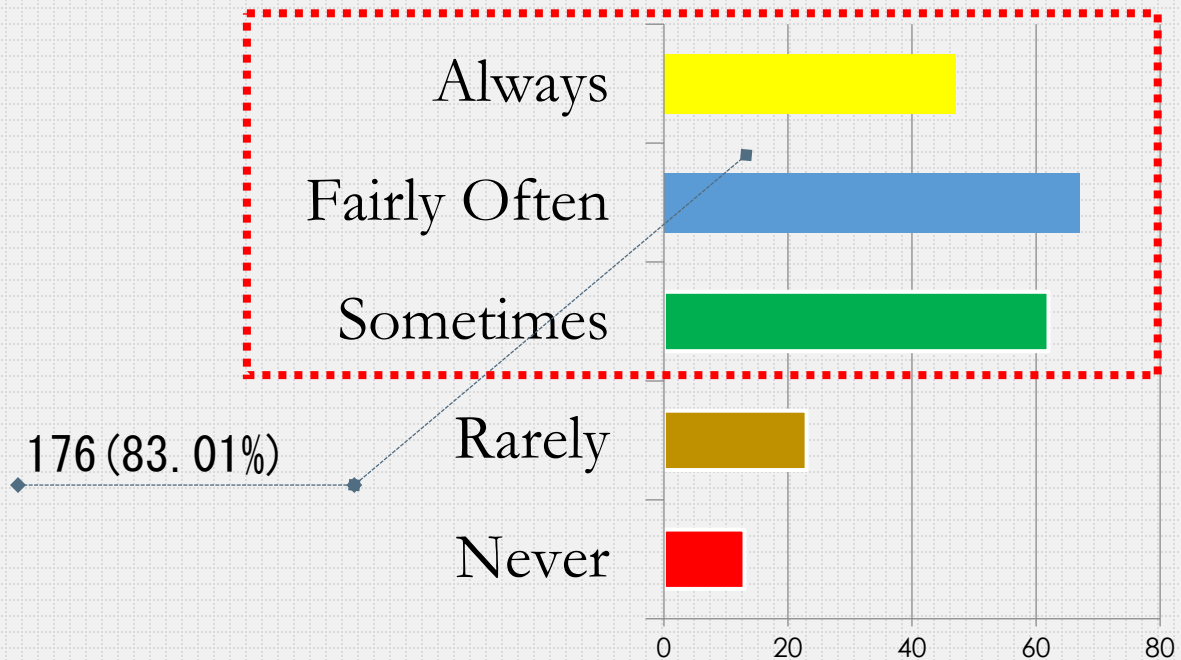


2. How often do you write comments for unit test cases?



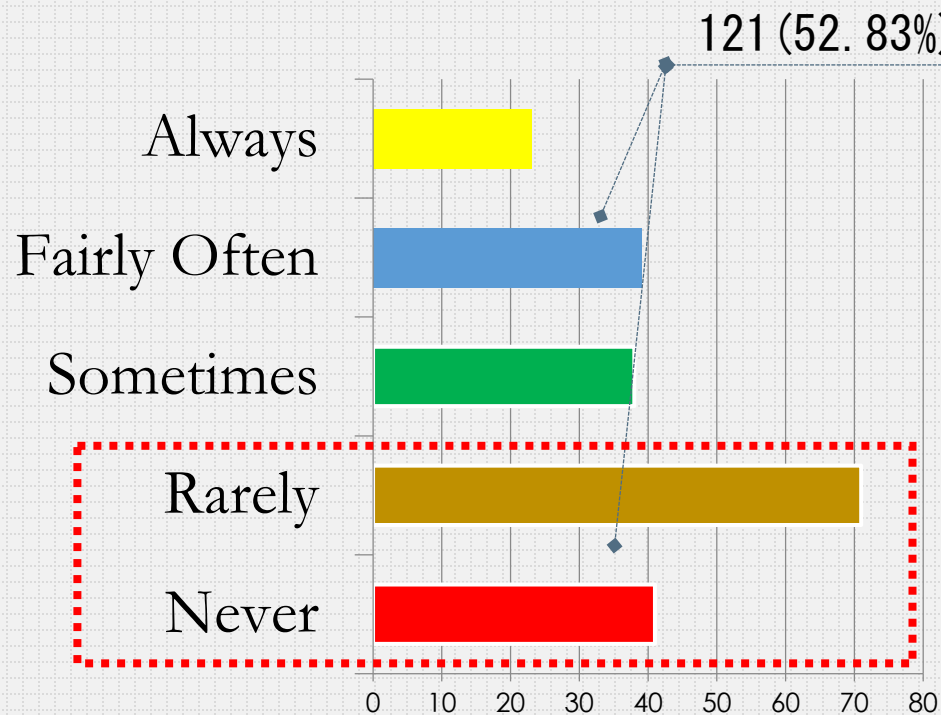
RQ1. To what extent do unit test cases contain comments?

Survey results



RQ1. To what extent do unit test cases contain comments?

Survey results

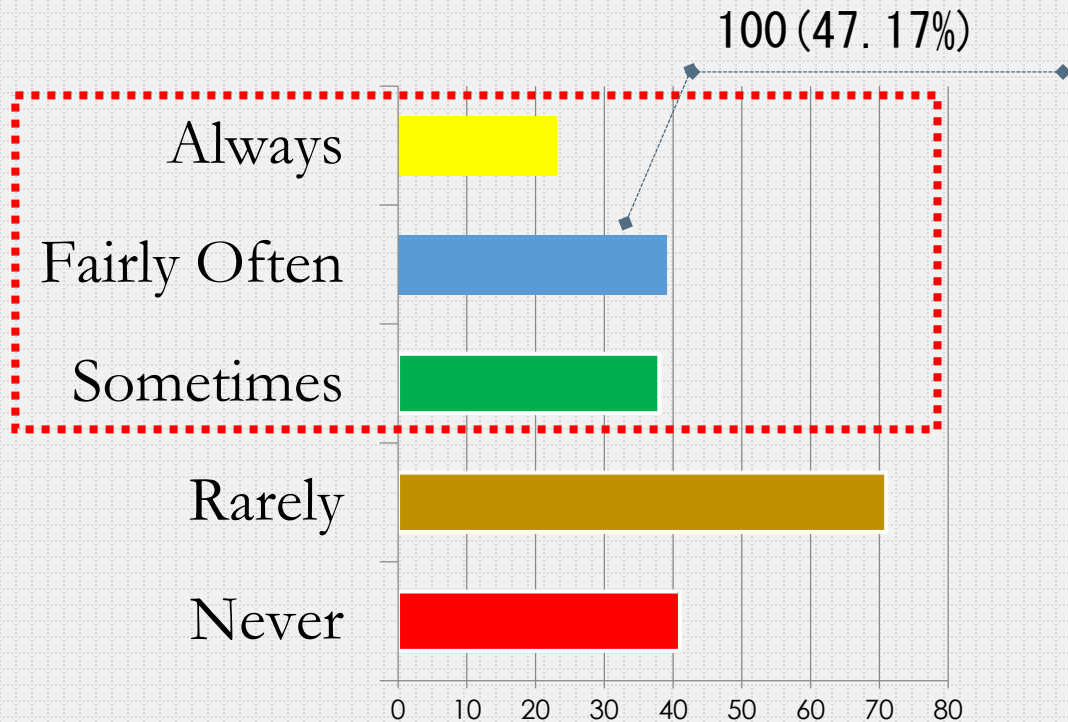


"Comments need to be maintained which adds complexity to the task."

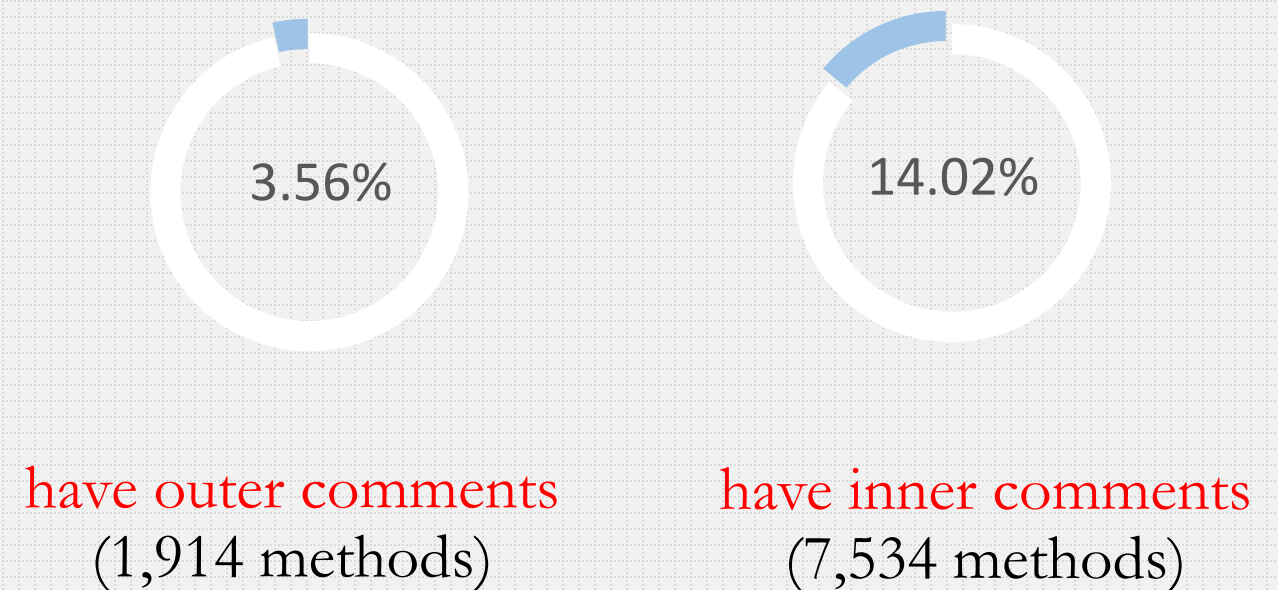
"I use very verbose naming of tests to be the documentation, along with meaningful naming of methods and variables used in the test."

RQ1. To what extent do unit test cases contain comments?

Survey results

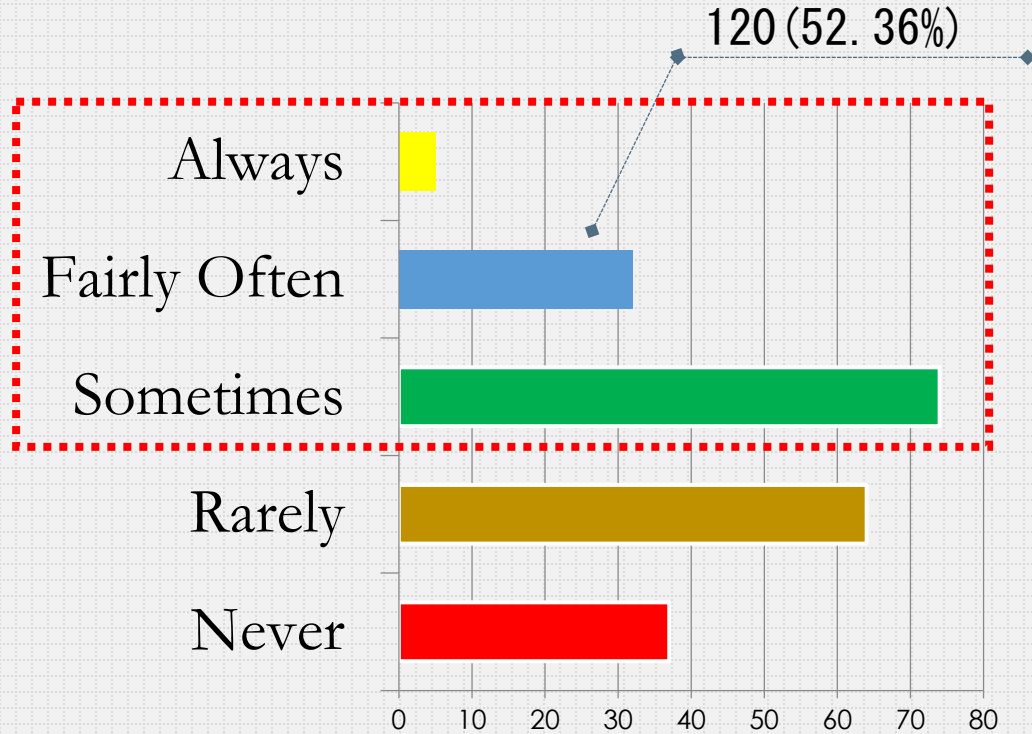


Repository mining results

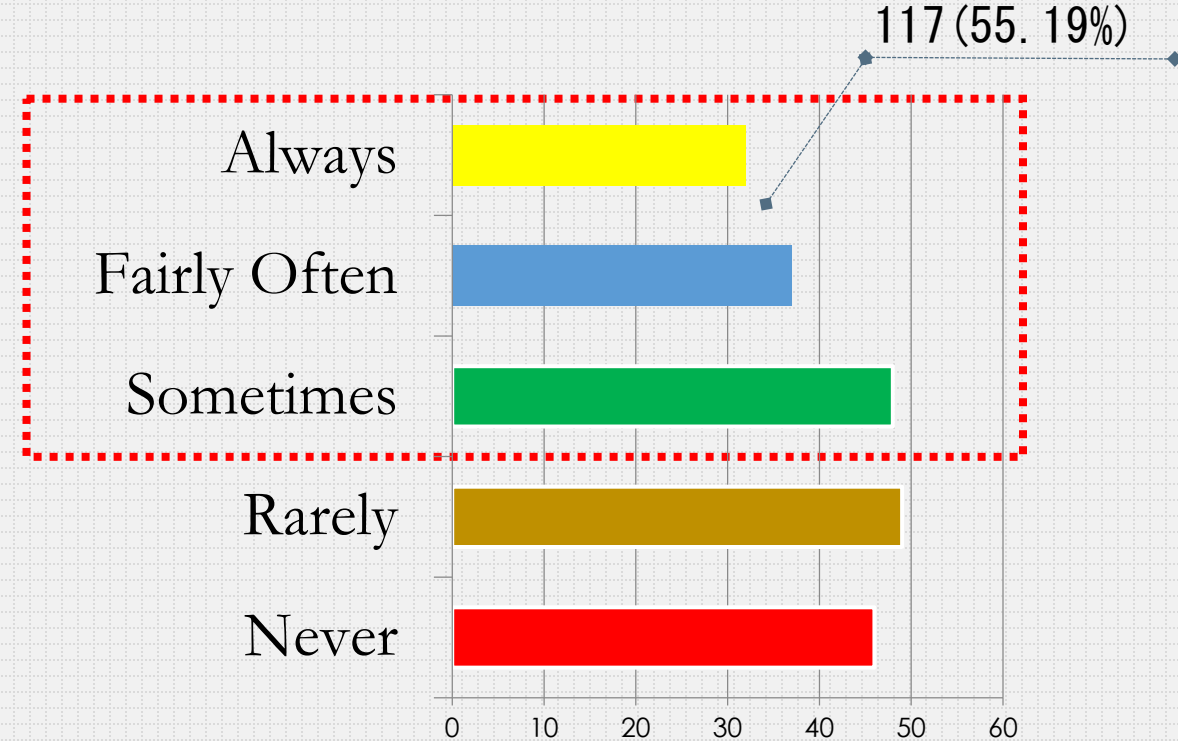


RQ2. To what extent do developers update unit test case comments?

3. How often do you find outdated comments in unit tests?

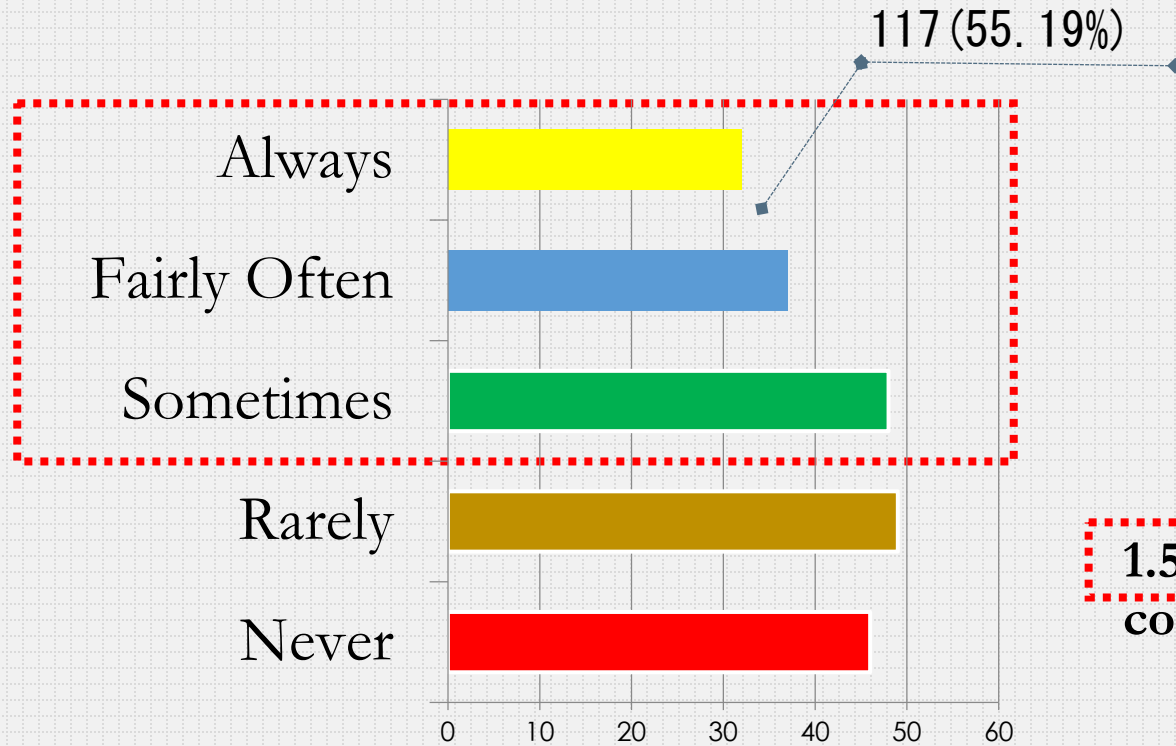


4. When you make changes to the unit tests, how often do you comment the changes?

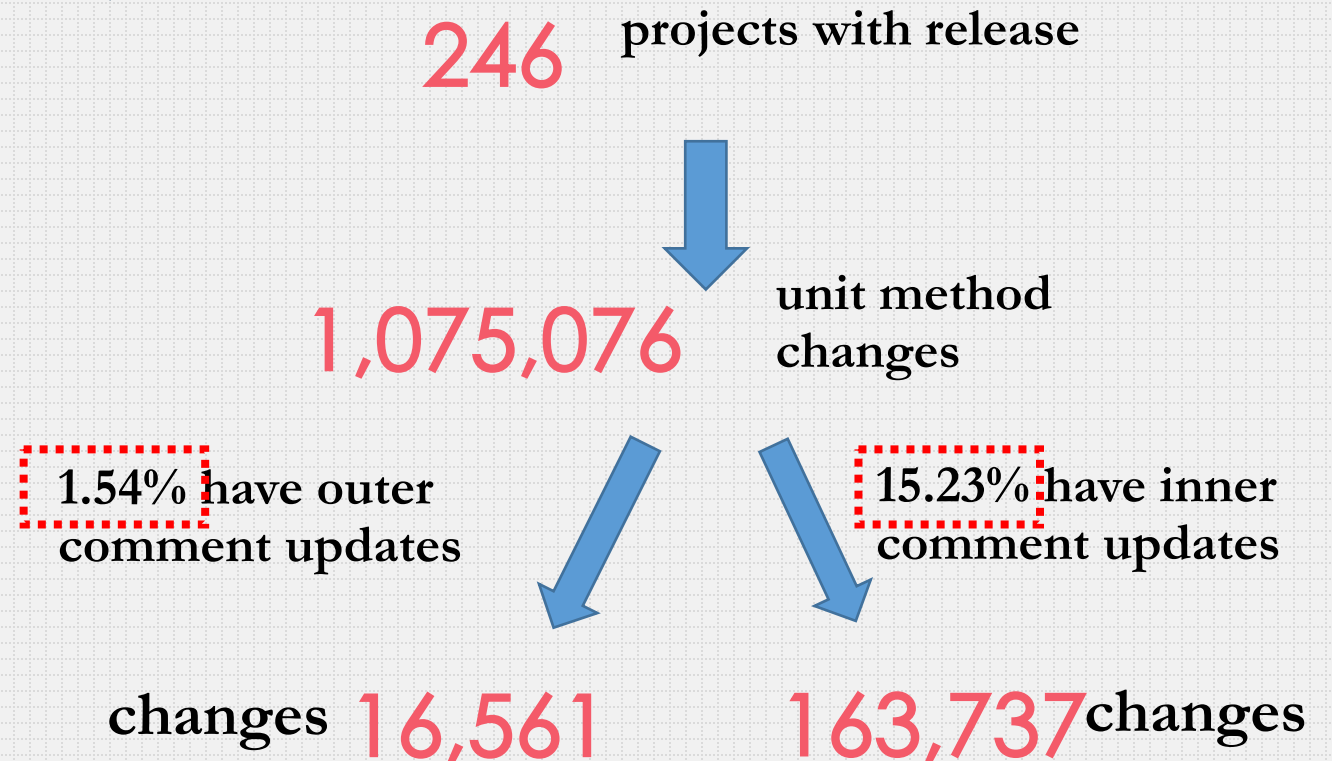


RQ2. To what extent do developers update unit test case comments?

Survey results

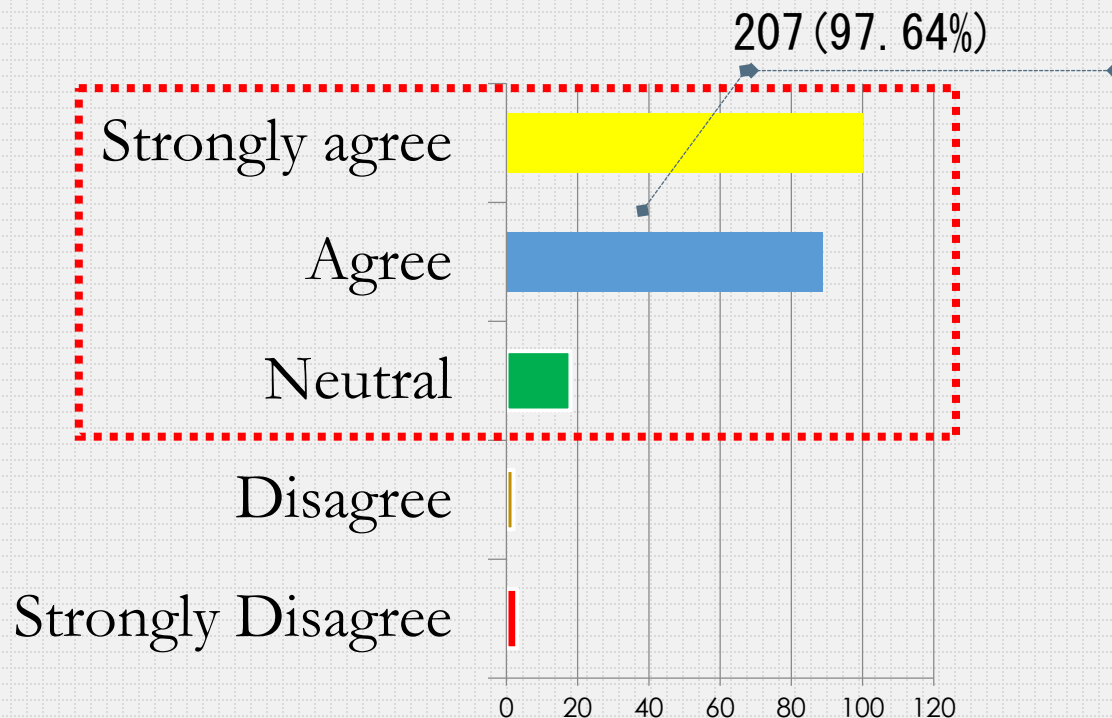


Repository mining results

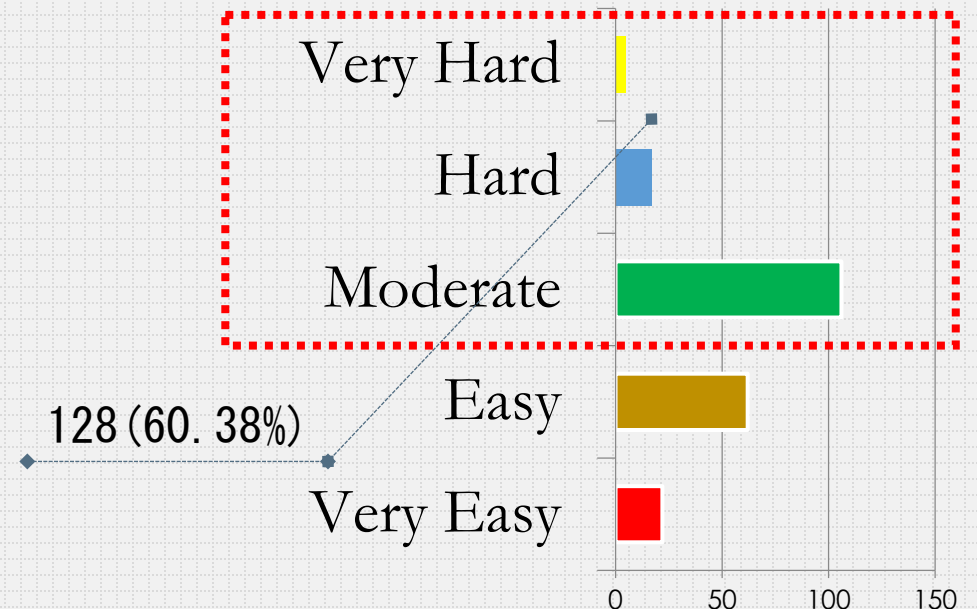


RQ3. To what extent, do developers have difficulty understanding unit test cases?

5. Maintaining good unit test cases and documentation is important to the quality of a system?



6. How difficult is it to understand a unit test (i.e., identifying focal methods)?



Lessons learnt from RQ1 – RQ3

(i) Documenting unit test cases is not a common practice in Github projects

Documentation

(ii) Developers do not update comments when changes are made to unit test cases

Automation

(iii) Understanding unit test cases is generally not an easy task

Understanding

UnitTestScribe

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

🗨 This unit test case method tests add words several times .

Code with line numbers

🔔 This unit test case includes following focal methods:

(1) `var count = matrix.GetCoOccurrenceCount(word1,word2);`(@line 47)

This focal method is related to assertions at [line 48](#)

🔔 This unit test case validates that:

(1) `count > 0` is true.

`count` is obtained from

1) variable `listLength` through [slicing path](#)

2) variable `i` through [slicing path](#)

3) variable `listLength` through [slicing path](#)

`listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)`

- Natural language descriptions
- Focal methods
- Assertion description
- Internal data dependencies

UnitTestScribe

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

Code with line numbers

💬 This unit test case method tests add words several times .

🔔 This unit test case includes following focal methods:

(1) `var count = matrix.GetCoOccurrenceCount(word1,word2);(@line 47)`

This focal method is related to assertions at [line 48](#)

🔔 This unit test case validates that:

(1) `count > 0` is true.

`count` is obtained from

1) variable `listLength` through [slicing path](#)

2) variable `i` through [slicing path](#)

3) variable `listLength` through [slicing path](#)

```
listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)
```

- Natural language descriptions
- Focal methods
- Assertion description
- Internal data dependencies

UnitTestScribe

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

Code with line numbers

💬 This unit test case method tests add words several times .

🔔 This unit test case includes following focal methods:

(1) `var count = matrix.GetCoOccurrenceCount(word1,word2);`(@line 47)

This focal method is related to assertions at [line 48](#)

🔔 This unit test case validates that:

(1) `count > 0` is true.

`count` is obtained from

1) variable `listLength` through [slicing path](#)

2) variable `i` through [slicing path](#)

3) variable `listLength` through [slicing path](#)

`listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)`

- Natural language descriptions
- Focal methods
- Assertion description
- Internal data dependencies

UnitTestScribe

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

Code with line numbers

💬 This unit test case method tests add words several times .

🔔 This unit test case includes following focal methods:

(1) `var count = matrix.GetCoOccurrenceCount(word1,word2);`(@line 47)

This focal method is related to assertions at [line 48](#)

🔔 This unit test case validates that:

(1) `count > 0` is true.

`count` is obtained from

1) variable `listLength` through [slicing path](#)

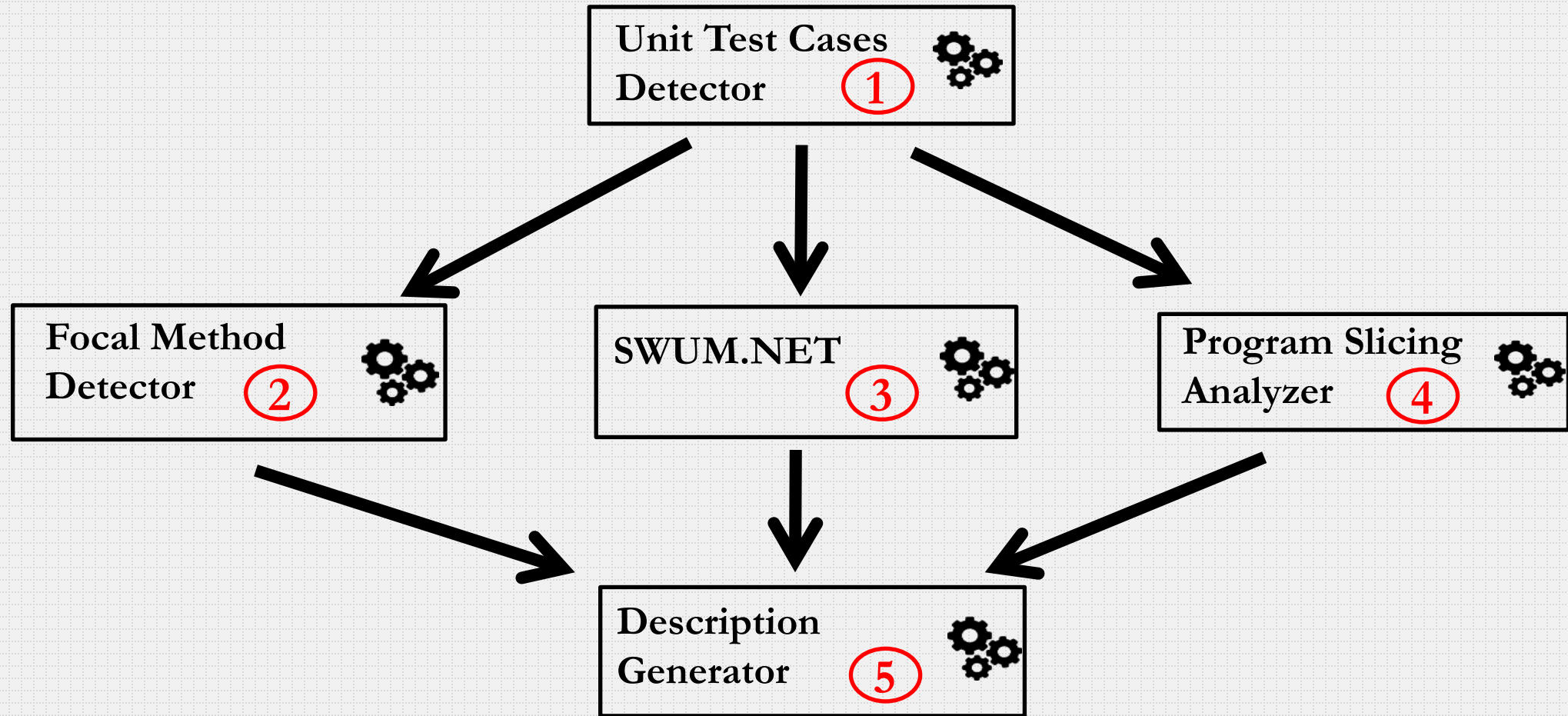
2) variable `i` through [slicing path](#)

3) variable `listLength` through [slicing path](#)

`listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)`

- Natural language descriptions
- Focal methods
- Assertion description
- Internal data dependencies

UnitTestScribe Architecture



1) Unit Tests Detector

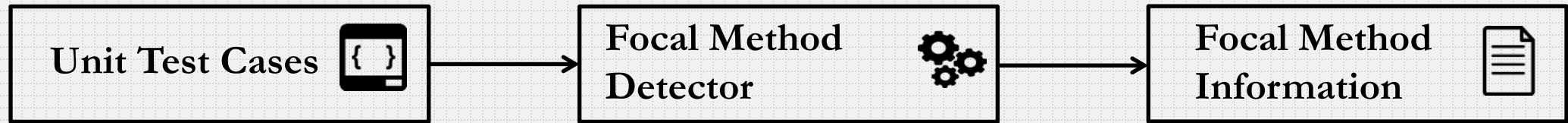


- Unit test annotations

[Test], [TestMethod], [TestCase], [Fact] etc.

- The test case detector detects unit tests based on the annotations

2) Focal Method Detector



- Focal method detection (Ghafari et al. SCAM'15)
- The last mutator/collaborator function(s) that modifies the variable(s) examined in a given assertion.

```
public ReturnResult (Metadata sut, ITabContext
context, View.Render.Message renderMessage){
    context.TabStore.Setup().Returns(true);
    context.TabStore.Setup().Returns(new List);
    var res = sut.GetData(context) as List;
    Assert.NotNull(res);
    Assert.NotEmpty(res);
}
```

3) SWUM.NET



- SWUM (Hill et al. ICSE'09) is an approach that can extract and tag NL phrases for source code.
- SWUM.NET by ABB

```
public SimpleApiClientFactory {  
    .....  
    public void ClearFromCache(ISimpleApiClient c){  
        .....  
    }  
}
```



“Clear simple api client from cache”

4) Variable Slicing Analyzer

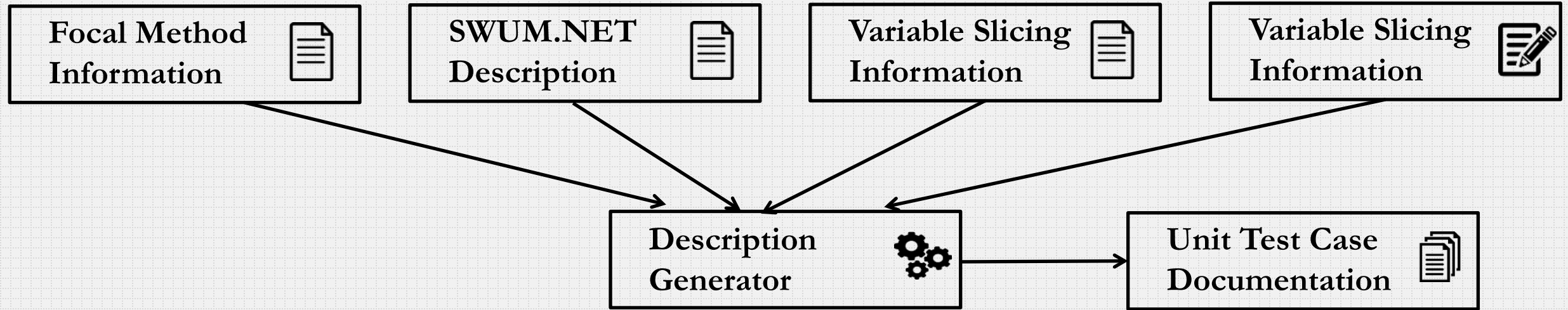


- Internal data dependencies for the variables in assertions

```
public AddWordsSeveralTimes(){  
    int listLength = 20;  
    int cooccurrenceCount = 3;  
    var words = GenerateRandomWordList(listLength);  
    for(int i = 0; i < cooccurrenceCount; i ++)  
        matrix.HandleCoOcurrentWordsSync(words);  
    for(int i = 0; i < listLength - 1; i ++){  
        var word1 = words.ElementAt(i);  
        var word2 = words.ElementAt(i + 1);  
        var count =  
            matrix.GetCoOccurrenceCount(word1, word2);  
        Assert.IsTrue(count > 0);  
    }  
}
```

listLength → words → word2 → count

5)Description Generator



- Using predefined templates
- Documentations in HTML format

UnitTestScribe HTML Report

UnitTestScribe report - Sando

Methods

- s.TextFileParserTest.ParseXAMLFile()
- s.TextFileParserTest.ParseXAMLFile2()
- s.XAMLFileParserTest.PerformanceTest()
- s.XAMLFileParserTest.TestLengthOfEachXamlElement()
- UnitTests.CamelIdSplitterTests.SplitTest()
- UnitTests.CamelIdSplitterTests.SplitTest_BadCamelCase()
- UnitTests.CamelIdSplitterTests.SplitTest_lowercase()
- UnitTests.CamelIdSplitterTests.SplitTest_uppercase()
- UnitTests.CamelIdSplitterTests.SplitTest_Uppercase()
- UnitTests.SwumDataRecordTests.QueryRecommendation()
- UnitTests.SwumDataRecordTests.TestRoundTrip()
- UnitTests.SwumDataRecordTests.TestRoundTrip_File()
- UnitTests.SwumManagerTests.TestAddSourceFile()
- UnitTests.SwumManagerTests.TestAddSourceFile_Cache()
- UnitTests.SwumManagerTests.TestAddSourceFile_Disk()
- UnitTests.SwumManagerTests.TestCacheRoundTrip()
- UnitTests.SwumManagerTests.TestConcurrentRead()
- UnitTests.SwumManagerTests.TestReadingRealCache()
- UnitTests.SwumManagerTests.TestRemoveSourceFile()
- UnitTests.SwumManagerTests.TestUpdateSourceFile()
- UnitTests.CodeSearcherFixture.PerformBasicSearch()
- UnitTests.CodeSearcherFixture.PerformBasicSearch2()
- UnitTests.CodeSearcherFixture.TestCreateCodeSearcher()
- UnitTests.CodeSearcherFixture.TestCreateCodeSearcher2()
- HighlightConverterTests.HandleTextWithHeadingAndTab()
- HighlightConverterTests.HighlightedWordAtBeginningTest()
- HighlightConverterTests.MakeSureEmptyLinesAreHandled()
- ArchViewControlTest.RemoveEscapesFromQuery()
- TranslatorTest.Translator_GetTranslationReturnsValidTranslation()

This unit test case method tests handle weird struct test .

This unit test case includes following focal methods:

- (1) `var elements = parser.Parse(WeirdStructFile);`(@line 150)
This focal method is related to assertions at [line 151](#)

This unit test case validates that:

- (1) `elements.Count == 2` is true.
`elements` is obtained from
1) variable `WeirdStructFile` through [slicing path](#).
- (2) `structElement` is not null.
`structElement` is obtained from
1) variable `pe` through [slicing path](#).
- (3) `"LangMenuItem"` is equal to `structElement.Name`.
- (4) `structId` is equal to `methodElement.ClassId`.
`structId` is obtained from
1) variable `pe` through [slicing path](#).
- (5) `hasStruct` is true.

281. Sando.Parser.UnitTests.CppParserTest.ParseCppConstructorTest

Code with line numbers

This unit test case method tests parse cpp constructor test .

This unit test case includes following focal methods:

- (1) `var elements = parser.Parse("TestFiles\Event.H.txt");`(@line 180)
This focal method is related to assertions at [line 181](#)

UnitTestScribe HTML Report

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

Code with line numbers

This unit test case method tests add words several times .

This unit test case includes following focal methods:

(1) var count = matrix.GetCoOccurrenceCount(word1,word2);(@line 47)
This focal method is related to assertions at [line 48](#)

This unit test case validates that:

(1) count > 0 is true.
count is obtained from

- 1) variable listLength through [slicing path](#)
- 2) variable i through [slicing path](#)
- 3) variable listLength through [slicing path](#)

listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)

- Natural language descriptions
- Assertion description

```
public AddWordsSeveralTimes(){  
    int listLength = 20;  
    int cooccurrenceCount = 3;  
    var words = GenerateRandomWordList(listLength);  
    for(int i = 0; i < cooccurrenceCount; i ++)  
        matrix.HandleCoOccurrenceWordsSync(words);  
    for(int i = 0; i < listLength - 1; i ++){  
        var word1 = words.ElementAt(i);  
        var word2 = words.ElementAt(i + 1);  
        var count =  
            matrix.GetCoOccurrenceCount(word1,word2);  
        Assert.IsTrue(count > 0);  
    }  
}
```

- Focal methods
- Internal data dependencies

UnitTestScribe HTML Report

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

Code with line numbers

🗨 This unit test case method tests add words several times .

📌 This unit test case includes following focal methods:

(1) var count = matrix.GetCoOccurrenceCount(word1,word2);(@line 47)
This focal method is related to assertions at [line 48](#)

📌 This unit test case validates that:

(1) count > 0 is true.
count is obtained from

- 1) variable listLength through [slicing path](#)
- 2) variable i through [slicing path](#)
- 3) variable listLength through [slicing path](#)

listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)

- Natural language descriptions
- Assertion description

```
public AddWordsSeveralTimes(){  
    int listLength = 20;  
    int cooccurrenceCount = 3;  
    var words = GenerateRandomWordList(listLength);  
    for(int i = 0; i < cooccurrenceCount; i ++)  
        matrix.HandleCoOccurrenceWordsSync(words);  
    for(int i = 0; i < listLength - 1; i ++){  
        var word1 = words.ElementAt(i);  
        var word2 = words.ElementAt(i + 1);  
        var count =  
            matrix.GetCoOccurrenceCount(word1,word2);  
        Assert.IsTrue(count > 0);  
    }  
}
```

- Focal methods
- Internal data dependencies

UnitTestScribe HTML Report

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

Code with line numbers

This unit test case method tests add words several times .

This unit test case includes following focal methods:

(1) var count = matrix.GetCoOccurrenceCount(word1,word2);(@line 47)
This focal method is related to assertions at [line 48](#)

This unit test case validates that:

(1) *count* > 0 is true.
count is obtained from

- 1) variable *listLength* through [slicing path](#)
- 2) variable *i* through [slicing path](#)
- 3) variable *listLength* through [slicing path](#)

listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)

- Natural language descriptions
- Assertion description

```
public AddWordsSeveralTimes(){
    int listLength = 20;
    int cooccurrenceCount = 3;
    var words = GenerateRandomWordList(listLength);
    for(int i = 0; i < cooccurrenceCount; i ++){
        matrix.HandleCoOccurrenceWordsSync(words);
    }
    for(int i = 0; i < listLength - 1; i ++){
        var word1 = words.ElementAt(i);
        var word2 = words.ElementAt(i + 1);
        var count =
            matrix.GetCoOccurrenceCount(word1,word2);
        Assert.IsTrue(count > 0);
    }
}
```

- Focal methods
- Internal data dependencies

UnitTestScribe HTML Report

15. Sando.Core.UnitTests.Tools.CoOccurrenceMatrixTests.AddWordsSeveralTimes

Code with line numbers

🗨 This unit test case method tests add words several times .

📌 This unit test case includes following focal methods:

(1) var count = matrix.GetCoOccurrenceCount(word1,word2);(@line 47)
This focal method is related to assertions at [line 48](#)

📌 This unit test case validates that:

(1) count > 0 is true.
count is obtained from

- 1) variable listLength through [slicing path](#)
- 2) variable i through [slicing path](#)
- 3) variable listLength through [slicing path](#)

listLength(@line 36) >>> words(@line 38) >>> word2(@line 46) >>> count(@line 47)

- Natural language descriptions
- Assertion description

```
public AddWordsSeveralTimes(){
    int listLength = 20;
    int cooccurrenceCount = 3;
    var words = GenerateRandomWordList(listLength);
    for(int i = 0; i < cooccurrenceCount; i ++){
        matrix.HandleCoOccurrenceWordsSync(words);
    }
    for(int i = 0; i < listLength - 1; i ++){
        var word1 = words.ElementAt(i);
        var word2 = words.ElementAt(i + 1);
        var count =
            matrix.GetCoOccurrenceCount(word1,word2);
        Assert.IsTrue(count > 0);
    }
}
```

- Focal methods
- Internal data dependencies

Empirical Study

- We evaluate UnitTestScribe on four open source systems:
 - 1) SrcML.NET, 2) Sando, 3) Glimpse, and 4) Google-api-dotnet
- We then measure the quality of descriptions generated by UnitTestScribe according to three criteria ([Sridhara et al. ASE'10](#); [Moreno et al. ICPC'13](#); [Cortes-Coy et al. SCAM'14](#)):
 - 1) Completeness 2) Conciseness 3) Expressiveness

Research Questions

RQ4. How **complete** are the descriptions generated by UnitTestScribe?

RQ5. How **concise** are the descriptions generated by UnitTestScribe?

RQ6. How **expressive** are the descriptions generated by UnitTestScribe?

RQ7. How important are **focal methods** and **program slicing** for understanding unit test cases?


RQ8. How well can UnitTestScribe help developers understand unit test cases?

Empirical Study

- Group 1 : **ABB**

7 researchers/interns from the ABB corporate research center

2 systems from ABB

- Group 2 : 

19 students/researchers from universities

2 systems from Github

Empirical Study



- Randomly selected **5** unit tests for each system
- Each participant evaluates **2** systems
- For each attribute (Completeness, Conciseness, and Expressiveness)
 - We had $5 \times 2 \times 7$ (participants) = **70** answers in group 1
 - We had $5 \times 2 \times 19$ (participants) = **190** answers in group 2

RQ4. How **complete** are the descriptions generated by UnitTestScribe?

ABB



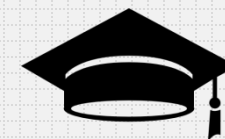
		Group 1 (Answers)	Group 2 (Answers)
Completeness	Does not miss any important info	33 (47.14%)	132 (69.47%)
	Misses some important info	28 (40.00%)	50 (26.32%)
	Misses the majority of the important info	9 (12.86%)	8 (4.21%)



Overall, UnitTestScribe is able to cover most essential information in most of the cases.

RQ5. How **concise** are the descriptions generated by UnitTestScribe?

ABB



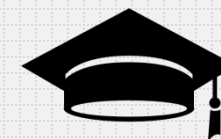
		Group 1 (Answers)	Group 2 (Answers)
Conciseness	Contains no redundant info	36 (51.43%)	100 (52.63%)
	Contains some redundant info	25 (35.71%)	77 (40.53%)
	Contains a lot of redundant info	9 (12.86%)	13 (6.84%)



Overall, UnitTestScribe can generate descriptions with little redundant information.

RQ6. How **expressive** are the descriptions generated by UnitTestScribe?

ABB



		Group 1 (Answers)	Group 2 (Answers)
Expressiveness	Is easy to read	43 (61.43%)	114(60.00%)
	Is somewhat readable	16(22.86%)	53(27.89%)
	Is hard to read	11(15.71%)	23(12.11%)



UnitTestScribe descriptions are easy to read in most of the cases.

RQ7. How important are focal methods and program slicing for understanding unit test cases?

ABB



Identifying focal method would help developers to understand the unit test.	Group 1 (Answers)	Group 2 (Answers)
Yes	7 (100.00%)	17 (89.00%)
No	0 (0%)	2 (11%)
Identifying slicing path would help developers to understand the unit test.	Group 1 (Answers)	Group 2 (Answers)
Yes	6 (86.00%)	13 (68.00%)
No	1 (14%)	6 (32%)



Focal methods and program slicing are important.

RQ7. How well can UnitTestScribe help developers understand unit test cases?

ABB



Are UnitTestScribe description useful for understanding the unit tests in the system?	Group 1 (Answers)	Group 2 (Answers)
Yes	4 (57%)	17 (89.00%)
No	3 (43%)	2 (11%)



For developers who are not familiar with an application, UnitTestScribe is very useful for understanding unit test methods.

Summary RQ4 – RQ8



Completeness



Conciseness



Expressiveness



Importance of focal methods and program slicing



Usefulness

Acknowledgment

- ABB colleagues: Vinay Augustine and Patrick Francis
- Anonymous reviewers
- All students, developers and researchers who responded to our survey



Summary

Lessons learnt from RQ1 – RQ3

(i) Documenting unit test cases is not a common practice

Documentation

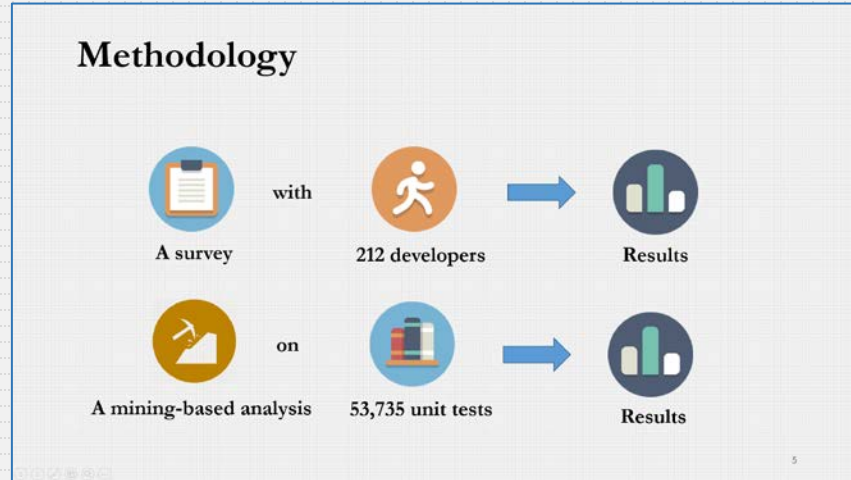
(ii) Developers do not update comments when changes are done to unit test cases

Automation

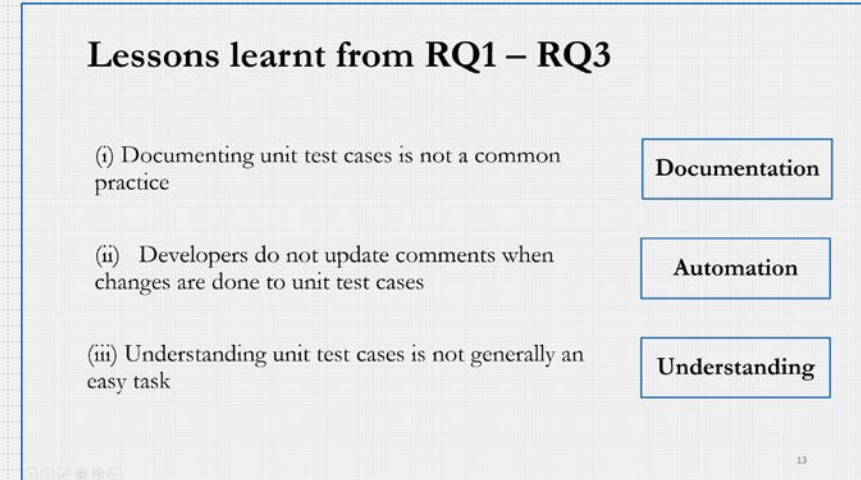
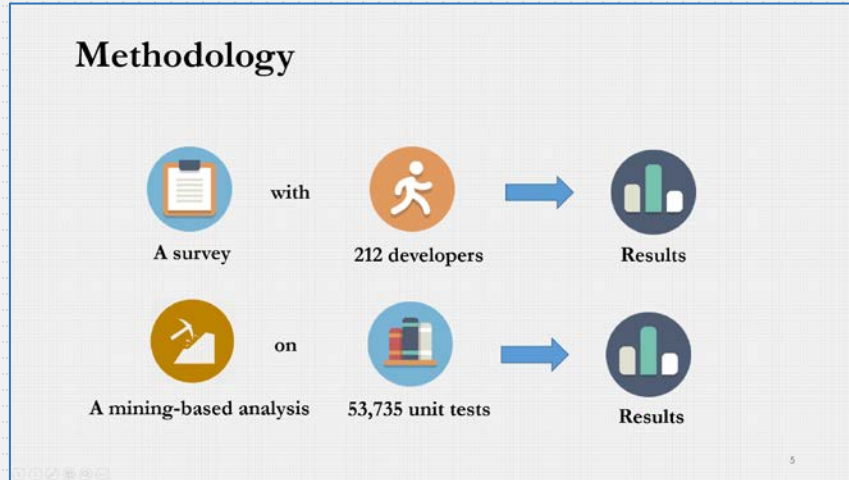
(iii) Understanding unit test cases is not generally an easy task

Understanding

Summary

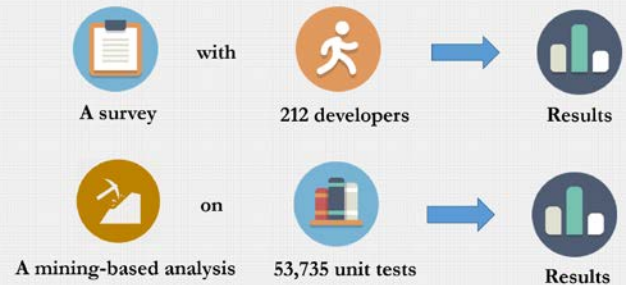


Summary



Summary

Methodology



Lessons learnt from RQ1 – RQ3

(i) Documenting unit test cases is not a common practice

Documentation

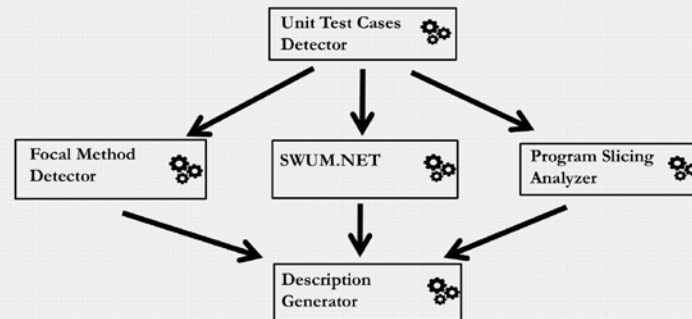
(ii) Developers do not update comments when changes are done to unit test cases

Automation

(iii) Understanding unit test cases is not generally an easy task

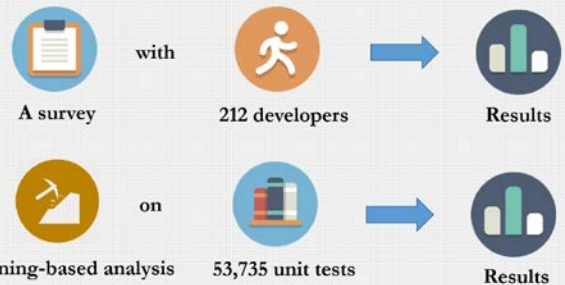
Understanding

UnitTestScribe Architecture



Summary

Methodology



Lessons learnt from RQ1 – RQ3

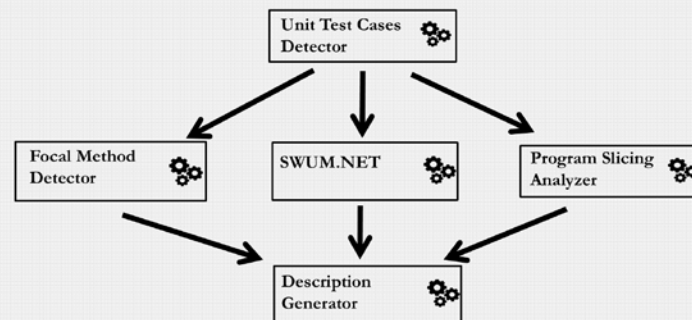
- (i) Documenting unit test cases is not a common practice
- (ii) Developers do not update comments when changes are done to unit test cases
- (iii) Understanding unit test cases is not generally an easy task

Documentation


Automation

Understanding

UnitTestScribe Architecture

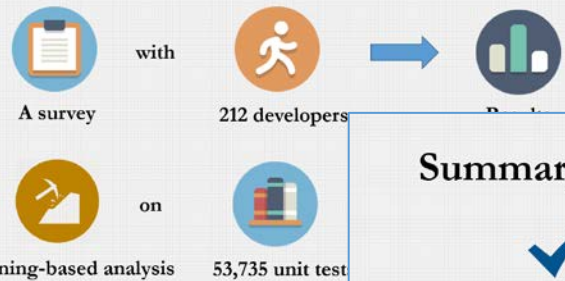


Empirical Study

- Group 1 : **ABB**
 - 7 researchers/interns from ABB corporate research center
 - 2 systems from ABB
- Group 2 : 
 - 19 students/researchers from universities
 - 2 systems from Github

Summary

Methodology



Lessons learnt from RQ1 – RQ3

(i) Documenting unit test cases is not a common practice

Documentation

(ii) Developers do not update comments when unit test cases

Automation

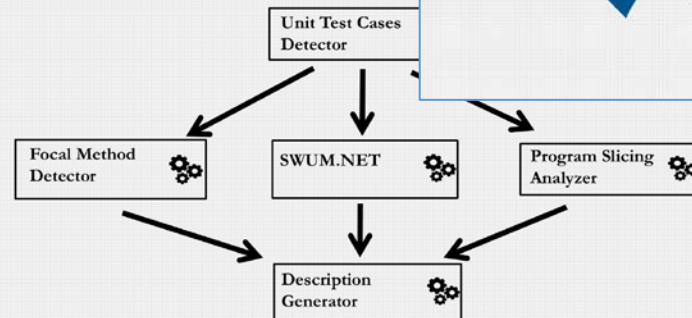
(iii) Documenting unit test cases is not generally an

Understanding

Summary RQ4 – RQ8

- ✓ Completeness
- ✓ Conciseness
- ✓ Expressiveness
- ✓ Importance of focal methods and program slicing
- ✓ Usefulness

UnitTestScribe Architecture



Study

interns from ABB corporate research center

2 systems from ABB

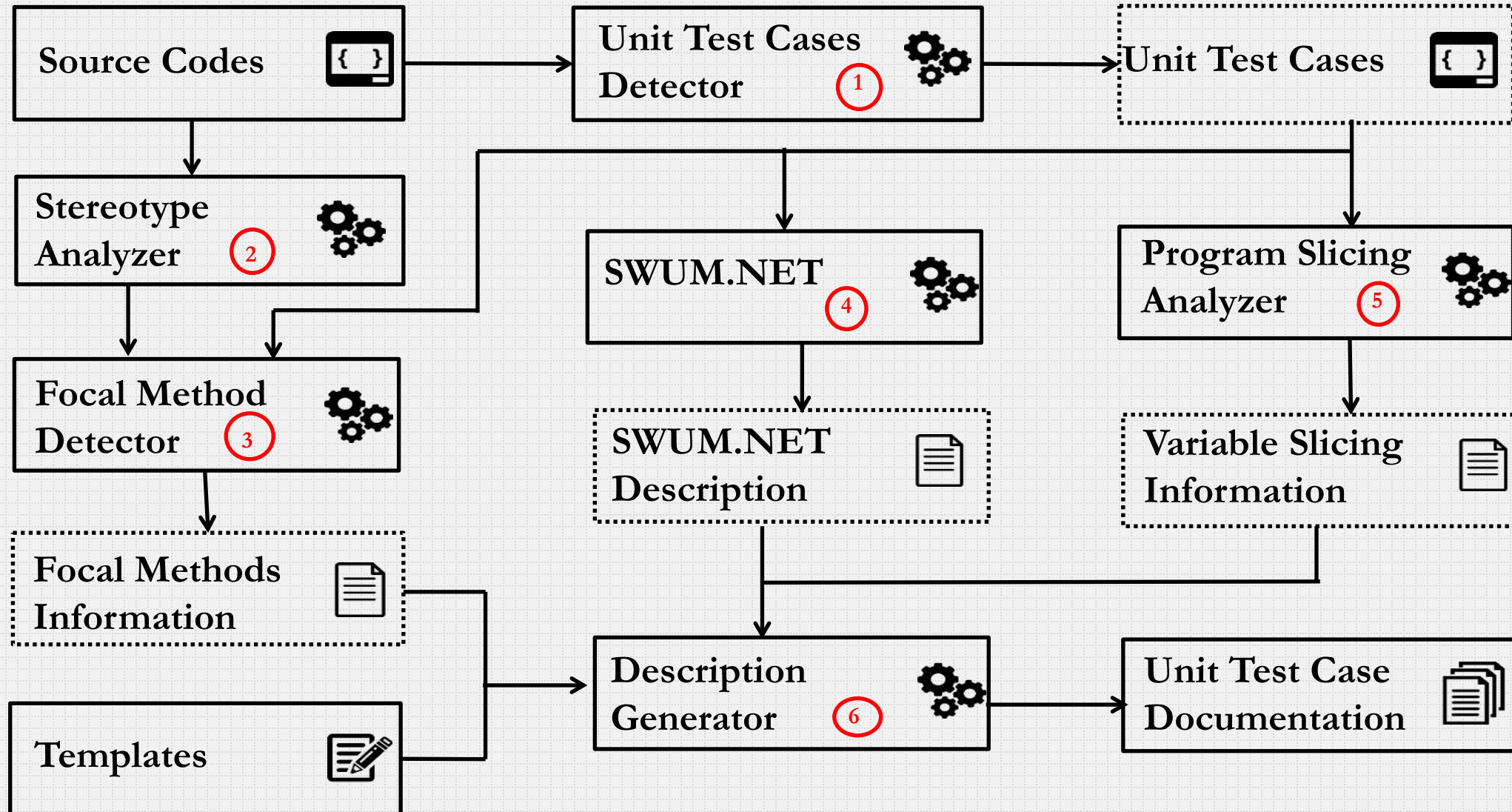
• Group 2 :

19 students/researchers from universities

2 systems from Github

BACKUP Slides...

UnitTestScribe Architecture



RQ7. How well can UnitTestScribe help developers understand unit test cases?

ABB



Are UnitTestScribe description useful for understanding the unit tests in the system?	Group 1	Group 2
Yes	4 (57%)	17 (89.00%)
No	3 (43%)	2 (11%)

"I see the SrcML.NET system, I know what's going on. Its usefulness drops off if you're talking to someone experienced with the code base, though. So I suppose this depends on who this is aimed at."

- From a participant in **group 1**

"It is useful if I am not familiar with an application."

- From a participant in **group 2**



For developers who are not familiar with an application, UnitTestScribe is very useful for understanding unit test methods

TABLE II
TAXONOMY OF METHOD STEREOTYPES PROPOSED BY DRAGAN ET AL.[23] WITH OUR PROPOSED MODIFICATIONS

Type	Category	Description	Modified rules
Getter	Accessor	Returns the value of a data member	No class field is changed && Return type is not void && Only return one class field.
Predicate	Accessor	Returns a Boolean result based on a data member(s)	No data member is changed && Return type is bool && Do not directly return any data member
Property	Accessor	Returns information about a data member	No data member is changed && Return type is not bool or not void. && Do not directly return any data member
Setter	Mutator	Changes the value of a data member	Only 1 data member is changed&&Return type is void or 0/1
Command	Mutator	Executes complex changes on data members	More than 1 class field is changed&&Return type is void or 0/1
Collaborator	Collaborator	Works on objects of classes different from the method	At least one of the method's parameters or local variables is an object Invokes external method(s)
Factory	Creator	Creates an object and returns it	Not returns primitive type Local && (A local variable is instantiated and returned Creates and returns a new object directly)

Accessor	Getter methods	Student.GetName()
	Predicate methods	Student.IsGraduate()
Mutator	Setter methods	Student.SetName(String name)
Collaborator		Classe.DropClass(Student st)

- Focal method detection (Ghafari et al. SCAM'15)

Algorithm 1: An Algorithm for Focal Method Detection

Input: MethodDefinition *m*, AssertionStatement *assert*

Output: Set<FunctionCall> *fmSet*

```
1 begin
2   fmSet ← new Set<FunctionCall>()
3   v ← GetEvaluatedVariable (assert)
4   queue.Push(v)
5   while queue.Size > 0 do
6     v ← queue.Pop()
7     decl_stmt_v ← FindDeclaration (m, v)
8     b ← IsExternalObject (decl_stmt_v)
9     if b == true then
10      vSet ← GetRelatedVariables (m, v)
11      queue.PushAll(vSet)
12    else
13      call ← FindTheLastMutatorCall (m, v)
14      fmSet.Add(call)
15  return fmSet
```

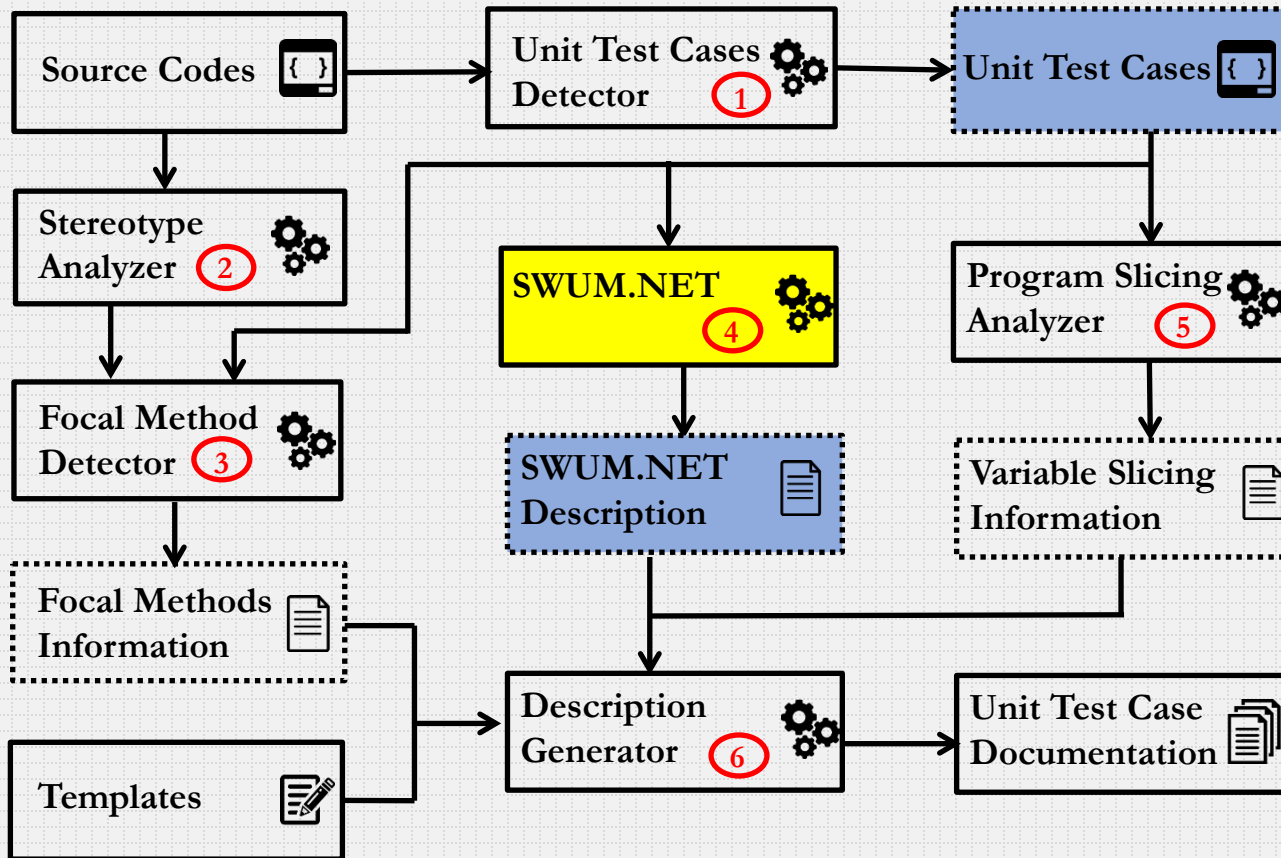
- The last mutator/collaborator function(s) that modifies the variable(s) used as arguments to a given assertion call.

```
public void foo(){
    Student a = new Student();
    a.setName("myname")
    assertTrue(a.Name == "myname");
}
```

```
public void foo(){
    Student a = new Student();
    a.setAge(28);
    int age = a.getAge();
    assertTrue(age == 28);
}
```

- Kamimura and Murphy ICPC'13. The approach identified the focal method based on how many times the test method invokes the function.

SWUM.NET



- SWUM (Hill et al.) is an approach which can extract and tag NL phrases for source codes.
- The SWUM.NET tool implemented by ABB in c#

```
public SimpleApiClientFactory {  
    .....  
    public void ClearFromCache(ISimpleApiClient c){  
        .....  
    }  
}
```



Clear simple api client from cache

<Action><Theme><Preposition><SecondObject>