When and How Using Structural Information to Improve IR-based Traceability Recovery

Annibale Panichella[1], Collin McMillan[2], Evan Moritz[3], Davide Palmieri[4], Rocco Oliveto[4], Denys Poshyvaniyk[3], Andrea De Lucia[1]

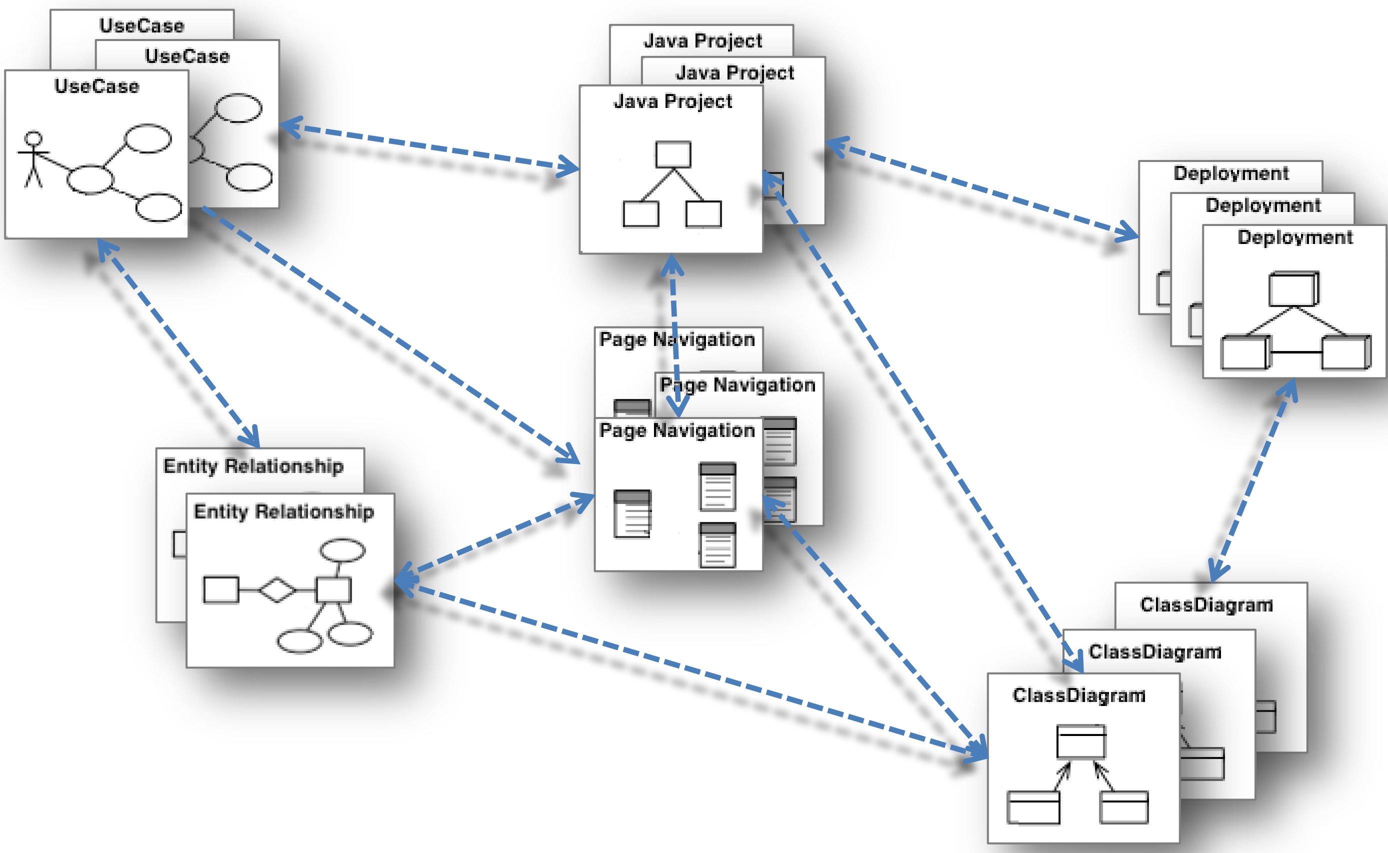
[1] Software Engineering Lab , University of Salerno, Italy

[2] University of Notre Dame, Notre Dame, USA

[3] The College of William and Mary, Williamsburg, USA

[2] University of Molise, Pesche (IS), Italiy

Traceability Recovery is the ability to describe and follow the artifacts life-cicle

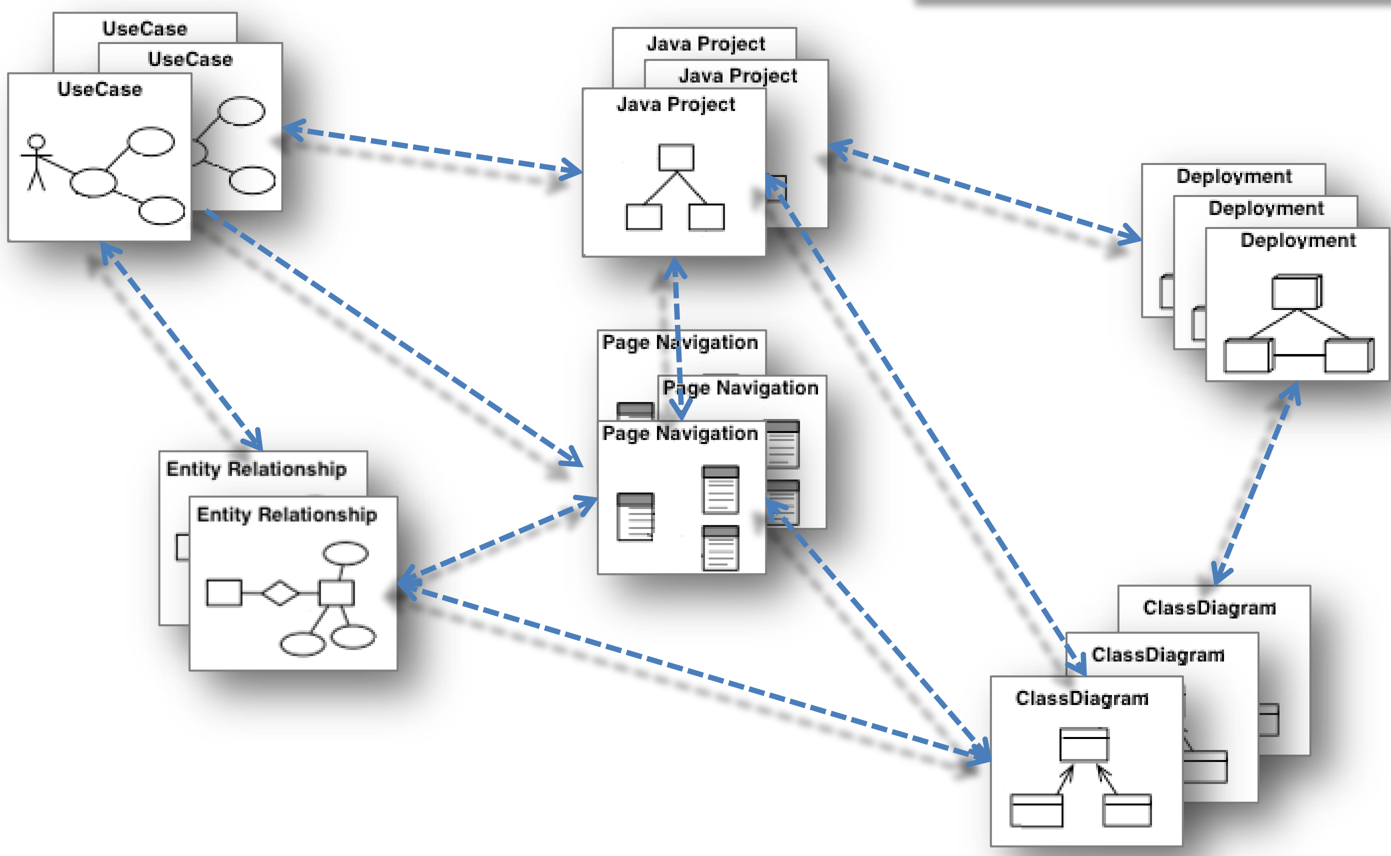


Traceability Recovery is the ability to describe and follow the artifacts life-cicle

Imapct Analysis

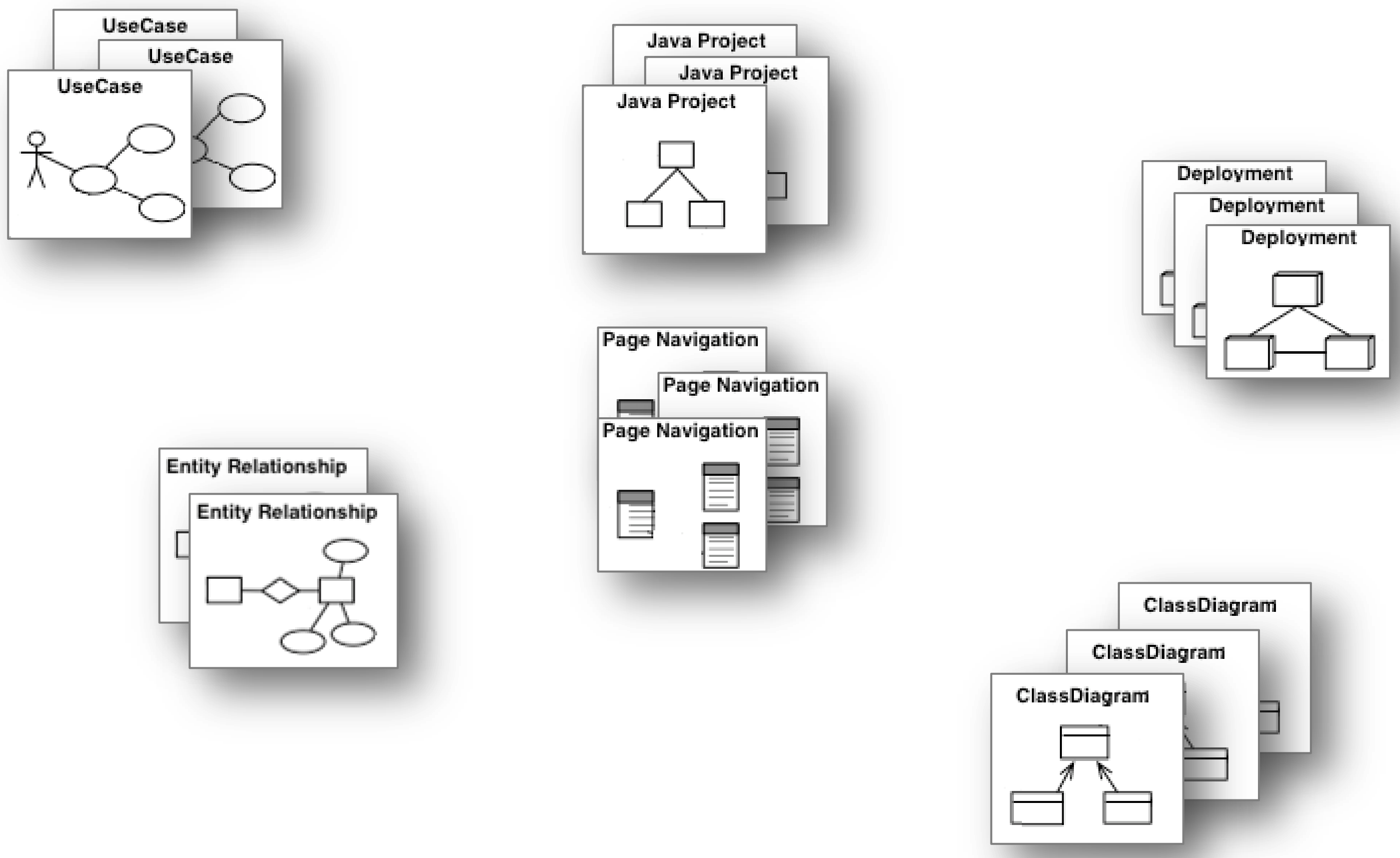
Program Comprehension

Requirements tracing



Traceability in practice

Traceability information is still not commonplace in software projects!



Information Retrieval

IR-based Traceability Recovery

Use Case	*Insert Laboratory Data*
Description	*The user inserts the data of a specific laboratory*
Events	*1. The user opens the Laboratory GUI* *2. The user inserts the laboratory data* . . .

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUI Laboratory Data {
private jFrame window;
private jButton insert;
...

public GUI Laboratory Data(){
window = ne JFrame();
insert = new JButton();
...
}

...
}
```

IR-based Traceability Recovery

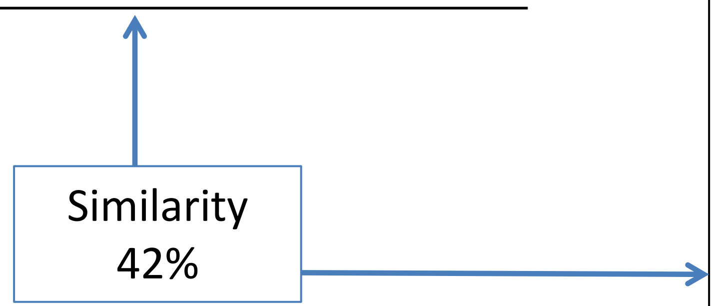
Use Case	Insert Laboratory Data
Description	The user **inserts** the **data** of a specific **laboratory**
Events	1. The user opens the **Laboratory GUI** 2. The user **inserts** the **laboratory data** . . .

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUI Laboratory Data {
private jFrame window;
private jButton insert;
...

public GUI Laboratory Data(){
window = new JFrame();
insert = new JButton();
...
}


...
}
```

IR-based Traceability Recovery

Use Case	Insert Laboratory Data
Description	The user inserts the data of a specific laboratory
Events	1. The user opens the Laboratory GUI 2. The user inserts the laboratory data . . .



GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUI Laboratory Data {
private jFrame window;
private jButton insert;
...

public GUI Laboratory Data(){
window = new JFrame();
insert = new JButton();
...
}


...
}
```

IR-based Traceability Recovery

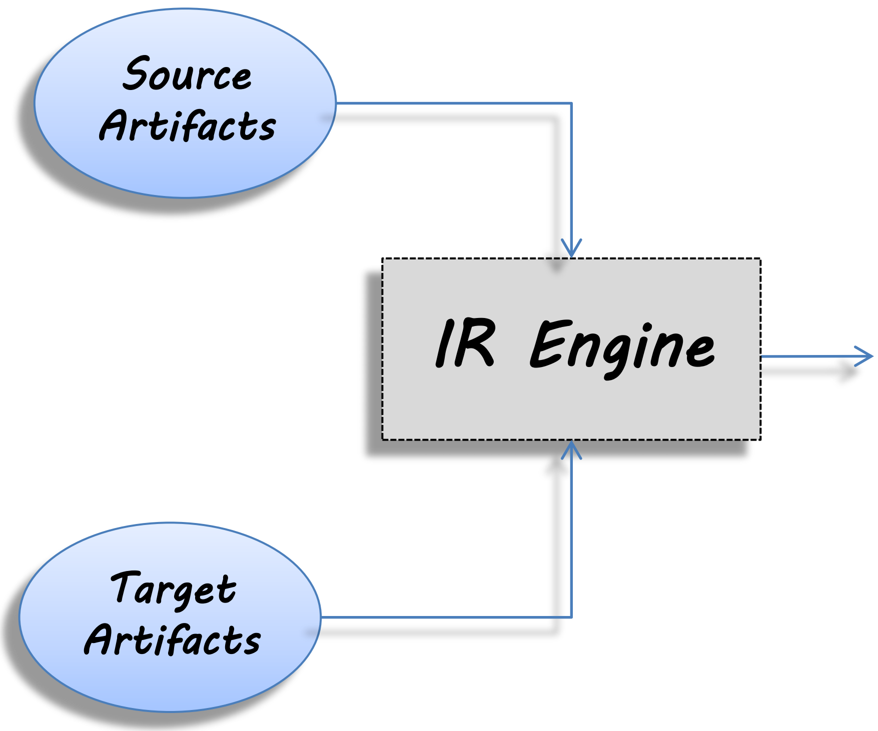
Use Case	*Insert Laboratory Data*
Description	The user **inserts** the **data** of a specific **laboratory**
Events	1. The user opens the **Laboratory GUI** 2. The user **inserts** the **laboratory data** . . .

True Link

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUI Laboratory Data {
private jFrame window;
private jButton insert;
...

public GUI Laboratory Data(){
window = new JFrame();
insert = new JButton();
...
}

...
}
```

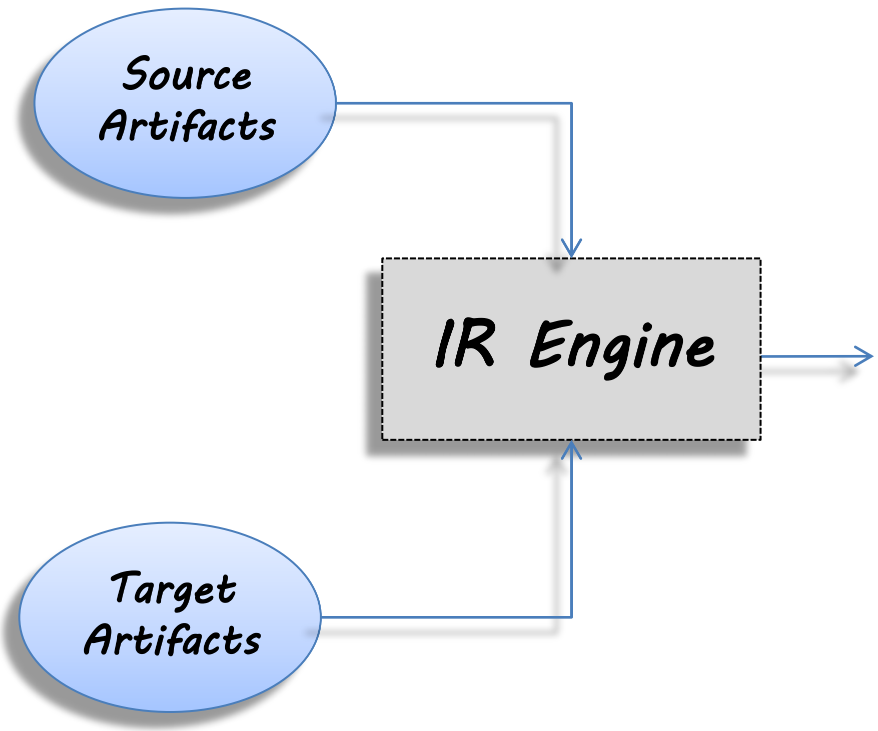
IR-based Traceability Recovery



Candidate Links

Source_1	Target_2	95.4%
Source_3	Target_4	92.1%
Source_1	Target_1	85.6%
Source_2	Target_2	83.2%
Source_3	Target_3	81.2%
Source_1	Target_3	79.0%
Source_3	Target_2	77.5%
Source_2	Target_4	64.3%
Source_2	Target_3	53.2%

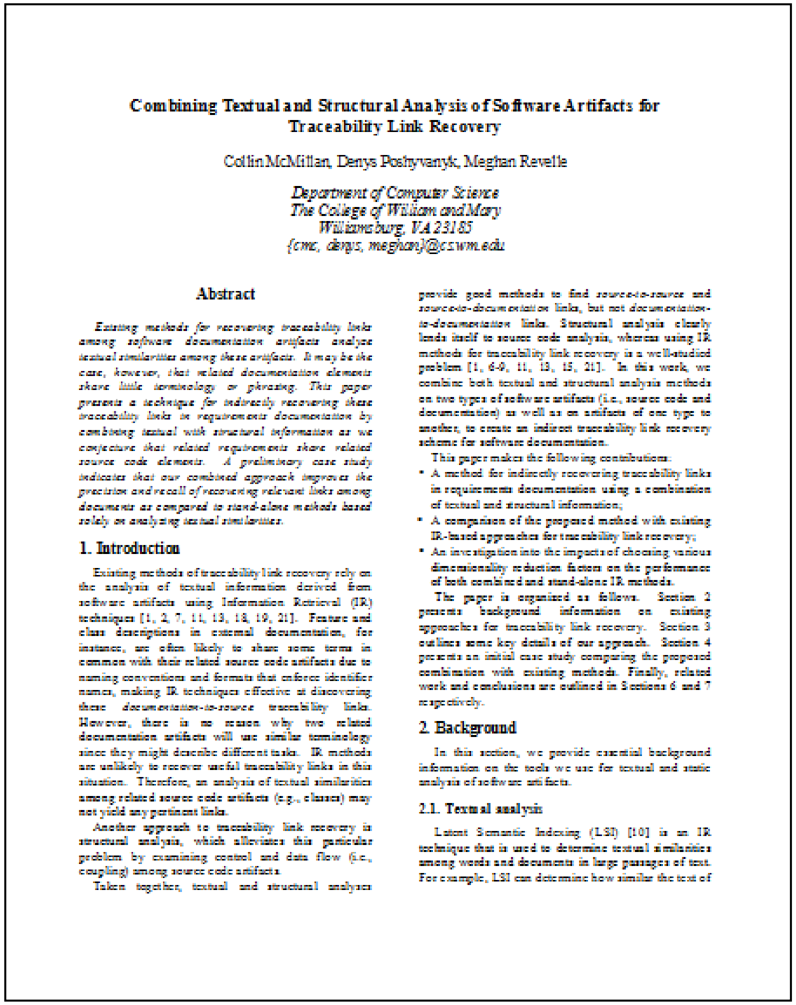
IR-based Traceability Recovery



Candidate Links

Source_1	Target_2	95.4%
Source_3	Target_4	92.1%
Source_1	Target_1	85.6%
Source_2	Target_2	83.2%
Source_3	Target_3	81.2%
Source_1	Target_3	79.0%
Source_3	Target_2	77.5%
Source_2	Target_4	64.3%
Source_2	Target_3	53.2%

Structural Information and Traceability



Combining Textual and Structural Analysis of Software Artifacts for Traceability Link Recovery

Collin McMillan, Denys Poshyvanyk, Meghan Revelle

Department of Computer Science
The College of William and Mary
Williamsburg, VA 23185
{cmc, denys, meghan}@cs.wm.edu

Abstract

Existing methods for recovering traceability links among software documentation artifacts analyze textual similarities among these artifacts. It may be the case, however, that related documentation elements share little terminology or phrasing. This paper presents a technique for indirectly recovering these traceability links in requirements documentation by combining textual with structural information as we conjecture that related requirements share related source code elements. A preliminary case study indicates that our combined approach improves the precision and recall of recovering relevant links among documents as compared to stand-alone methods based solely on analyzing textual similarities.

1. Introduction

Existing methods of traceability link recovery rely on the analysis of textual information derived from software artifacts using Information Retrieval (IR) techniques [1, 2, 7, 11, 13, 18, 19, 21]. Feature and class descriptions in external documentation, for instance, are often likely to share some terms in common with their related source code artifacts due to naming conventions and formats that enforce identifier names, making IR techniques effective at discovering these *documentation-to-source* traceability links. However, there is no reason why two related documentation artifacts will use similar terminology since they might describe different tasks. IR methods are unlikely to recover useful traceability links in this situation. Therefore, an analysis of textual similarities among related source code artifacts (e.g., classes) may not yield any pertinent links.

Another approach to traceability link recovery is structural analysis, which alleviates this particular problem by examining control and data flow (i.e., coupling) among source code artifacts.

Taken together, textual and structural analyses provide good methods to find *source-to-source* and *source-to-documentation* links, but not *documentation-to-documentation* links. Structural analysis clearly lends itself to source code analysis, whereas using IR methods for traceability link recovery is a well-studied problem [1, 6-9, 11, 13, 15, 21]. In this work, we combine both textual and structural analysis methods on two types of software artifacts (i.e., source code and documentation) as well as on artifacts of one type to another, to create an indirect traceability link recovery scheme for software documentation.

This paper makes the following contributions:

- A method for indirectly recovering traceability links in requirements documentation using a combination of textual and structural information;
- A comparison of the proposed method with existing IR-based approaches for traceability link recovery;
- An investigation into the impacts of choosing various dimensionality reduction factors on the performance of both combined and stand-alone IR methods.

The paper is organized as follows. Section 2 presents background information on existing approaches for traceability link recovery. Section 3 outlines some key details of our approach. Section 4 presents an initial case study comparing the proposed combination with existing methods. Finally, related work and conclusions are outlined in Sections 6 and 7 respectively.

2. Background

In this section, we provide essential background information on the tools we use for textual and static analysis of software artifacts.

2.1. Textual analysis

Latent Semantic Indexing (LSI) [10] is an IR technique that is used to determine textual similarities among words and documents in large passages of text. For example, LSI can determine how similar the text of

Combining Textual and Structural Analysis of Software Artifacts for Traceability Link Recovery - Collin McMillan, Denys Poshyvanyk, Meghan Revelle
TEFSE 2009

Structural Information and Traceability

Use Case	*Insert Laboratory Data*
Description	*The user inserts the data of a specific laboratory*
Events	*1. The user opens the Laboratory GUI* *2. The user inserts the laboratory data* . . .

linked ↔

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUILaboratoryData {
private jFrame window;
private jButton insert;
...

public GUILa
window = ne
insert = new
...
}

...
}
```

Similarity = 42%

Structural Information and Traceability

Use Case	*Insert **Laboratory** Data*
Description	*The user inserts the data of a specific **laboratory***
Events	*1. The user opens the **Laboratory** GUI* *2. The user inserts the **laboratory** data* . . .

linked

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUILaboratoryData {
private jFrame window;
private jButton insert;
...

public GUILa
window = ne
insert = new
...
}

...
}
```

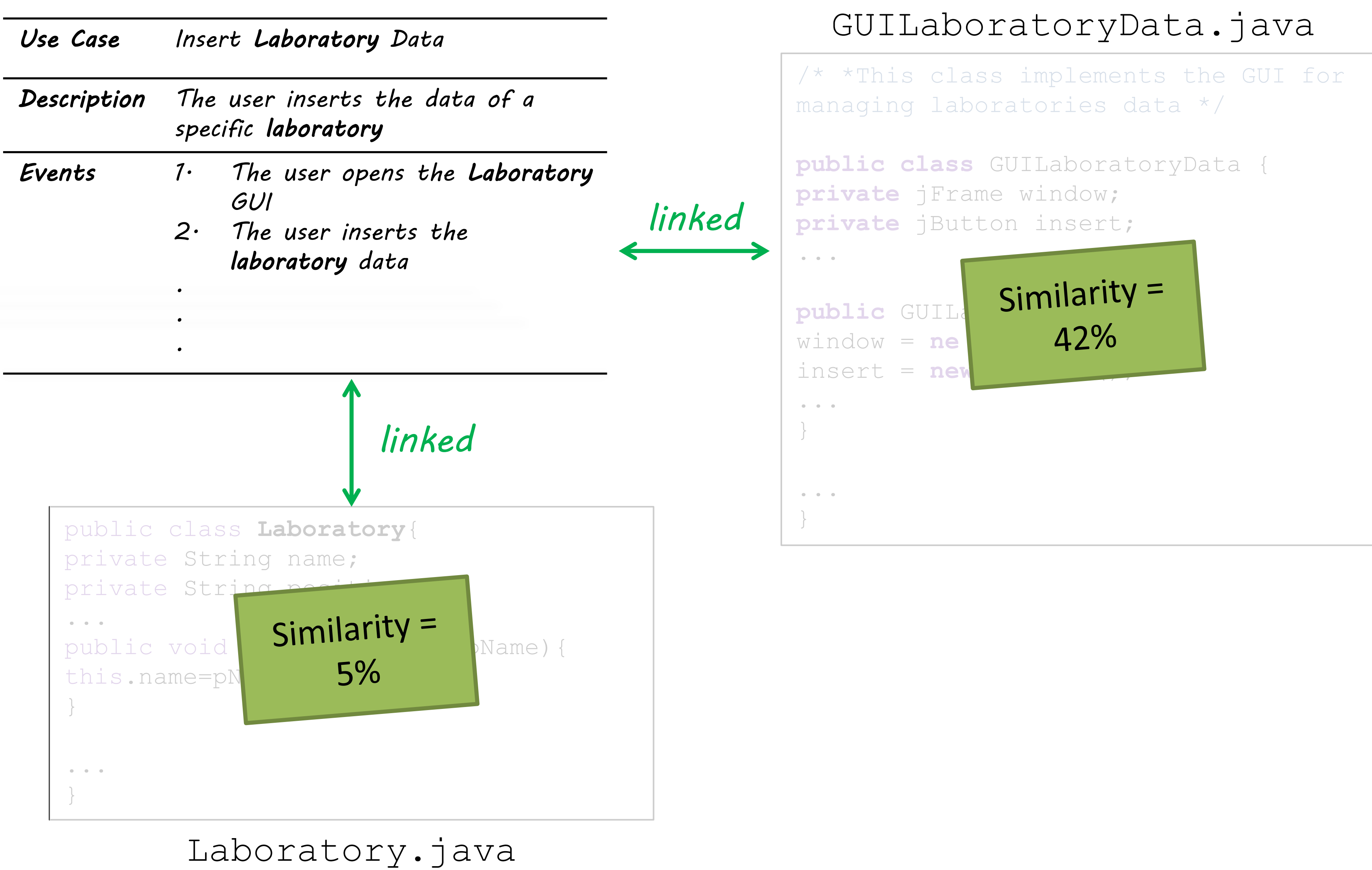
Similarity = 42%

linked

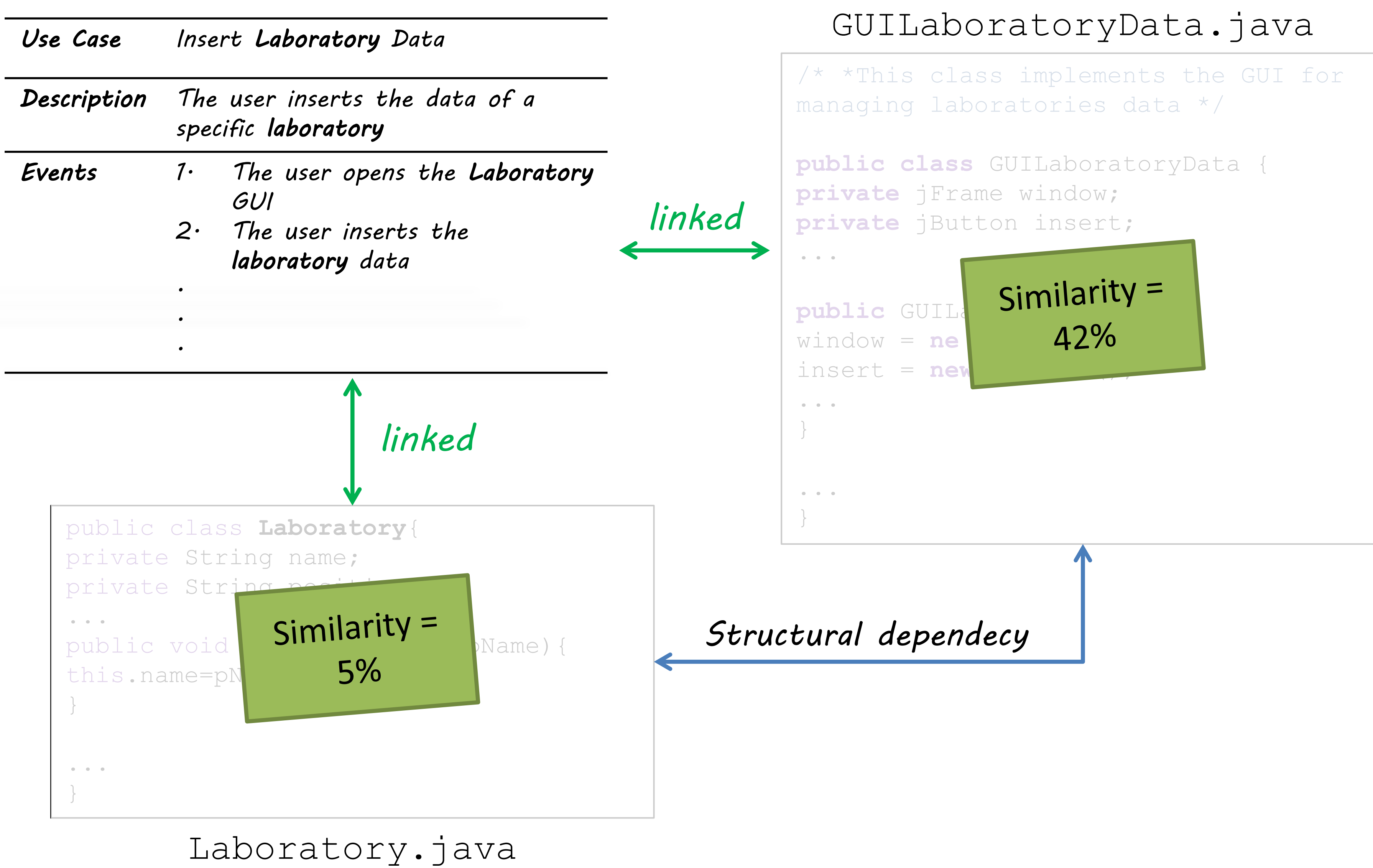
```
public class Laboratory{
private String name;
private String position;
...
public void setName(String pName){
this.name=pName;
}

...
}
```

Laboratory.java

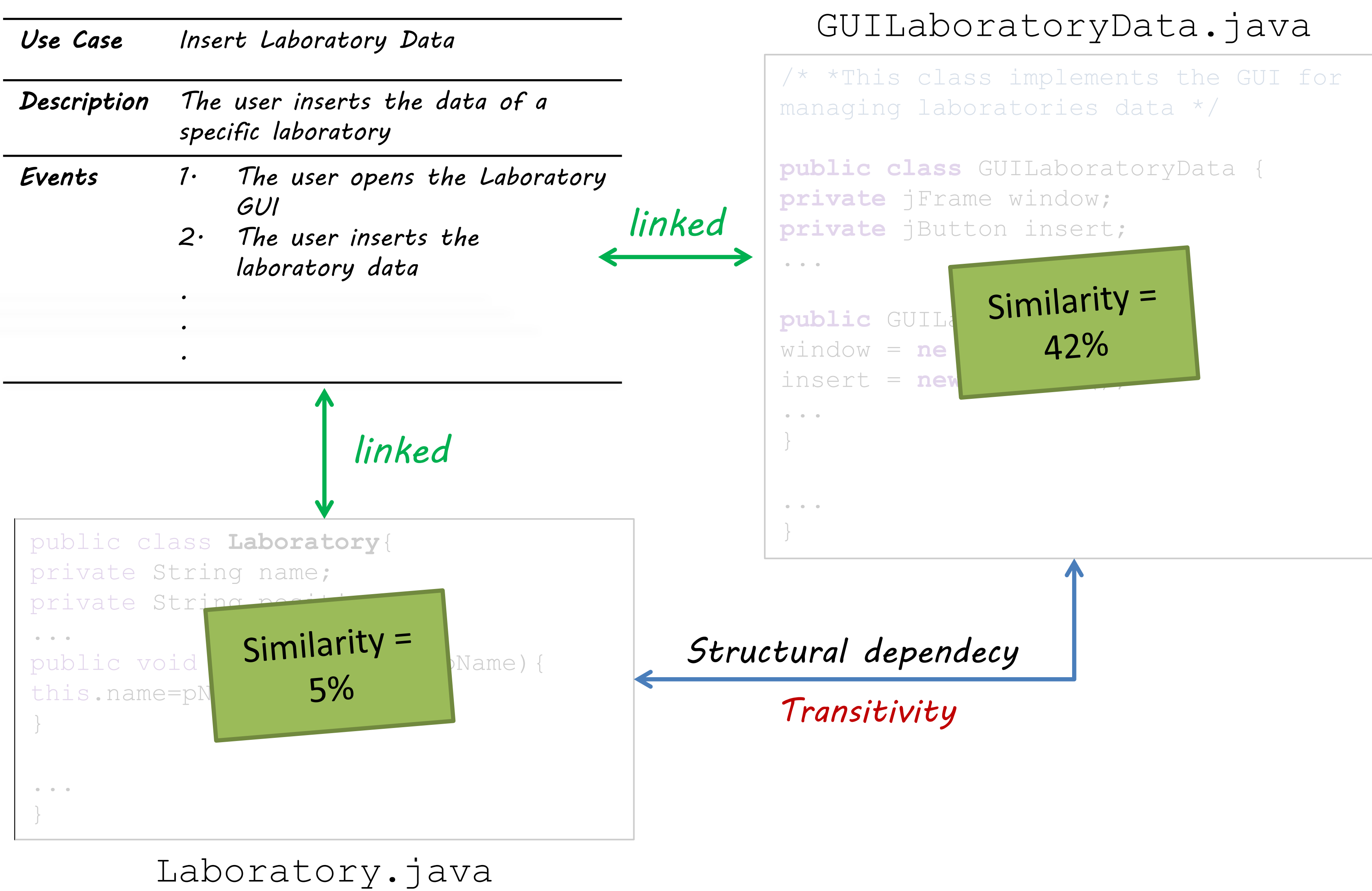
Structural Information and Traceability



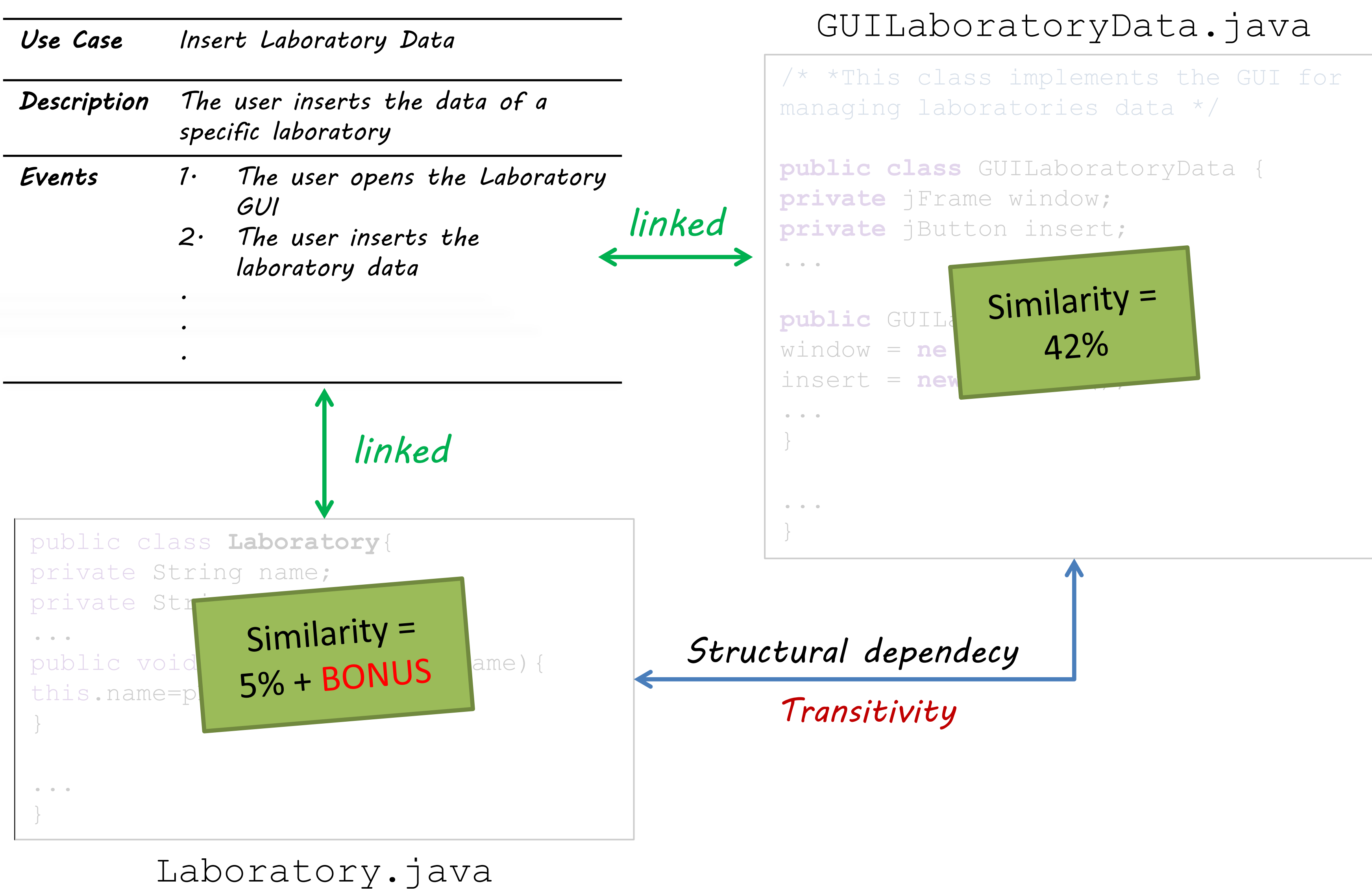
Structural Information and Traceability



Structural Information and Traceability



Structural Information and Traceability



Open Issues

1) The choice of the bonus value is cricual
- Different systems require different bonus
- Different IR methods require different bonus

Open Issues

1) The choice of the bonus value is cricual
   - Different systems require different bonus
   - Different IR methods require different bonus

2) When applying the bonus?
   - The transitivity property does not always hold

Example

Use Case	***Insert** Laboratory **Data***
Description	*The user **inserts** the **data** of a specific laboratory*
Events	*1. The user opens the Laboratory **GUI*** *2. The user **inserts** the laboratory **data*** . . .

Not linked

GUIDoctorData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUIDoctorData {
private jFrame window;
private jButton insert;
...

public GUIDoctorData(){
window = ne JFrame();
insert = new JButton();
...
}

...
}
```

Example

Use Case	Insert **Laboratory** Data
Description	The user inserts the data of a specific **laboratory**
Events	1. The user opens the **Laboratory** GUI 2. The user inserts the **laboratory** data . . .

Not linked

GUIDoctorData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUILaboratoryData {
private jFrame window;
private jButton insert;
...

public GUILa
window = ne
insert = new
...
}

...
}
```

Similarity = 42%

Example

Use Case	Insert **Laboratory** Data
Description	The user inserts the data of a specific **laboratory**
Events	1. The user opens the **Laboratory** GUI 2. The user inserts the **laboratory** data . . .

Not linked

GUIDoctorData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUILaboratoryData {
private jFrame window;
private jButton insert;
...

public GUILa
window = ne
insert = new
...
}

...
}
```

Similarity = 42%

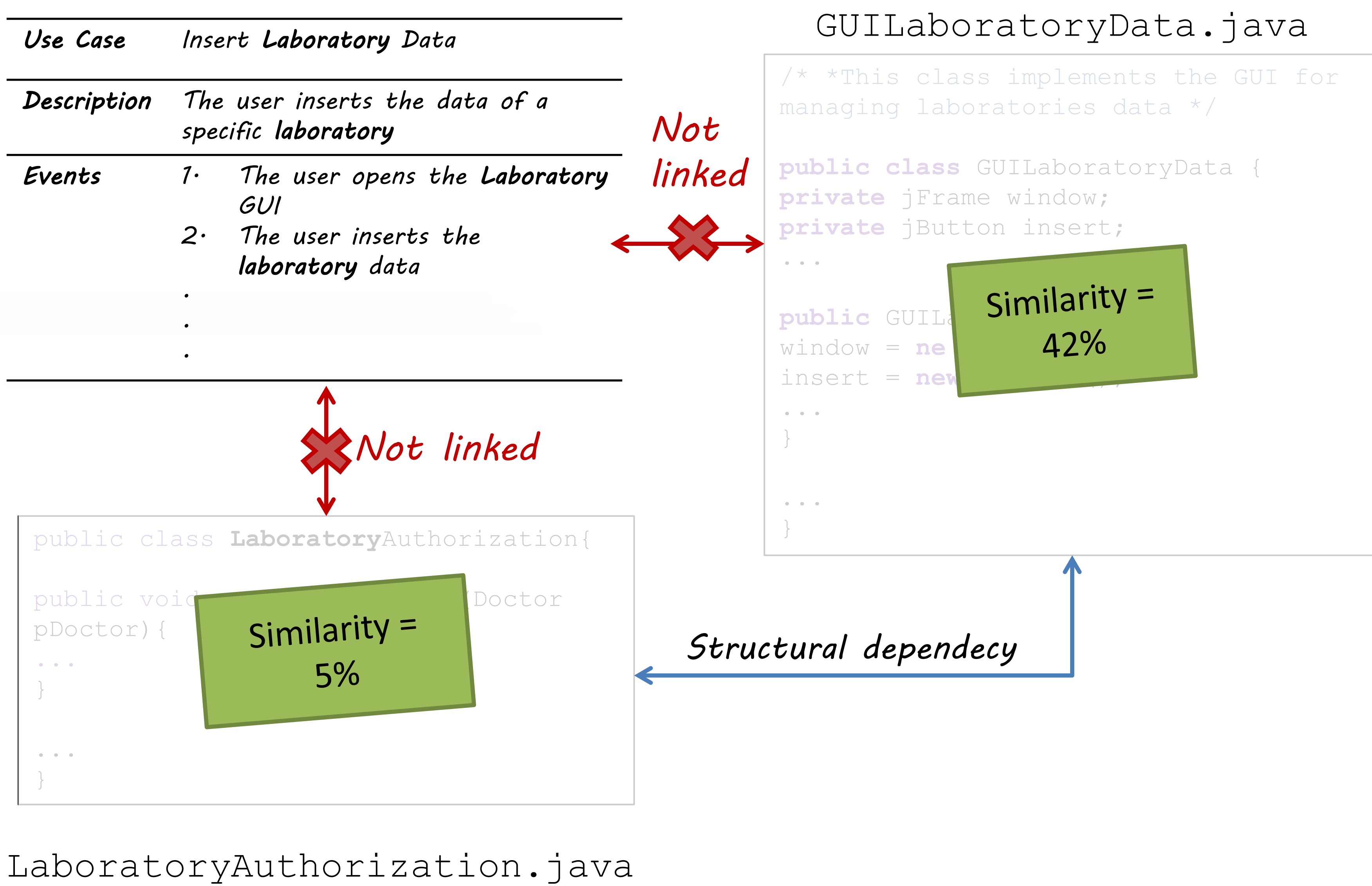
Not linked

```
public class Authorization{

public void setAuthorization(Doctor
pDoctor, Laboratory pLab){
...
}

...
}
```

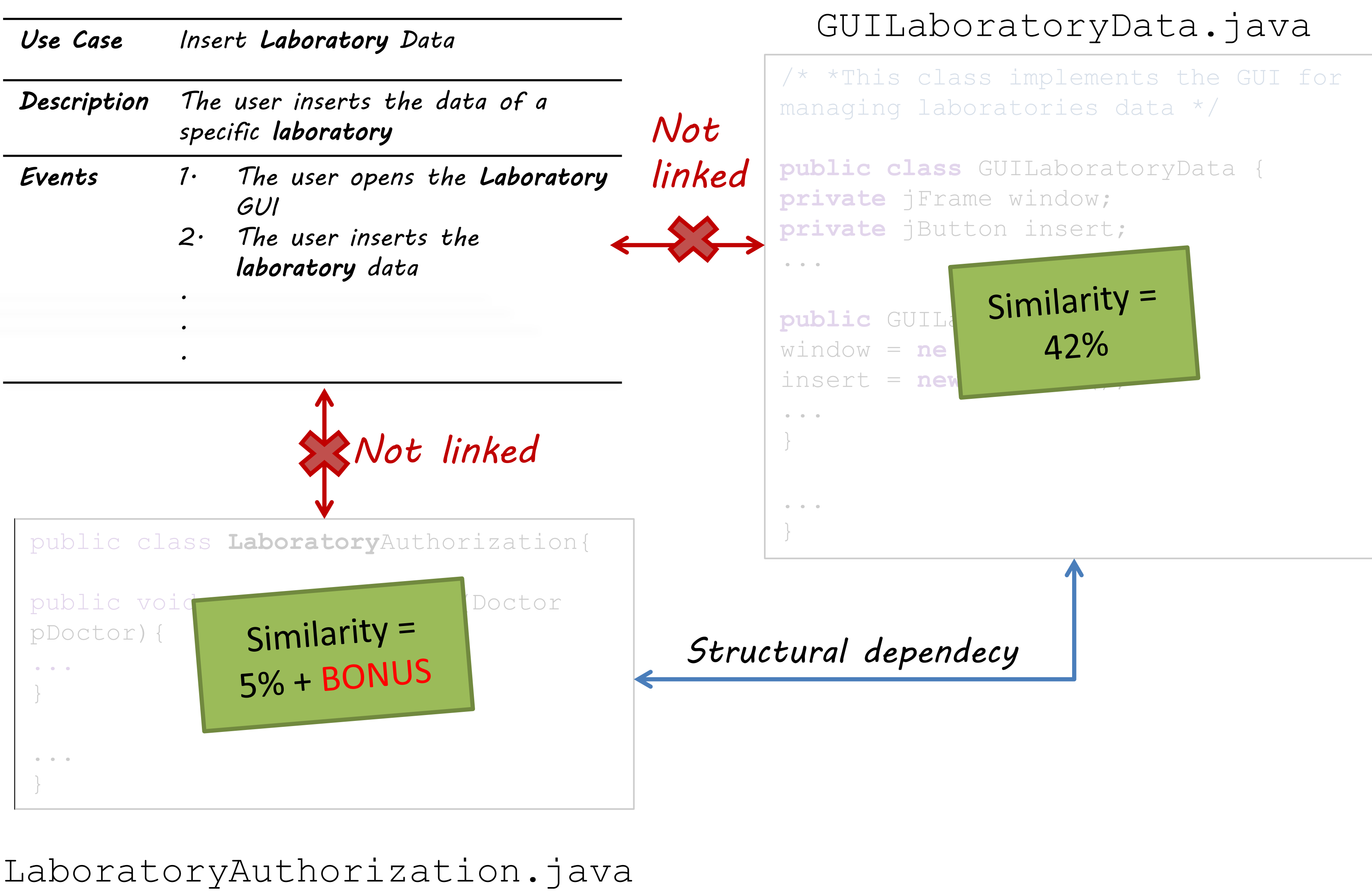
Autorization.java

Example

Use Case	Insert **Laboratory** Data
Description	The user inserts the data of a specific **laboratory**
Events	1. The user opens the **Laboratory** GUI 2. The user inserts the **laboratory** data . . .



Example



Use Case
Insert Laboratory Data
Description
The user inserts the data of a specific laboratory
Events
1. The user opens the Laboratory GUI
2. The user inserts the laboratory data
.
.
.
Not linked
Not linked
GUILaboratoryData.java
/* *This class implements the GUI for managing laboratories data */
public class GUILaboratoryData {
private jFrame window;
private jButton insert;
...
public GUILa
window = ne
insert = new
...
}
...
}
Similarity = 42%
public class LaboratoryAuthorization{
public void
Doctor pDoctor){
...
}
...
}
Similarity = 5% + BONUS
Structural dependecy
LaboratoryAuthorization.java

Example

Use Case	Insert **Laboratory** Data
Description	The user inserts the data of a specific **laboratory**
Events	1. The user opens the **Laboratory** GUI

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUILaboratoryData {
private jFrame window;
private jButton insert;
```

linked

Structural information is not always useful

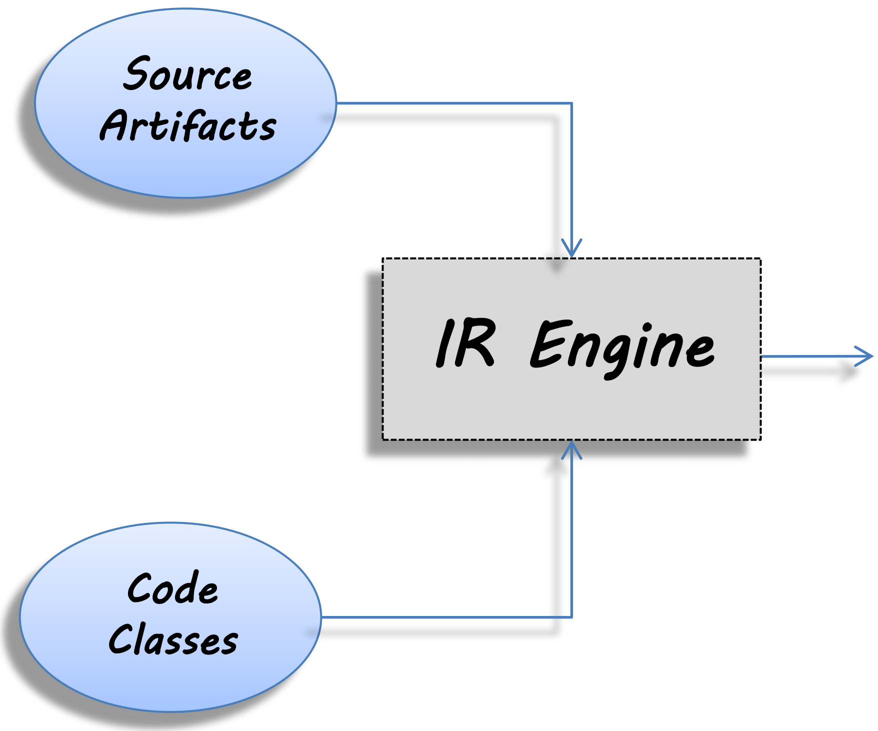
```
public class LaboratoryAuthorization{

public void setAuthorization(Doctor
pDoctor){
...
}

...
}
```

Structural dependecy

Laboratory.java

Approach 1: Optimistic Combination (O-CSTI)

Step 1: traditional IR process

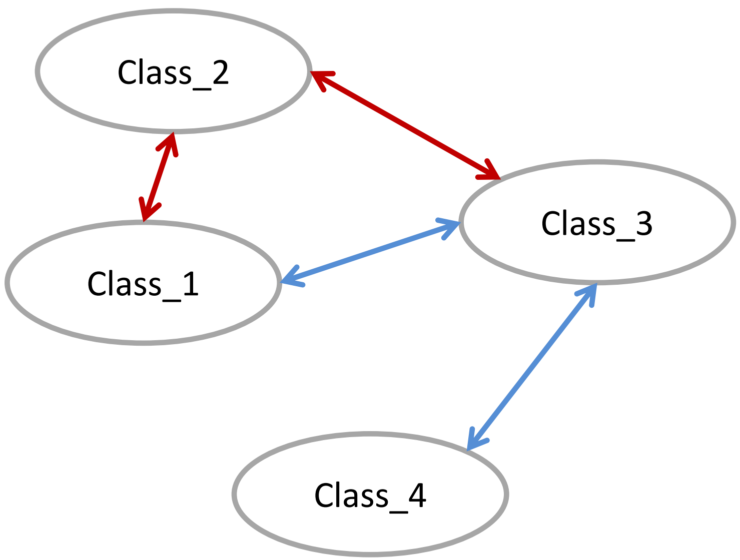


Candidate Links

Source_1	Class_2	95.4%
Source_3	Class_4	92.1%
Source_1	Class_1	85.6%
Source_2	Class_2	83.2%
Source_3	Class_3	81.2%
Source_1	Class_3	79.0%
Source_3	Class_2	77.5%
Source_2	Class_4	64.3%
Source_2	Class_3	53.2%

Approach 1: Optimistic Combination (O-CSTI)

Step 2: applying bonus to all the links

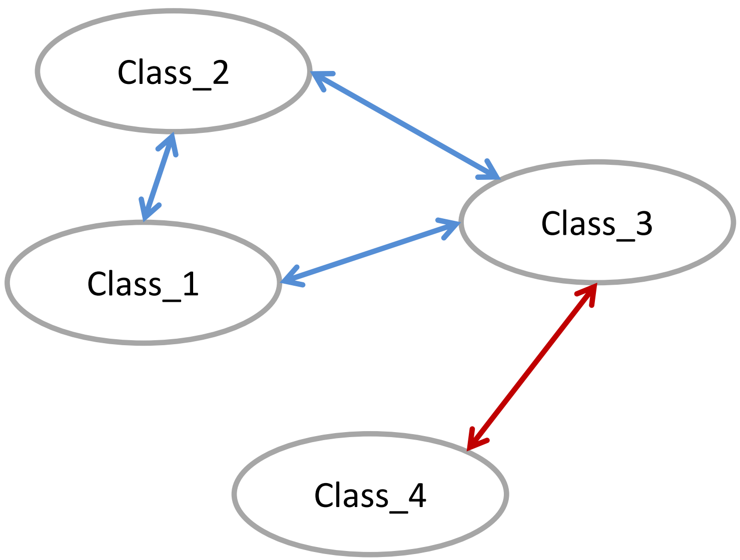


Candidate Links

Source_1	**Class_2**	**95.4%**	
Source_3	Class_4	92.1%	
Source_1	Class_1	85.6%	+ Bonus
Source_2	Class_2	83.2%	
Source_3	Class_3	81.2%	
Source_1	Class_3	79.0%	+ Bonus
Source_3	Class_2	77.5%	
Source_2	Class_4	64.3%	
Source_3	Class_4	53.2%	

Approach 1: Optimistic Combination (O-CSTI)

Step 2: applying bonus to all the links

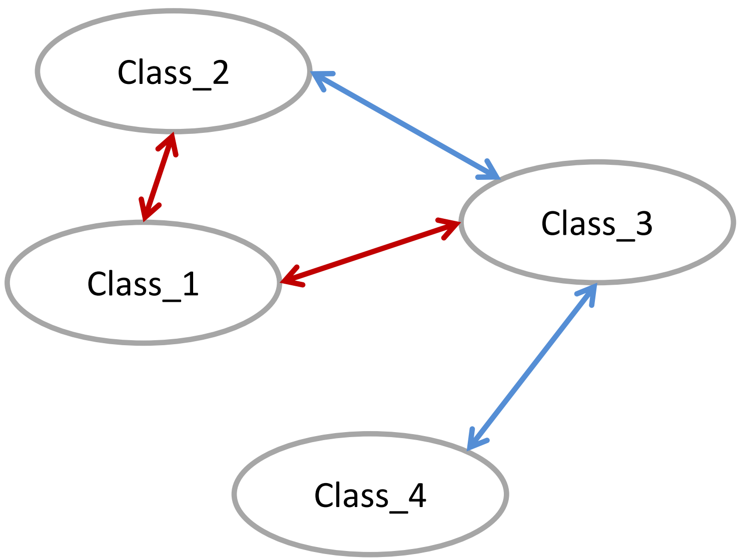


Candidate Links

Source_1	Class_2	95.4%	
Source_3	**Class_4**	**92.1%**	
Source_1	Class_1	85.6%	+ Bonus
Source_2	Class_2	83.2%	
Source_3	Class_3	81.2%	
Source_1	Class_3	79.0%	+ Bonus
Source_3	Class_2	77.5%	
Source_2	Class_4	64.3%	
Source_3	Class_4	53.2%	+ Bonus

Approach 1: Optimistic Combination (O-CSTI)

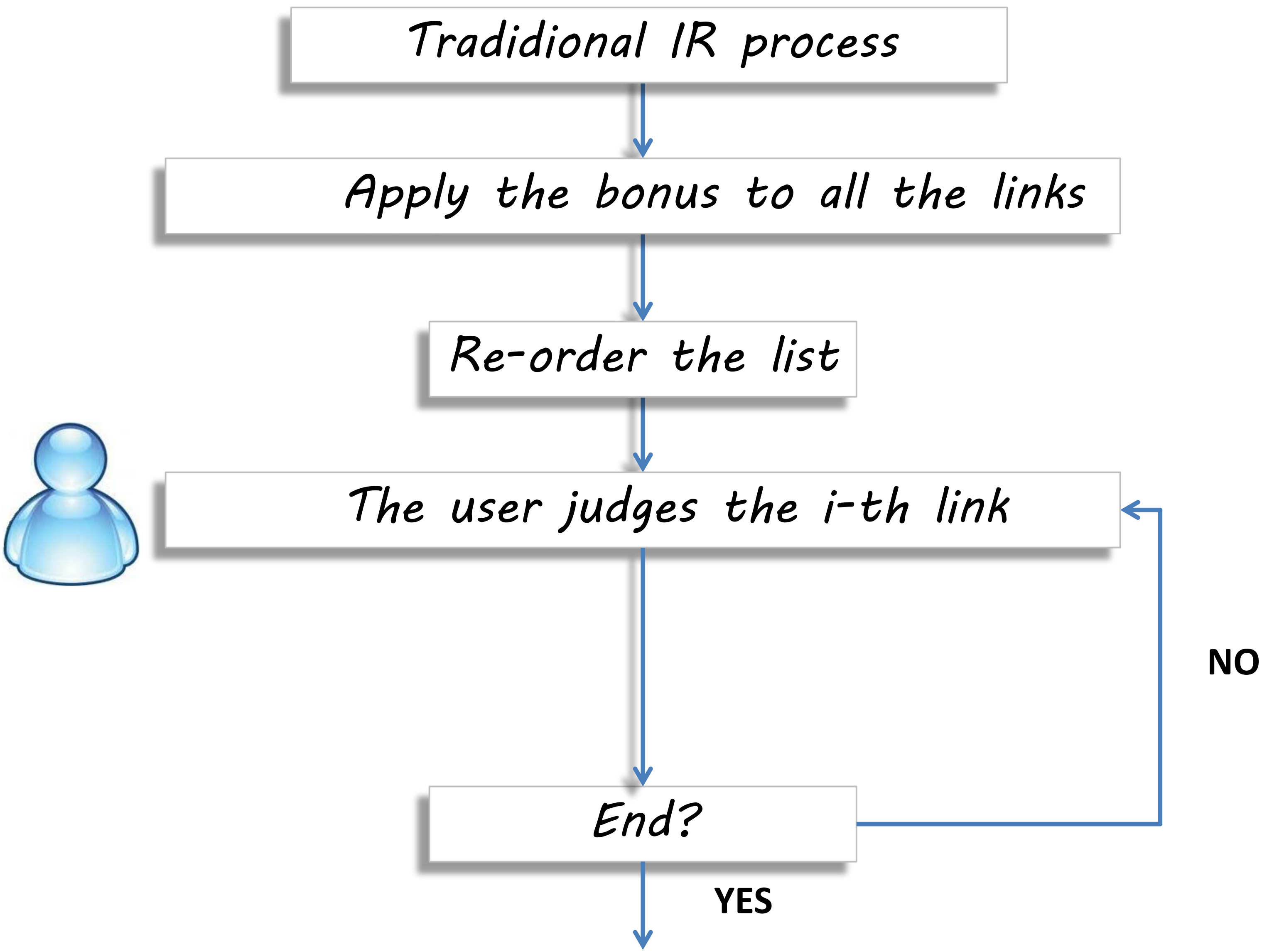
Step 2: applying bonus to all the links



Candidate Links

Source_1	Class_2	95.4%	+ Bonus
Source_3	Class_4	92.1%	
Source_1	**Class_1**	**85.6%**	+ Bonus
Source_2	Class_2	83.2%	
Source_3	Class_3	81.2%	
Source_1	Class_3	79.0%	+ 2Bonus
Source_3	Class_2	77.5%	
Source_2	Class_4	64.3%	
Source_3	Class_4	53.2%	+ Bonus

Approach 1: Optimistic Combination (O-CSTI)



Approach 2: User Driven Combination (U-CSTI)

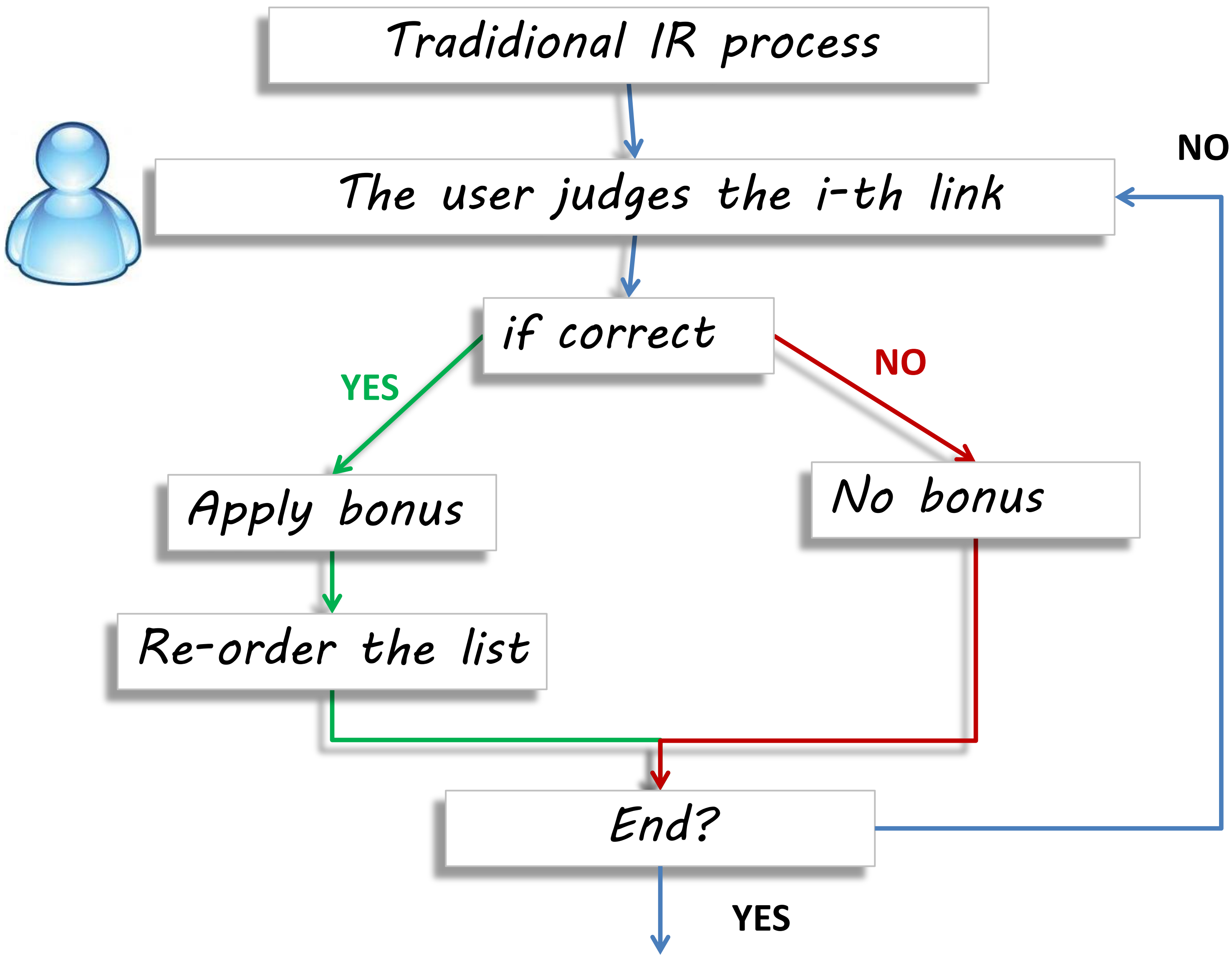
Only the user can say If a link is correct or not

Approach 2: User Driven Combination (U-CSTI)

Only the user can say If a link is correct or not

Only the user can say If applying the combination or not

Approach 2: User Driven Combination (U-CSTI)



Adaptive bonus

Sim(Source1, Class1) = Sim(Source1, Class1) **+ δ * Sim(Source1, Class1)**

Adaptive bonus

Sim(Source1, Class1) = Sim(Source1, Class1) + δ * Sim(Source1, Class1)

$$\delta = \frac{\max(Sim) - \min(Sim)}{2} \approx \text{ranked list variability}$$

Candidate Links

Source_1	Class_2	95.4%
Source_3	Class_4	92.1%
Source_1	Class_1	85.6%
Source_2	Class_2	83.2%
Source_3	Class_3	81.2%
Source_1	Class_3	79.0%
Source_3	Class_2	77.5%
Source_2	Class_4	64.3%
Source_2	Class_3	53.2%

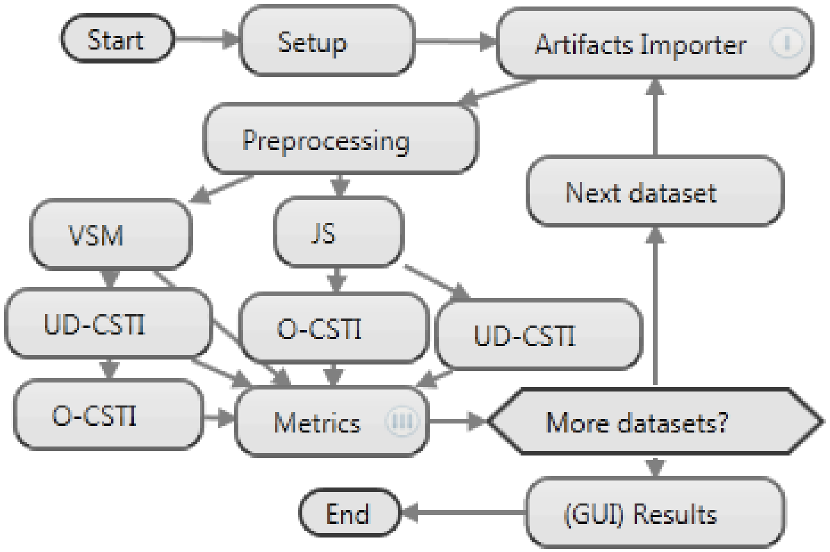
max(Sim)

variability

min(Sim)

Implementation

TraceLab Components

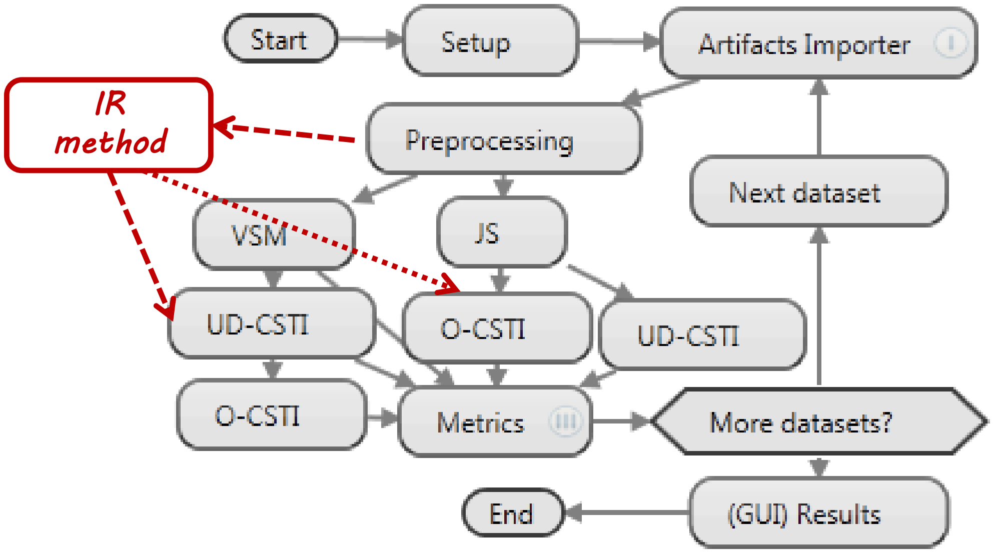


1) Adaptive Bonus
2) Optimistic Comibination (O-CSTI)
3) User Driven Combination (U-CSTI)
4) Different IR Methods:
   - Vector Space Model
   - Jensen-Shannon

We provide the experiments and datasets for download at http://www.cs.wm.edu/semeru/data/csmr13/

Implementation

TraceLab Components



1) Adaptive Bonus
2) Optimistic Comibination (O-CSTI)
3) User Driven Combination (U-CSTI)
4) Different IR Methods:
   - Vector Space Model
   - Jensen-Shannon

We provide the experiments and datasets for download at http://www.cs.wm.edu/semeru/data/csmr13/

Empirical Evaluation

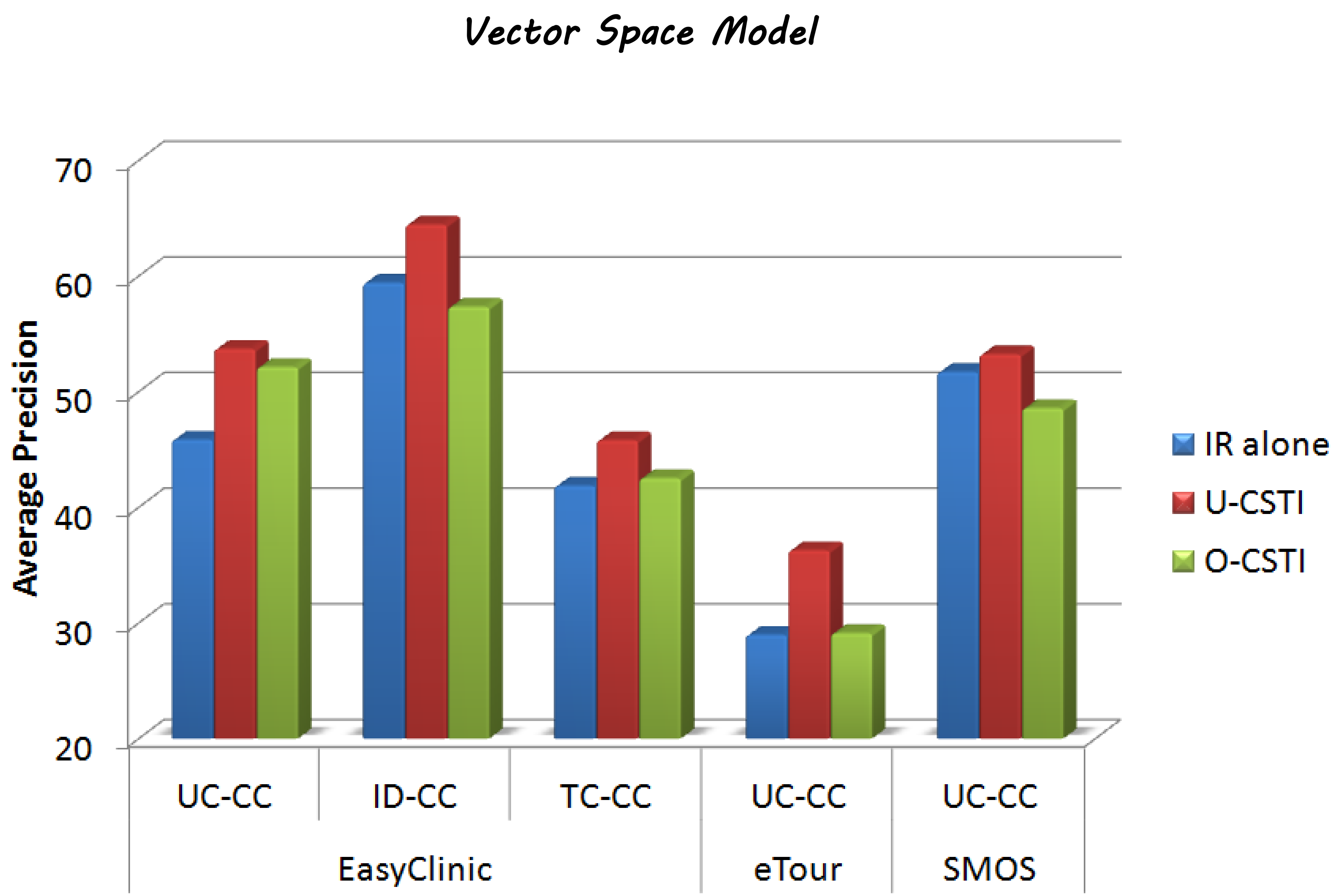
Context

System	Description	KLOC	Source Artifact (#)	Target Artifact (#)	Correct links
EasyClinic	A system used to manage a doctor's office	20	UC (30) UML (20) TC (63)	CC (37) CC (37) CC (37)	93 69 204
eTour	An electronic touristic guide developed by students.	45	UC (58)	CC (174)	366
SMOS	A system used to monitor high school students	23	UC (67)	CC (100)	1,044

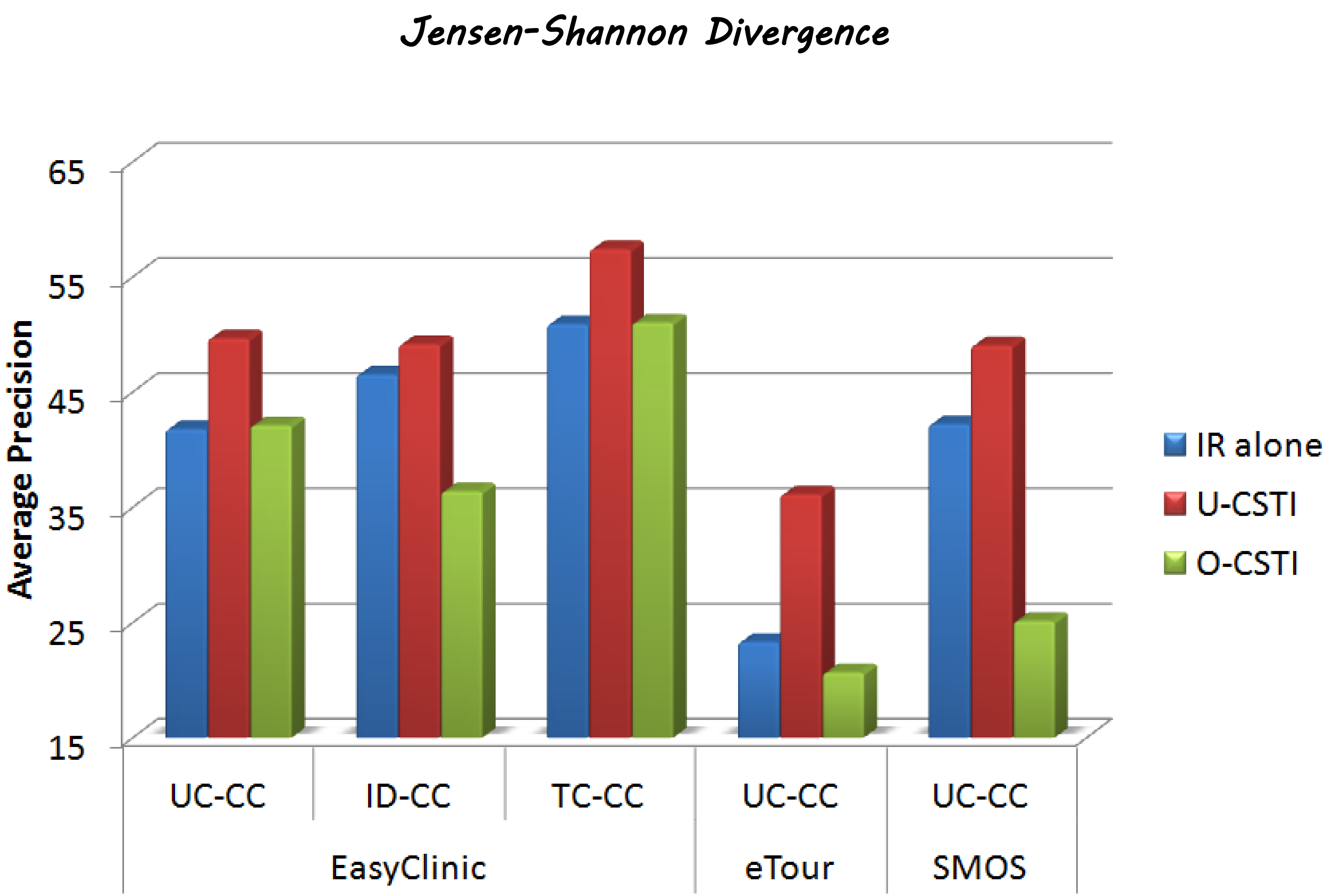
UC: Use case, TC: Test case, CC: Code class

We compared three IR-based processes:
1) IR process alone
2) O-CSTI (optimistic combination)
3) U-CSTI (user driven combination)

Results

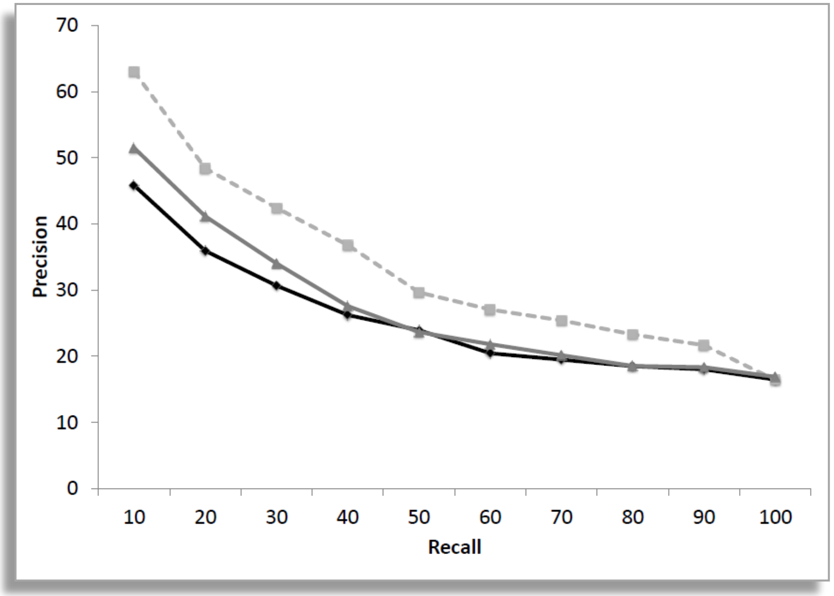

Vector Space Model
Average Precision
70
60
50
40
30
20
UC-CC
ID-CC
TC-CC
UC-CC
UC-CC
EasyClinic
eTour
SMOS
IR alone
U-CSTI
O-CSTI

Results


Jensen-Shannon Divergence
Average Precision
65
55
45
35
25
15
UC-CC
ID-CC
TC-CC
UC-CC
UC-CC
EasyClinic
eTour
SMOS
IR alone
U-CSTI
O-CSTI

Results

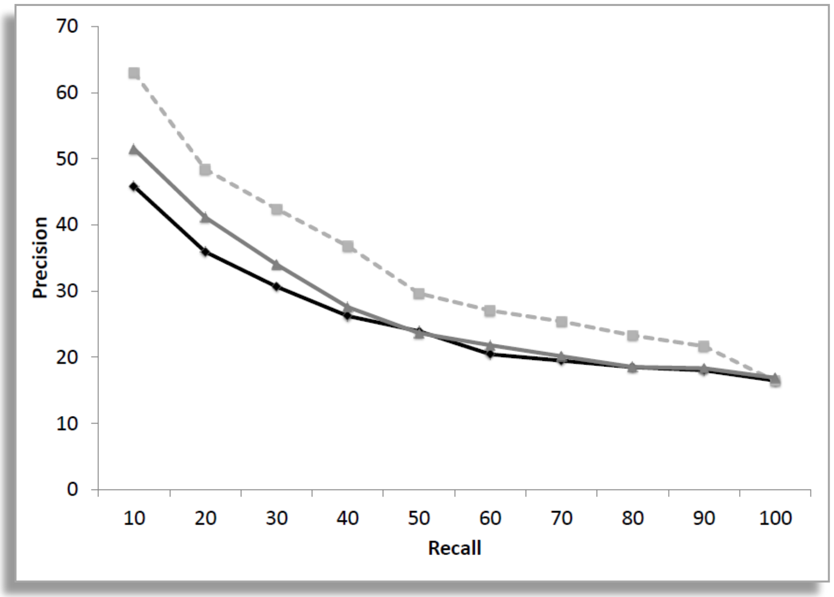
Tracing Use Cases onto Code Classes on SMOS



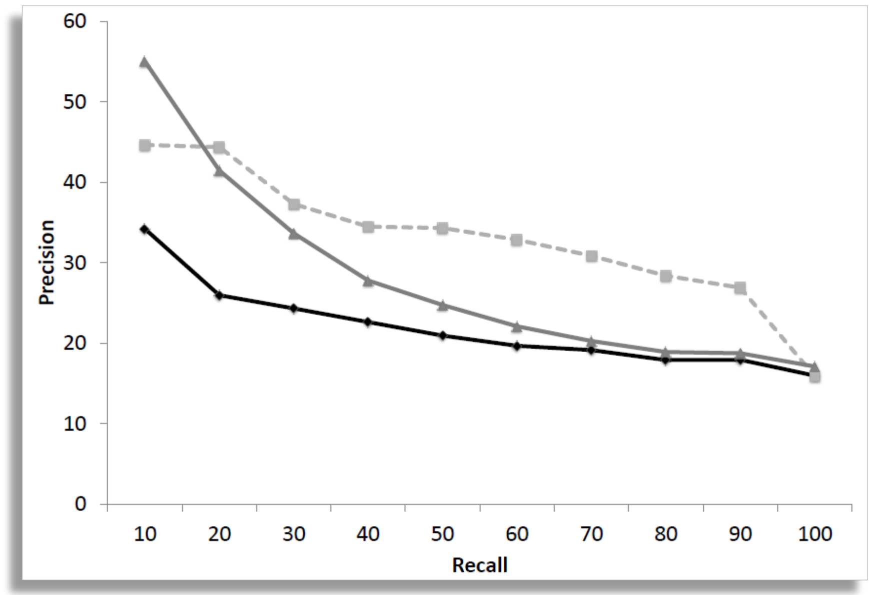
Vector Space Model

Results

Tracing Use Cases onto Code Classes on SMOS



Vector Space Model



Jensen-Shannon Divergence

IR alone UD-CSTI O-CSTI

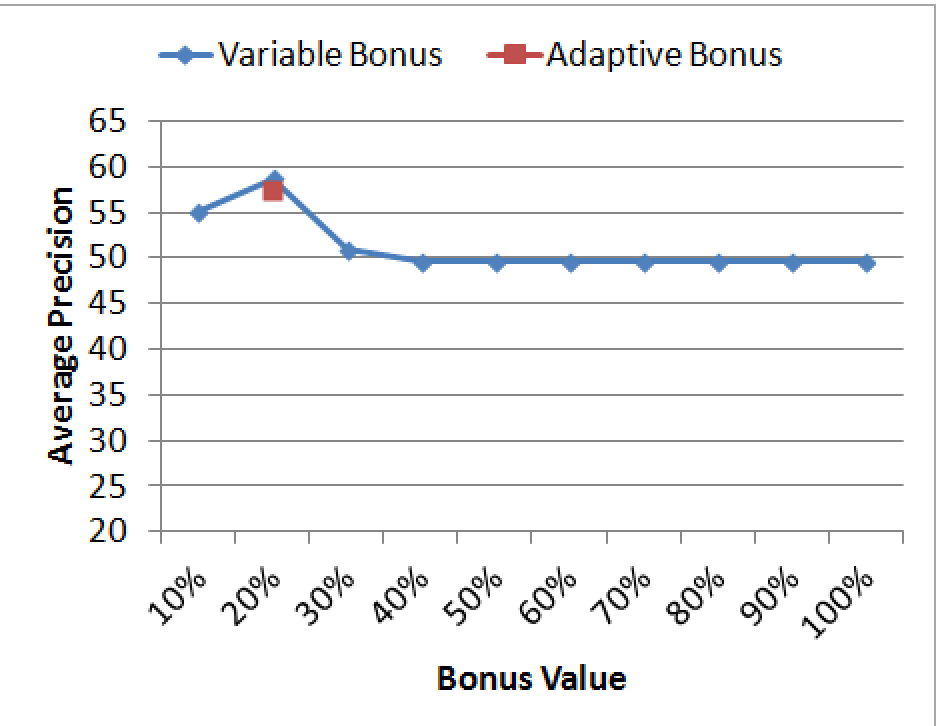
Optimality of the Adaptive Bonus

Adaptive Bonus vs Fixed Bonus
- We used different fixed bonus values

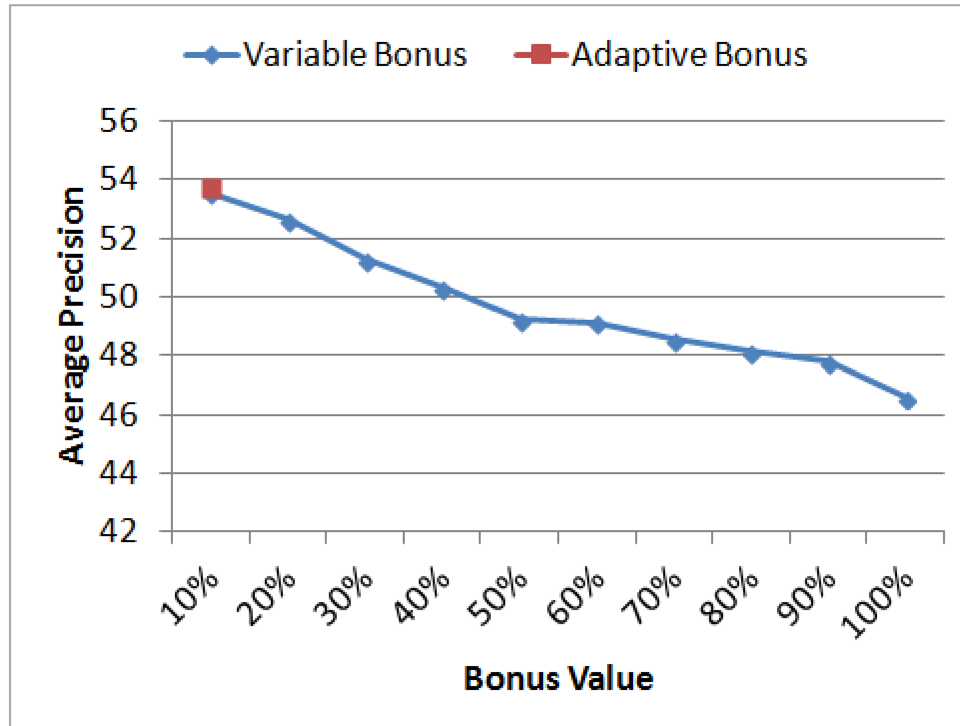
Optimality of the Adaptive Bonus

Adaptive Bonus vs Fixed Bonus
- We used different fixed bonus values

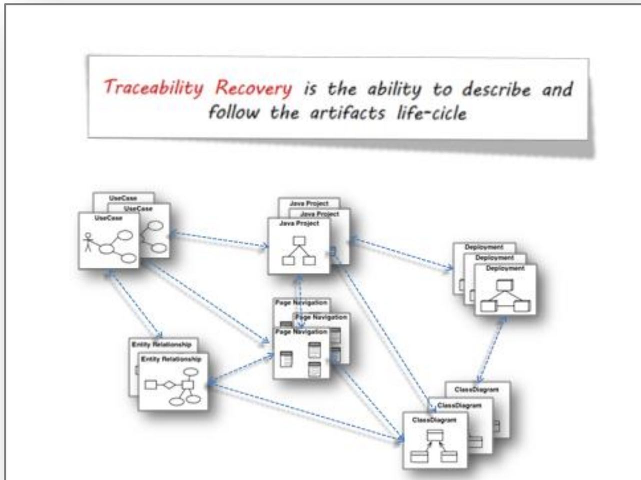
EasyClinic UC-CC with VSM

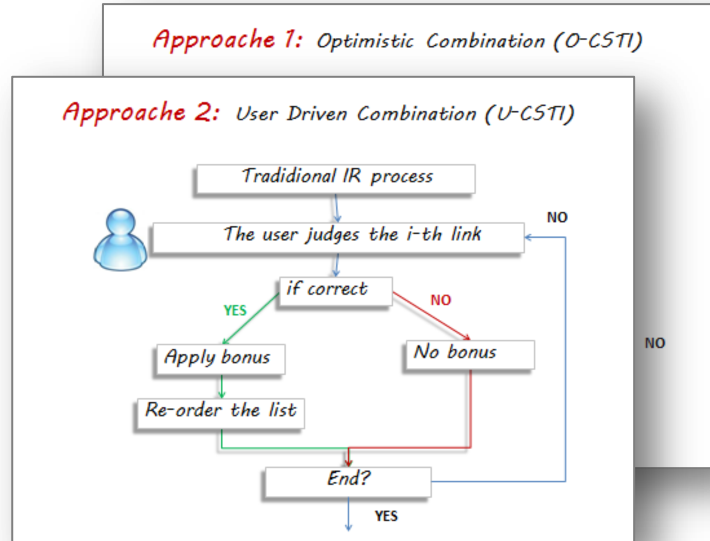


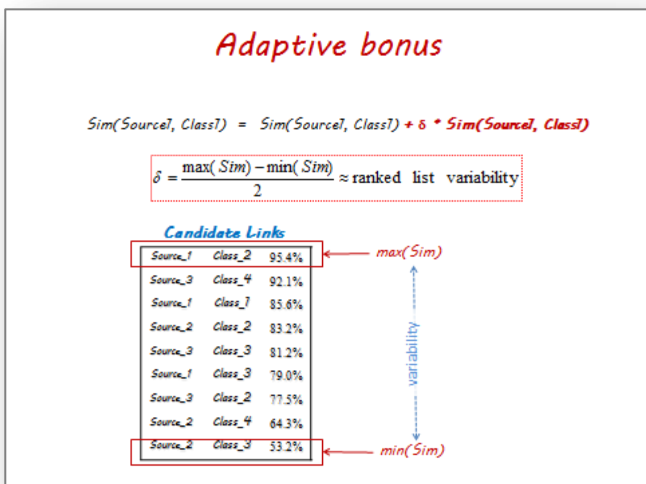
EasyClinic TC-CC with JS

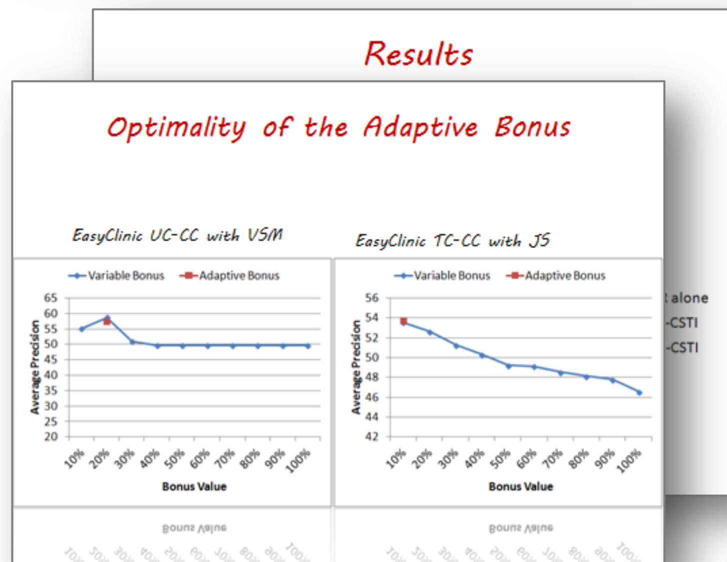


Conclusions









THANKS