ÉCOLE POLYTECHNIQUE MONTRÉAL

Ptidej
Pattern Trace Identification, Detection, and Enhancement in Java

Soccerlab
SOftware Cost-effective Change and Evolution Research Lab

ANALYSING SOURCE CODE STRUCTURE AND MINING SOFTWARE REPOSITORIES TO CREATE REQUIREMENTS TRACEABILITY LINKS

Montréal, 7th December 2012

Ph.D. Defense

Nasir Ali

Outline

- Introduction
- Related Work
- Creation of Experts
- Combining and Usage of Experts' Opinions
- Assigning Weights to Experts
- Empirical Evaluation
- Conclusion and Future Work

Requirements Traceability

Requirements traceability is defined as “the ability to describe and follow the life of a requirement, in both a forwards and backwards direction” [Gotel, 1994]

Requirements Traceability

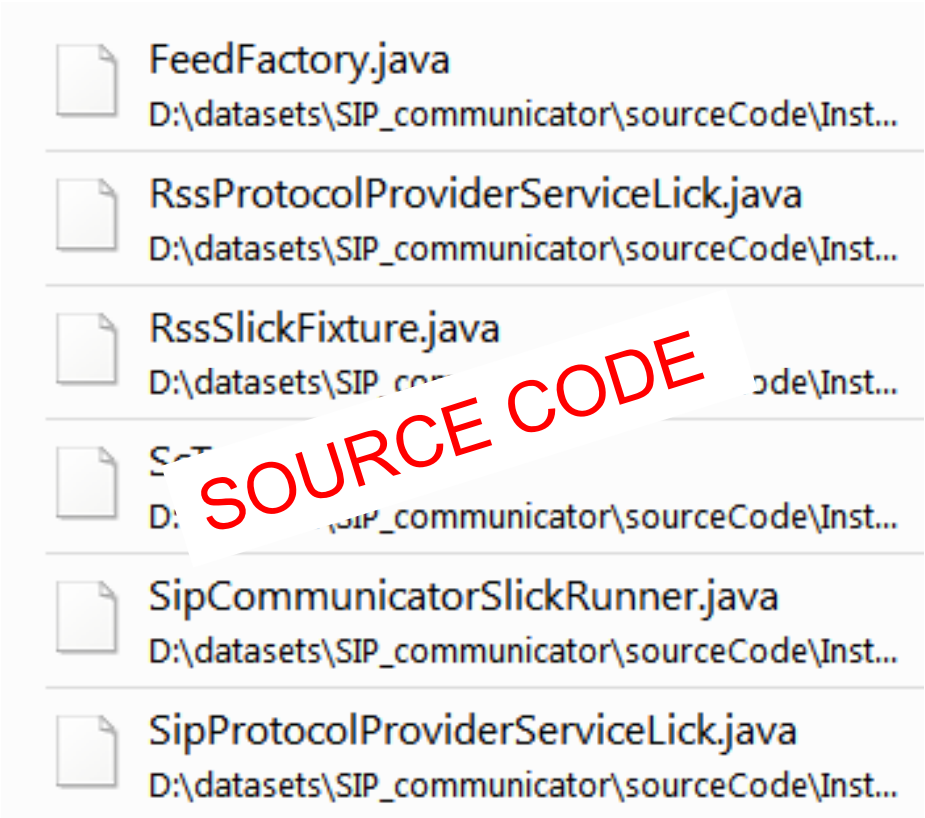
Requirements traceability is defined as “the ability to describe and follow the life of a requirement, in both a forwards and backwards direction” [Gotel, 1994]

Program comprehension

Code location of a requirement

Conformance to specification

Context

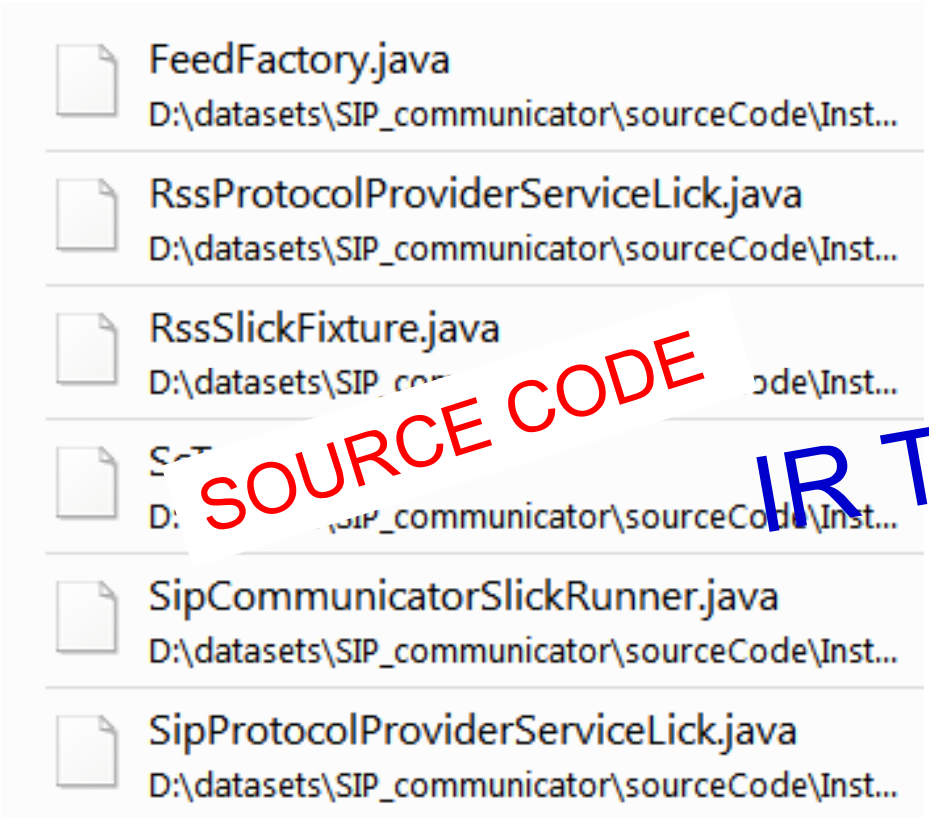


Requirements

RSS option for an instant messenger

.....

Context



SOURCE CODE

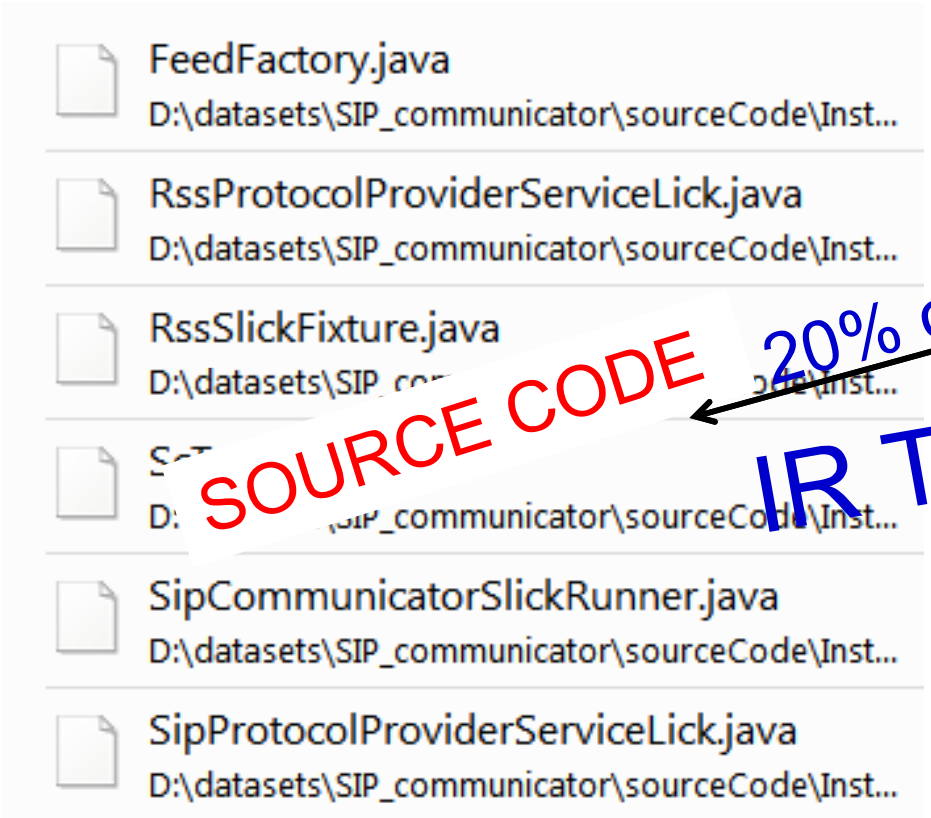
IR Techniques

Requirements

RSS option for an instant messenger

.....

Context



SOURCE CODE

20% of similarity

IR Techniques

Requirements

RSS option for an instant messenger

.....

Outline

- Introduction
- Related Work
- Creation of Experts
- Combining and Usage of Experts' Opinions
- Assigning Weights to Experts
- Empirical Evaluation
- Conclusion and Future Work

Related Work

	Single Expert	Multiple Experts	Combining Experts	Automated Weighting	Feature Location	Req. Traceability
Gethers et al. (2011)		✓	✓	✓	✓	
De Lucia et al. (2011)	✓					✓
Asuncion et al. (2010)	✓					✓
Maletic and Collard (2009)	✓					✓
Abadi et al. (2008)	✓					✓
Mader et al. (2008)	✓					✓
Poshyvanyk et al. (2007)		✓	✓		✓	
Marcus and Maletic (2003)	✓					✓
Antoniol et al. (2002b)	✓					✓

Problem

Requirement

check email address format before storing it in address book

```
public class SendEmail {

    SendEmail (){
        ..................
        ..................
    }

    public sendOutEmail(Sring emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker
        = new EmailAddressFormatChecker();
        int status;

        if(emailAddressFormatChecker.verify(emailId)){
            ..................
            ..................
        }

    }

}
```

Problem

Requirement

check email address format before storing it in address book

```
public class SendEmail {

    SendEmail (){
        ..................
        ..................
    }

    public sendOutEmail(Sring emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker
        = new EmailAddressFormatChecker();
        int status;

        if(emailAddressFormatChecker.verify(emailId)){
            ..................
            ..................
        }

    }

}
```

Problem

Poshyvanyk et al. (2007)

Probabilistic Ranking + Execution Traces

Requirement

check email address format before storing it in address book

```
public class SendEmail

    SendEmail (){
        ................
        ................
    }

    public sendOutEmail(Sring emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker
        = new EmailAddressFormatChecker();
        int status;

        if(emailAddressFormatChecker.verify(emailId)){
            ...................
            ...................
        }

    }

}
```

Problem

Gethers et al. (2011)

VSM + JSM + RTM

Requirement

check email address format before storing it in address book

```
public class SendEmail

    SendEmail (){
        ...............
        ................
    }

    public sendOutEmail(Sring emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker
        = new EmailAddressFormatChecker();
        int status;

        if(emailAddressFormatChecker.verify(emailId)){
            ..................
            ..................
        }

    }

}
```

Thesis

Adding more sources of information and combining them with IR techniques could improve the accuracy of IR techniques for requirements traceability

Sources of Information

- Software Repositories
- Static Class Relationship
- Source Code Entities

We use each source of information to create experts that verify a link created by an IR technique.

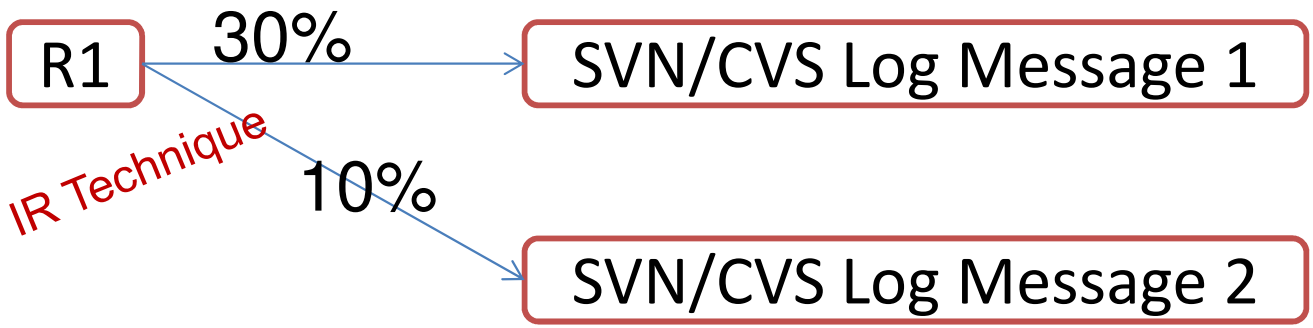
Outline

- Introduction
- Related Work
- Creation of Experts
- Combining and Usage of Experts' Opinions
- Assigning Weights to Experts
- Empirical Evaluation
- Conclusion and Future Work

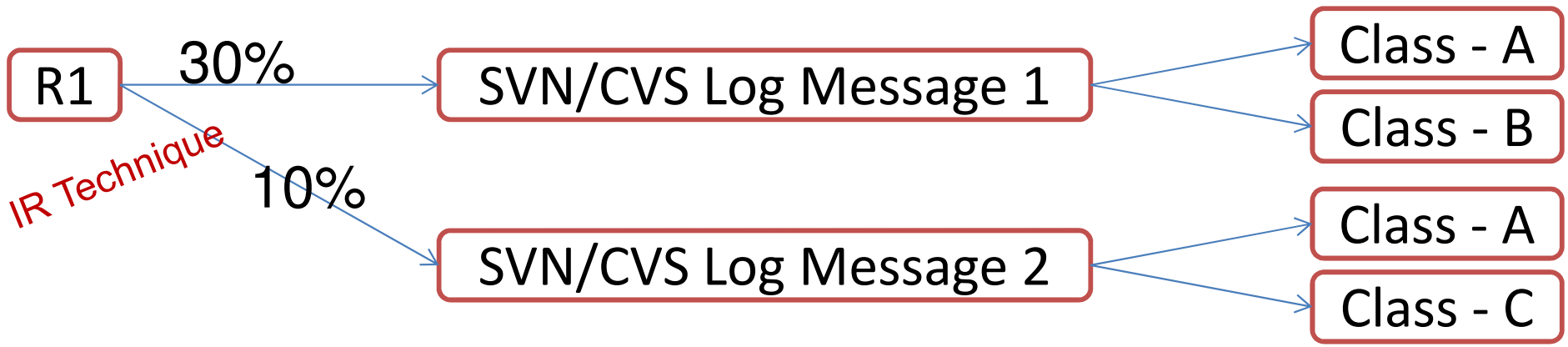
Creation of Experts

- **Histrace:** It mines software repositories to build experts
- **BCRTrace:** It uses static relationships among classes to build experts
- **Partrace:** It partitions source code to use them as experts

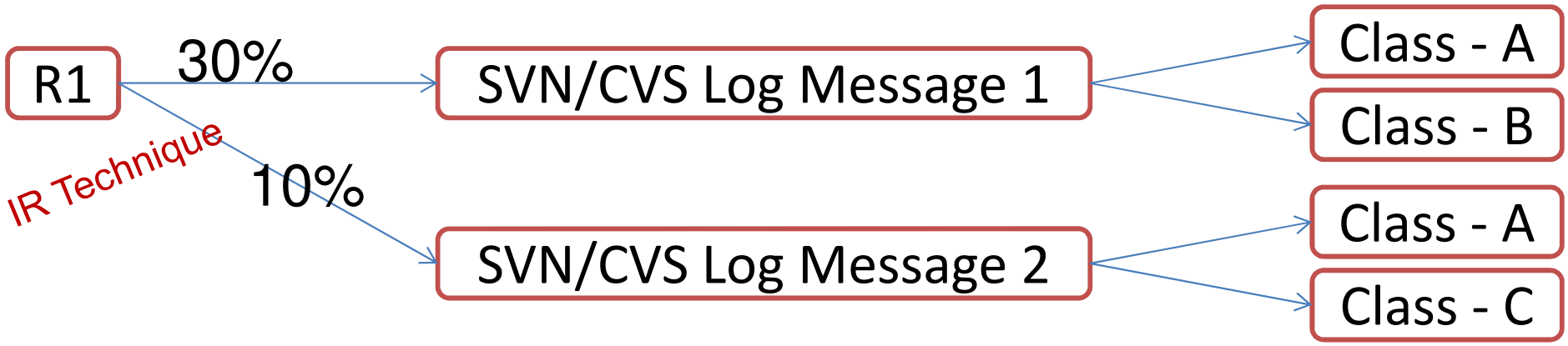
Creation of Histrace Expert

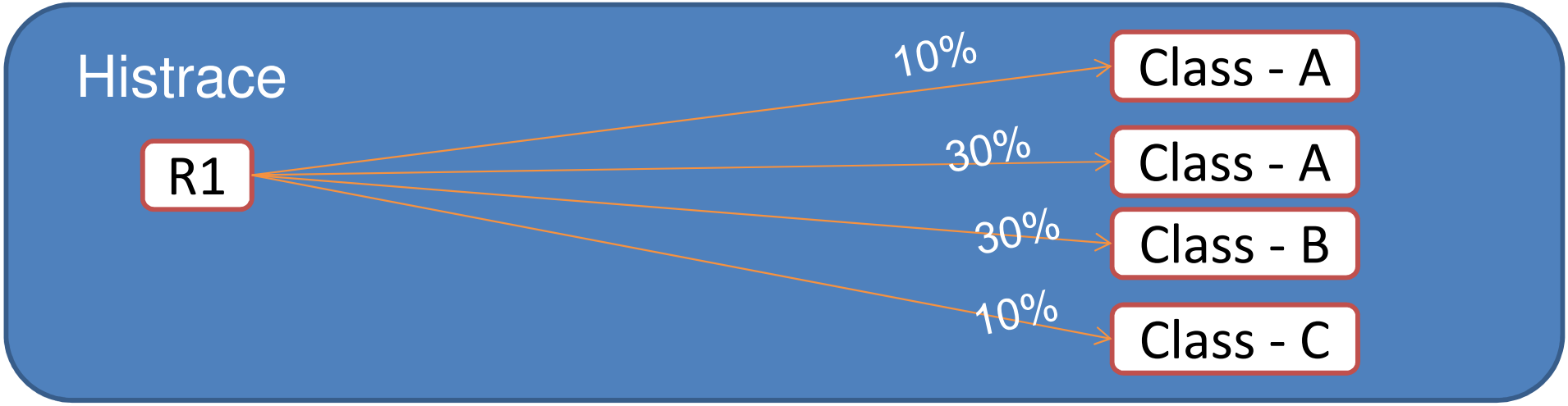


Creation of Histrace Expert



Creation of Histrace Expert





Creation of BCRTrace Expert

0.70	FeedFactory.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.65	RssProtocolProviderServiceLick.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.59	RssSlickFixture.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.56	ScTestRunner.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.35	SipCommunicatorSlickRunner.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.34	SipProtocolProviderServiceLick.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.27	SipSlickFixture.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.19	SlicklessTests.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.12	TestAccountInstallation.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.09	TestAccountInstallation.java	D:\datasets\SIP_communicator\sourceCode\Inst...
0.06	TestAccountInstallation.java	D:\datasets\SIP_communicator\sourceCode\Inst...

IR

R1 – Instant Messenger should support RSS protocol

Creation of BCRTrace Expert

BCR

0.70	FeedFactory.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.65	RssProtocolProviderServiceLick.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.59	RssSlickFixture.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.56	ScTestRunner.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.35	SipCommunicatorSlickRunner.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.34	SipProtocolProviderServiceLick.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.27	SipSlickFixture.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.19	SlicklessTests.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.12	TestAccountInstallation.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.09	TestAccountInstallation.java D:\datasets\SIP_communicator\sourceCode\Inst...
0.06	TestAccountInstallation.java D:\datasets\SIP_communicator\sourceCode\Inst...

IR

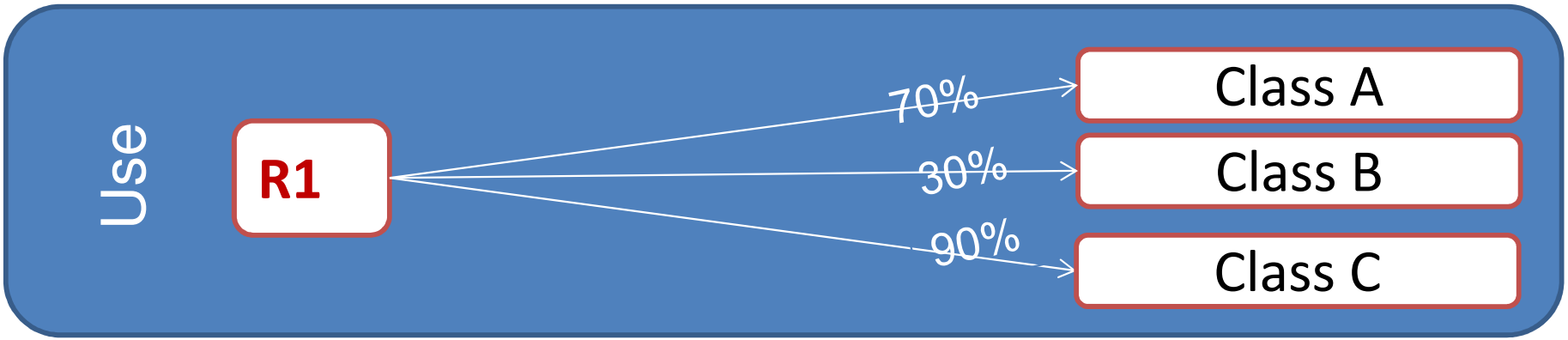
R1 – Instant Messenger should support RSS protocol

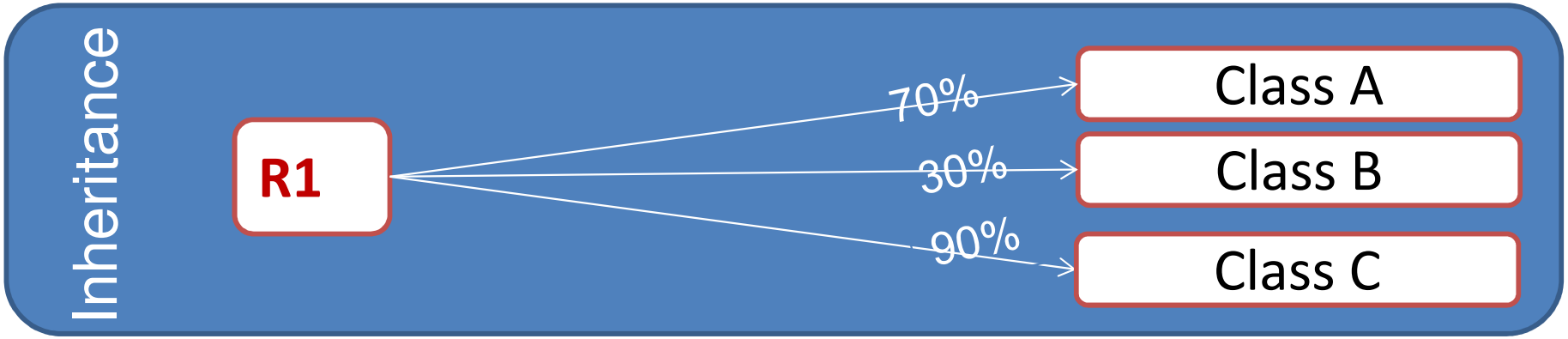
Creation of BCRTrace Expert

IR

R1 – Instant Messenger should support RSS protocol

Creation of BCRTrace Expert



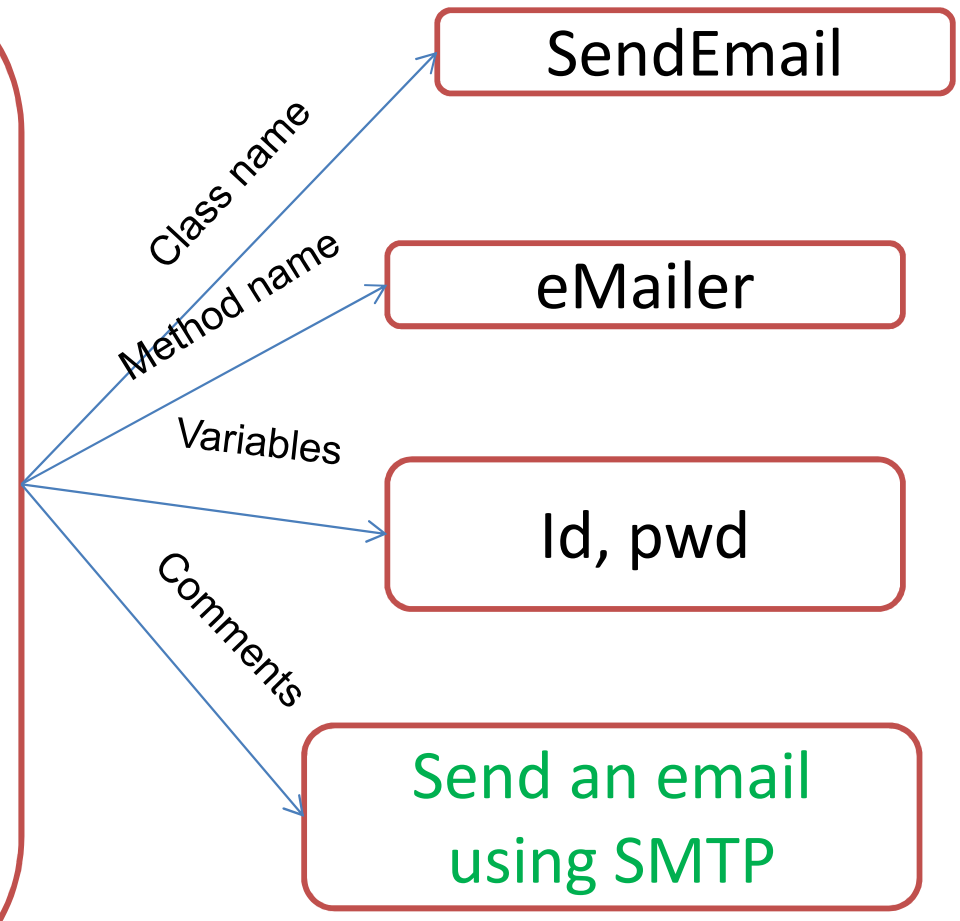


Creation of Partrace Expert

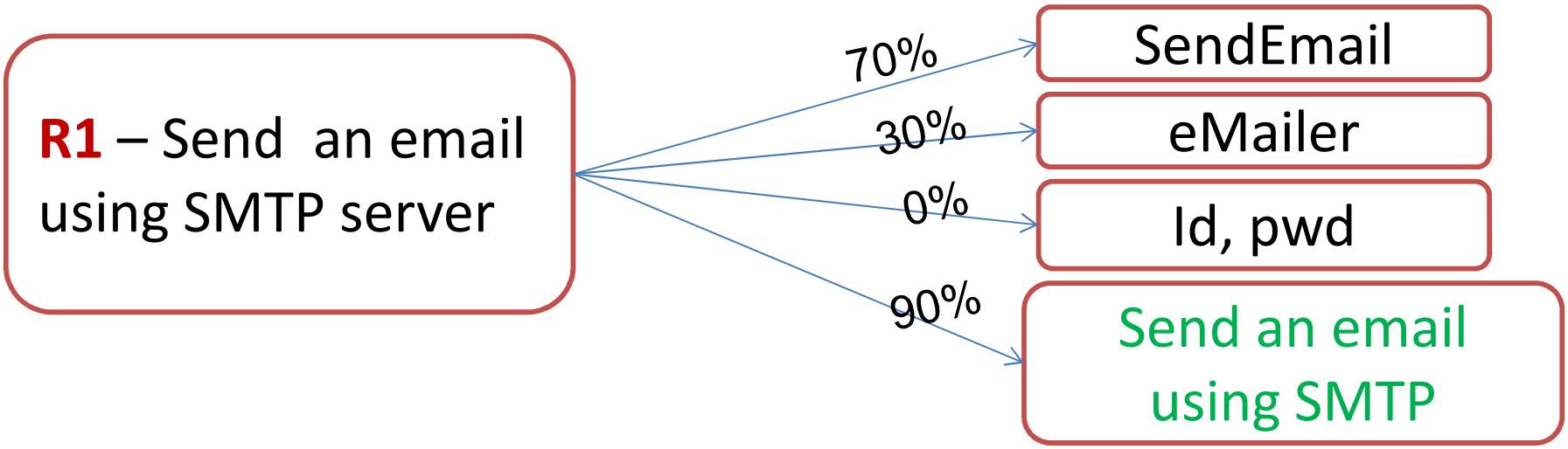
```
//Send an email using SMTP

class SendEmail {

    private void eMailer(){

        String id;
        String pwd;

        ................

    }

}
```

Creation of Partrace Expert

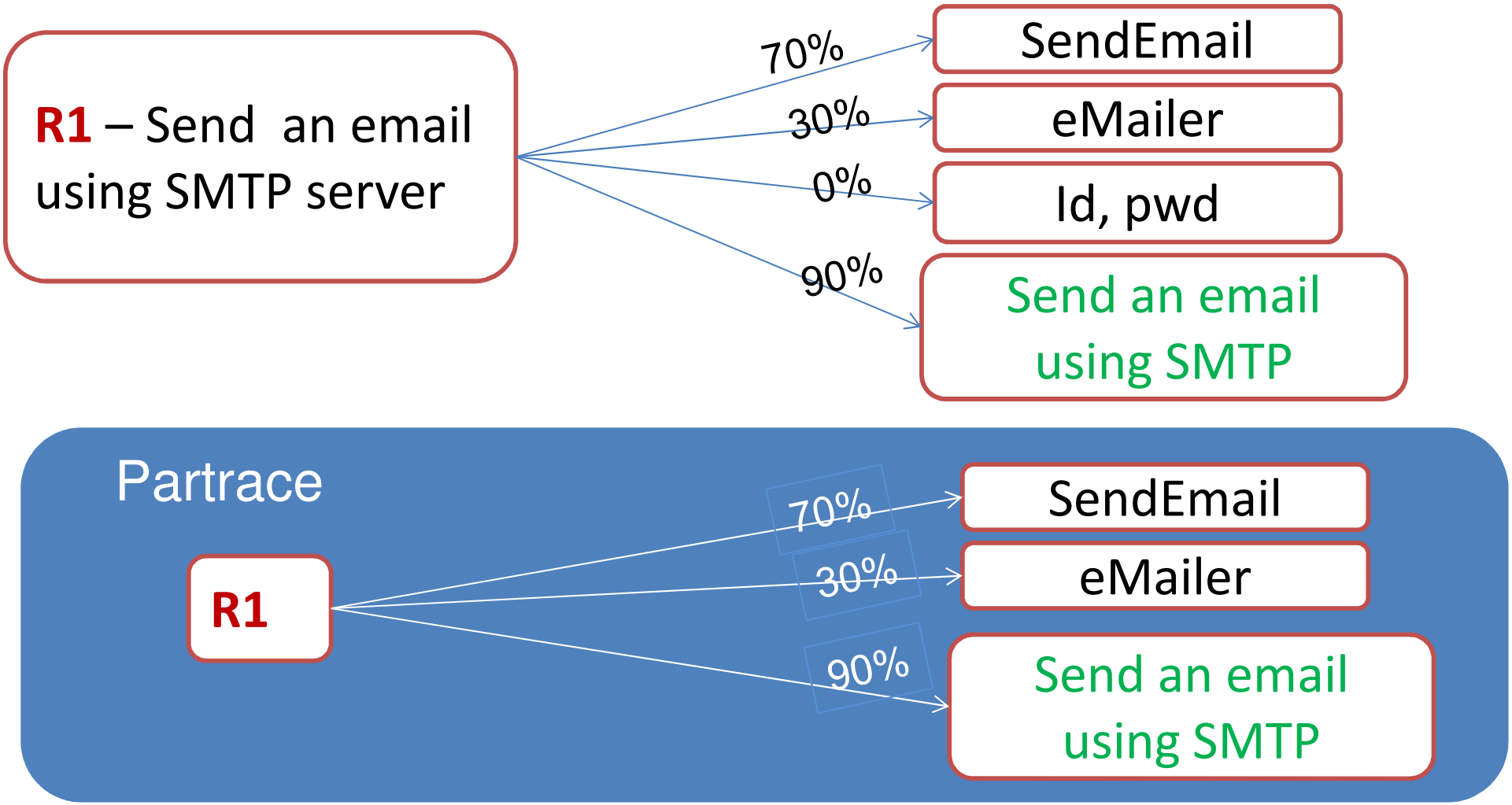
```
//Send an email using SMTP
class SendEmail {
    private void eMailer(){
        String id;
        String pwd;
        ................
    }
}
```



Creation of Partrace Expert



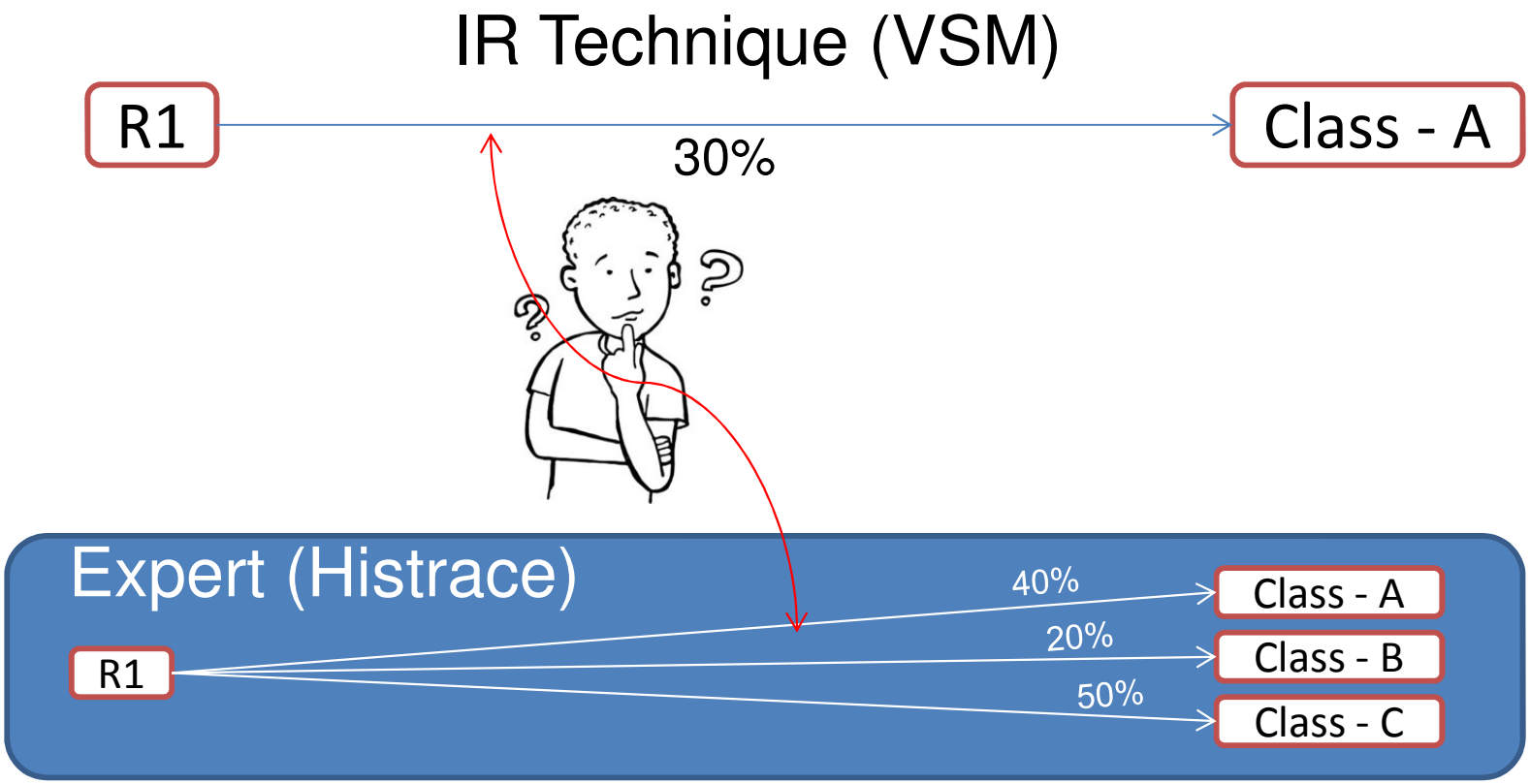
Creation of Partrace Expert



Outline

- Introduction
- Related Work
- Creation of Experts
- Combining and Usage of Experts' Opinions
- Assigning Weights to Experts
- Empirical Evaluation
- Conclusion and Future Work

Combining Experts' Opinions



Combining Experts' Opinions

TRUMO – Trust Model

Combining Experts' Opinions

TRUMO – Trust Model

- It uses IR created links as baseline links (initial trust)

Combining Experts' Opinions

TRUMO – Trust Model

- It uses IR created links as baseline links (initial trust)
- It asks experts, e.g., Histrace, for the evidence of baseline links (reputation trust)

Combining Experts' Opinions

TRUMO – Trust Model

- It uses IR created links as baseline links (initial trust)
- It asks experts, e.g., Histrace, for the evidence of baseline links (reputation trust)
- Only keep a link if experts provide any evidence and discard remaining (constraint)

Combining Experts' Opinions

TRUMO – Trust Model

Combining Experts' Opinions

TRUMO – Trust Model

- It counts how many times an experts provides evidence for a link

Combining Experts' Opinions

TRUMO – Trust Model

- It counts how many times an experts provides evidence for a link
- It keeps the similarity values returned from the expert for a link and baseline links similarity values

Combining Experts' Opinions

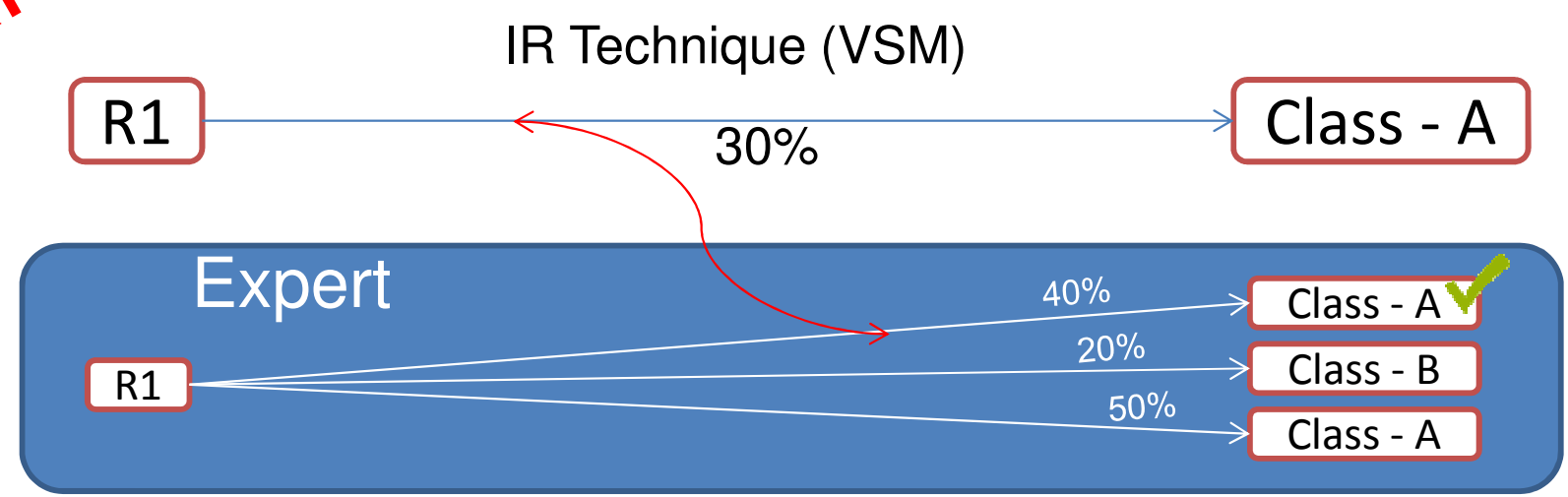
TRUMO – Trust Model

- It counts how many times an experts provides evidence for a link
- It keeps the similarity values returned from the expert for a link and baseline links similarity values
- It assigns weights to: (i) similarity values (ii) number of times a link referred by an expert, to compute a new similarity for a link

Combining Experts' Opinions

TRUMO – Trust Model

Example

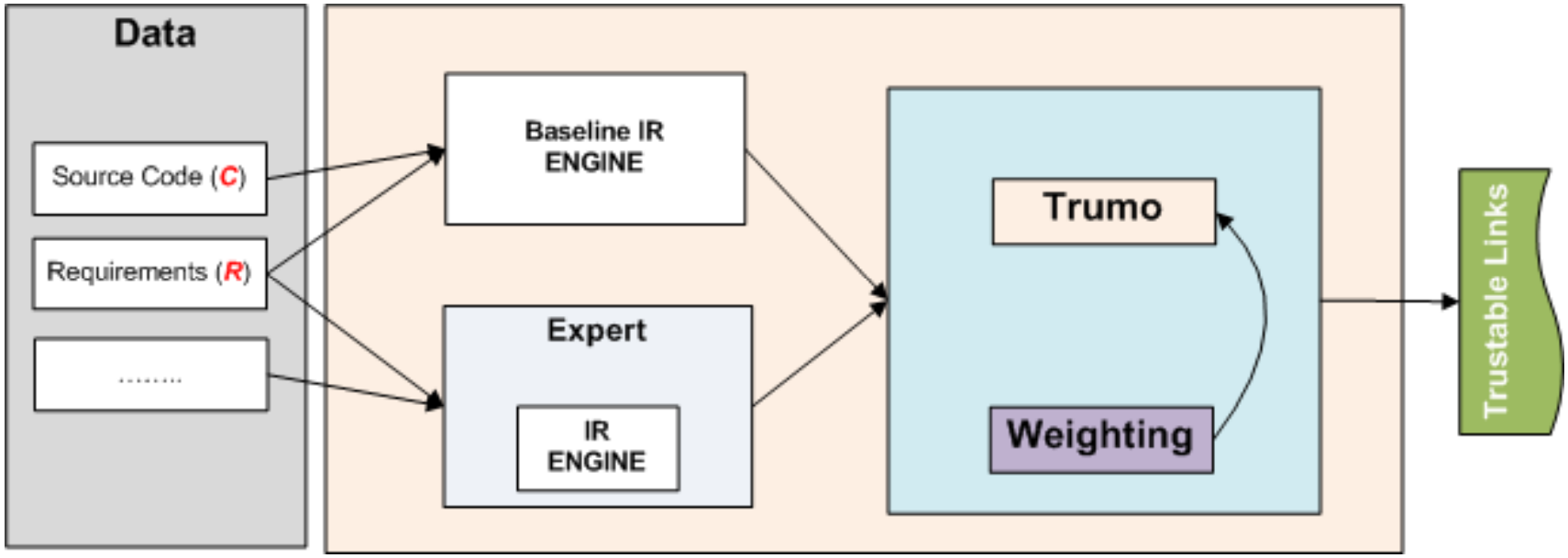


λ_1 0.4% + λ_2 0.5 + λ_3 0.2

R1 → Class - A

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$

Usage of Experts



Outline

- Introduction
- Related Work
- Creation of Experts
- Combining and Usage of Experts' Opinions
- Assigning Weights to Experts
- Empirical Evaluation
- Conclusion and Future Work

DynWing Weighting

Addressing assigning weights to different experts problem as maximization Problem

Example

Link ID	Expert 1	Expert 2	% of time	Final Similarity
1	0.21	0.35	14/75 = 0.19	?
	λ_1	λ_2	λ_3	

$$\lambda_1 = 0.1 \ , \ \lambda_2 = 0.1 \ , \ \lambda_3 = 0.8$$

$$\lambda_1 = 0.3 \ , \ \lambda_2 = 0.2 \ , \ \lambda_3 = 0.5$$

$$\lambda_1 = 0.2 \ , \ \lambda_2 = 0.5 \ , \ \lambda_3 = 0.3$$

Static Weighting

Manually assign weight to each expert [Poshyvanyk et al. (2007)]

Example

Link ID	Expert 1	Expert 2	% of time	Final Similarity
1	0.21	0.35	14/75 = 0.19	?
	λ_1	λ_2	λ_3	

$\lambda_1 = 0.33$, $\lambda_2 = 0.33$, $\lambda_3 = 0.33$

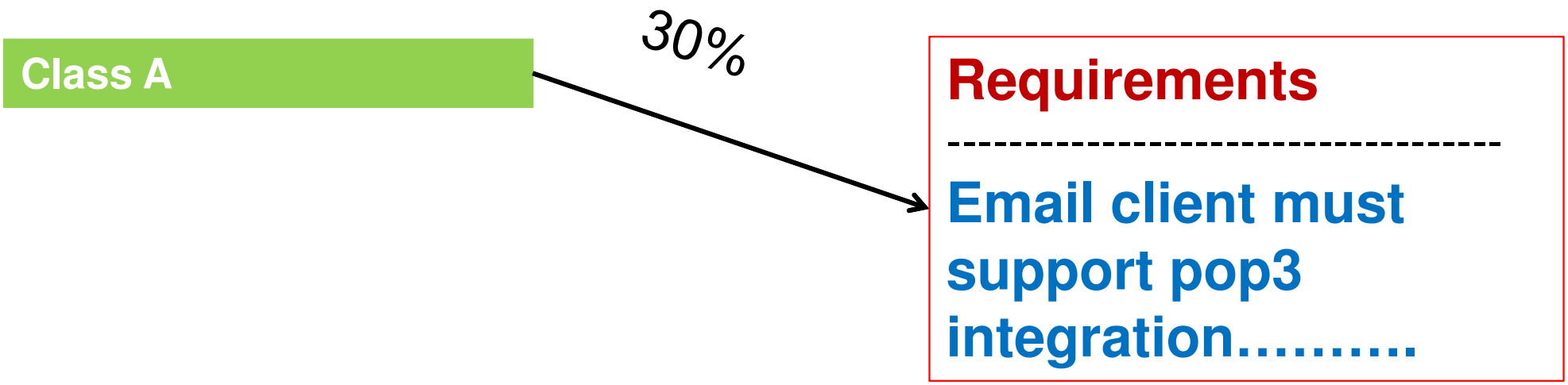
Voting

Class A

Requirements

Email client must support pop3 integration…….

Voting


Class A
30%
Requirements
Email client must support pop3 integration…….

Voting

Class A

30%

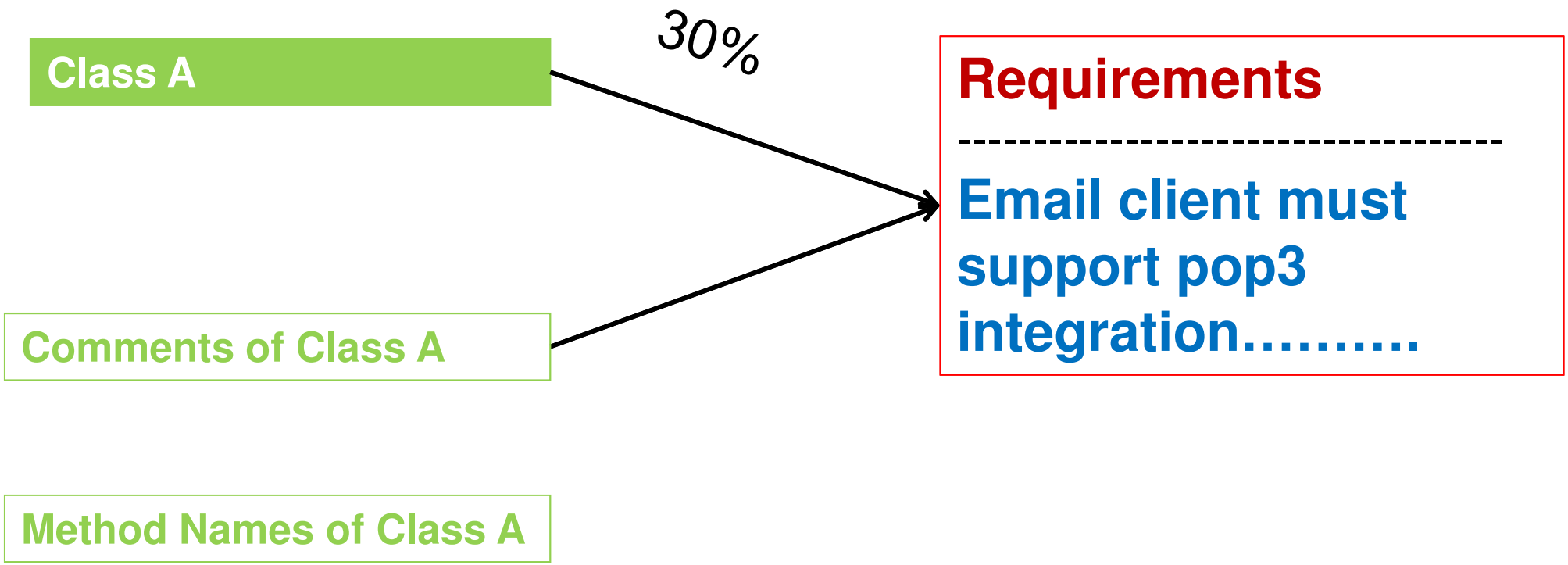
Requirements

Email client must support pop3 integration…….

Comments of Class A

Method Names of Class A

Voting

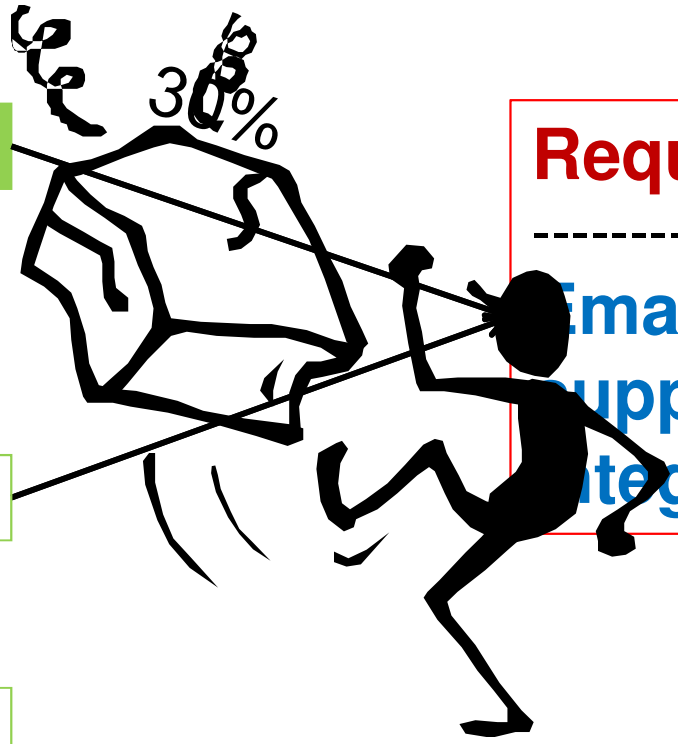

Class A
30%
Requirements
Email client must support pop3 integration…….
Comments of Class A
Method Names of Class A

Voting

Class A

Comments of Class A

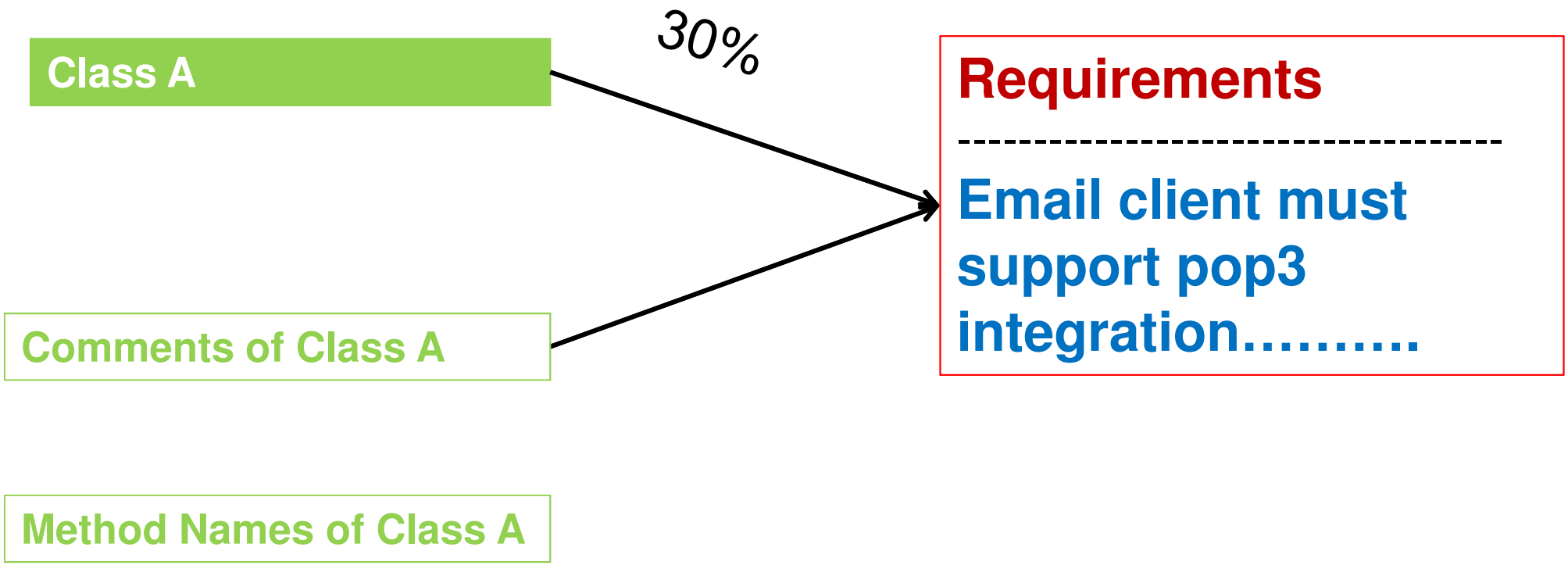
Method Names of Class A



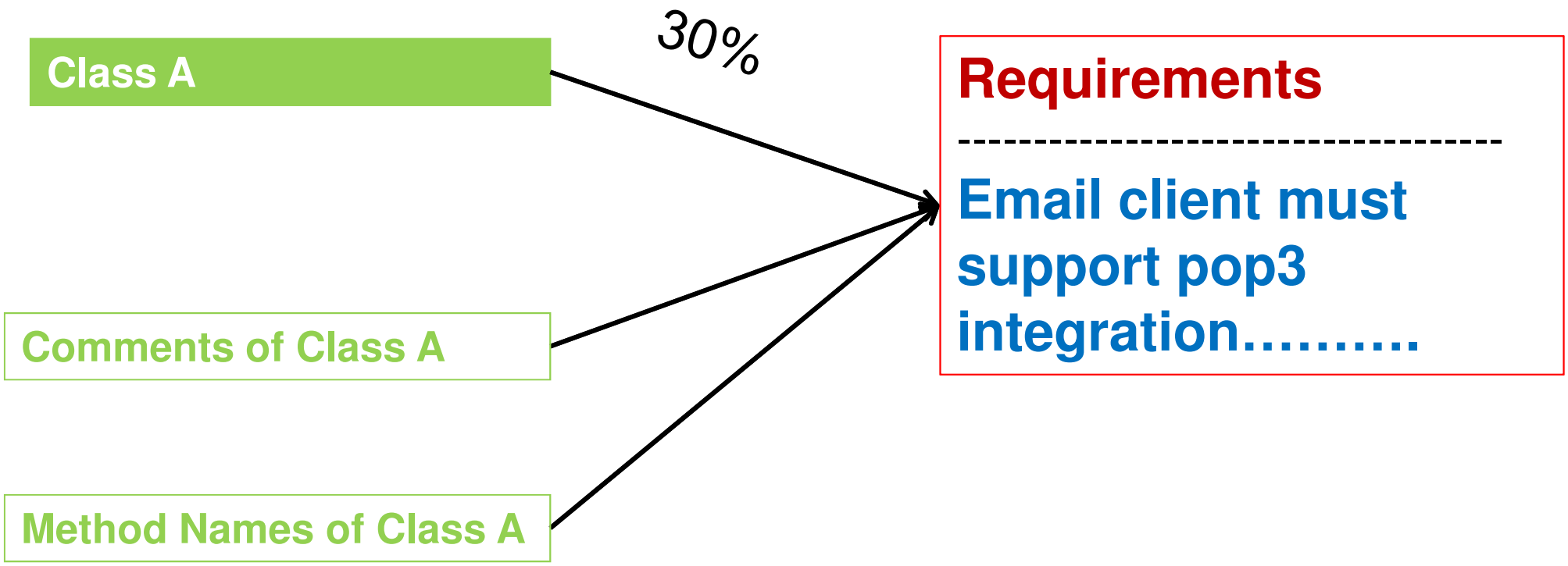
Requirements

Email client must support pop3 integration.....

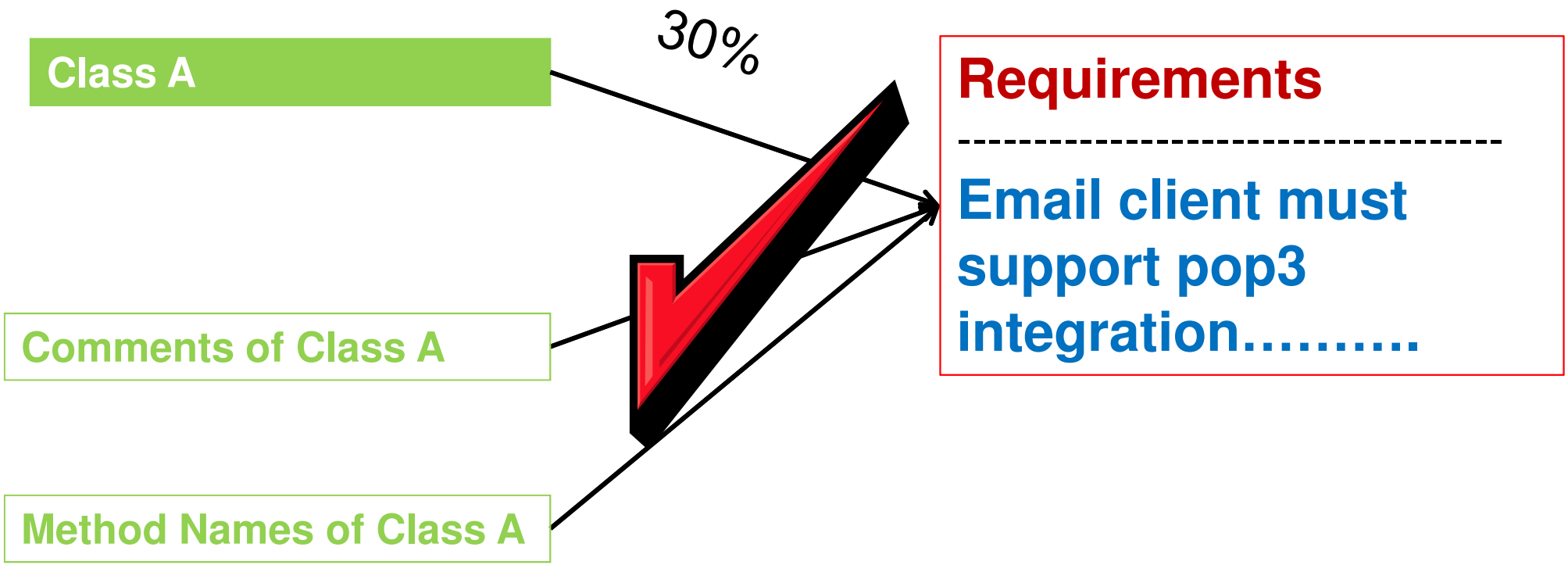
Voting


Class A
30%
Requirements
Email client must support pop3 integration.....
Comments of Class A
Method Names of Class A

Voting


Class A
30%
Comments of Class A
Method Names of Class A
Requirements
Email client must support pop3 integration…….

Voting


Class A
30%
Requirements
Email client must support pop3 integration.....
Comments of Class A
Method Names of Class A

Outline

- Introduction
- Related Work
- Creation of Experts
- Combining and Usage of Experts' Opinions
- Assigning Weights to Experts
- Empirical Evaluation
- Conclusion and Future Work

Empirical Evaluation

	Trumo	DynWing	Trumo (Ranker)	Static Weight	PCA-based Weights	Voting	JSM	LSI	VSM
Histrace	✓	✓		✓	✓		✓		✓
Partrace						✓			✓
BCRTrace			✓	✓				✓	✓

Empirical Evaluation

- RQ1 - Does using an expert provide better accuracy than IR technique?
- RQ2 - Can Trumo be used for other software maintenance task, i.e., bug location?
- RQ 3 - How does the accuracy of the traceability links recovered using DynWing compare to that using static weight and PCA?

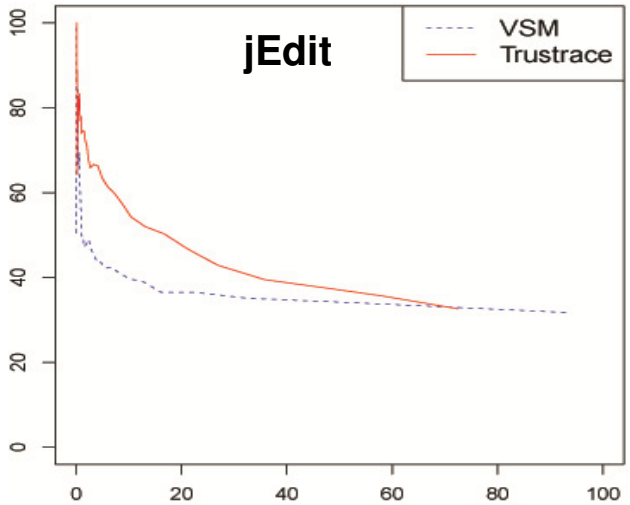
Empirical Evaluation

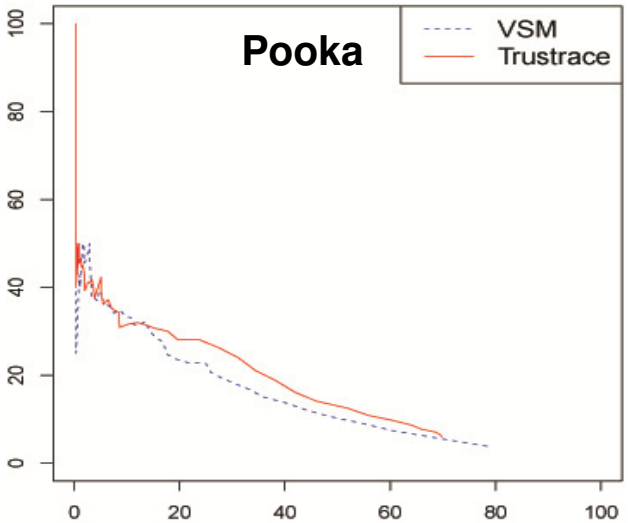
	Trumo	DynWing	Trumo (Ranker)	Static Weight	PCA-based Weights	Voting	JSM	LSI	VSM
Histrace	✓	✓		✓	✓		✓		✓
Partrace						✓			✓
BCRTrace			✓	✓				✓	✓

Trustrace

RQ1 – **Histrace** Provides Better **Accuracy** Than VSM and JSM

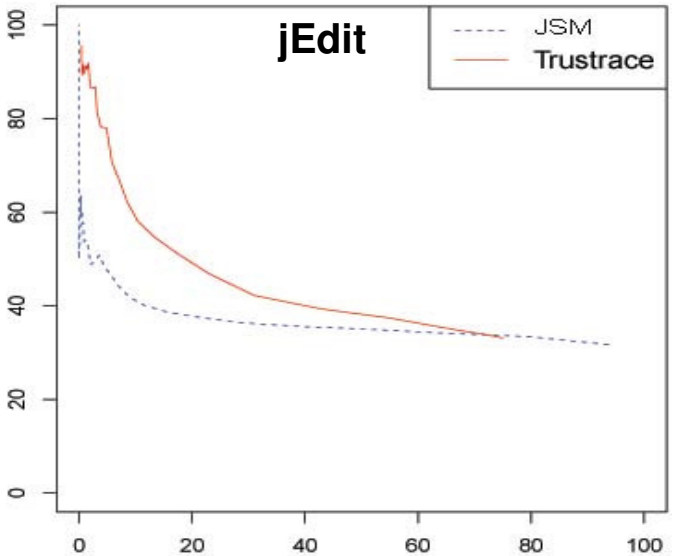
VSM

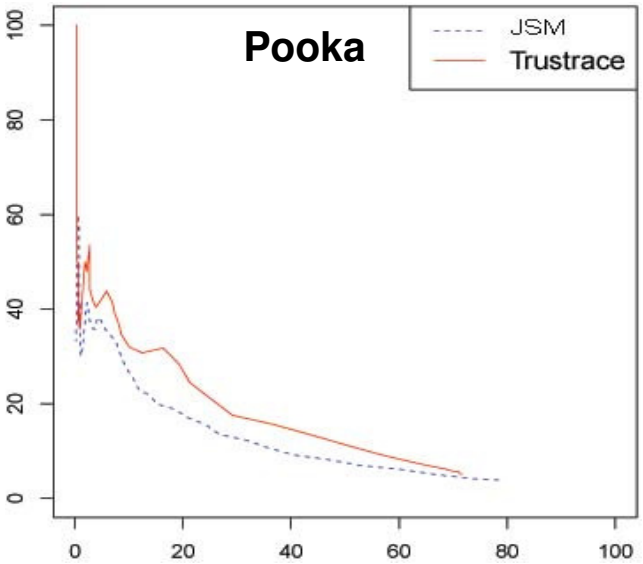




X Axis Shows Recall and Y Axis Shows Precision Values

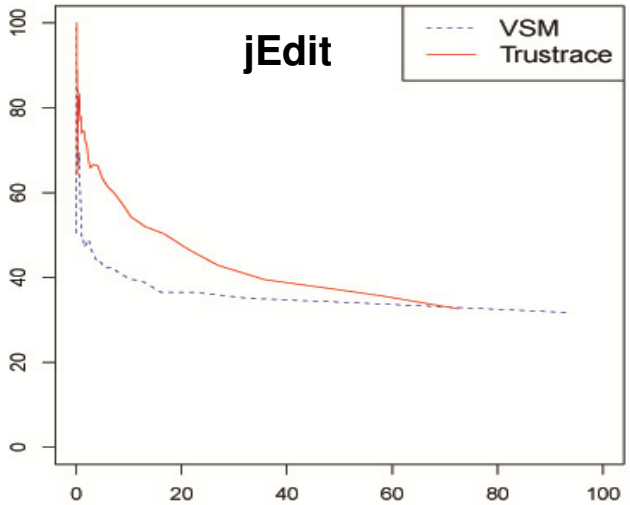
JSM



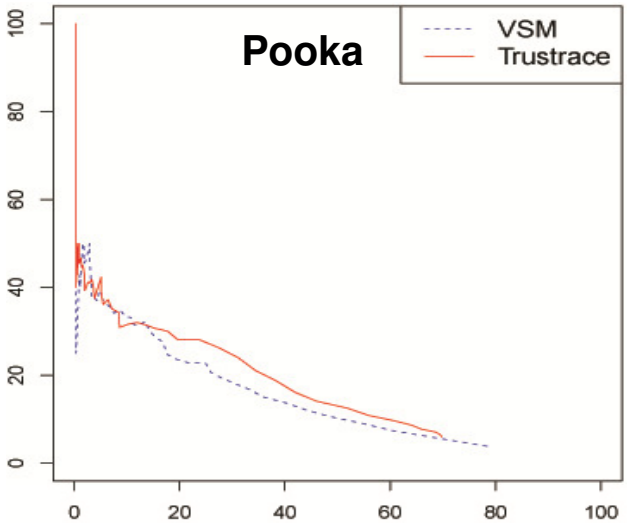


RQ1 – **Histrace** Provides Better **Accuracy** Than VSM and JSM

VSM

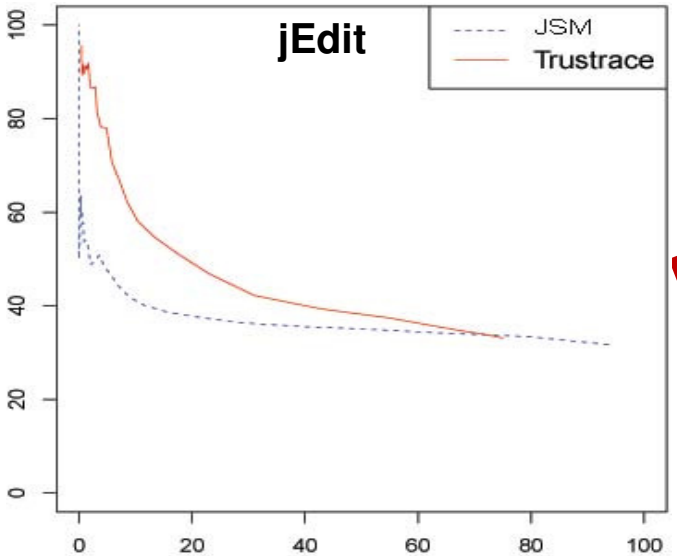


P<0.05

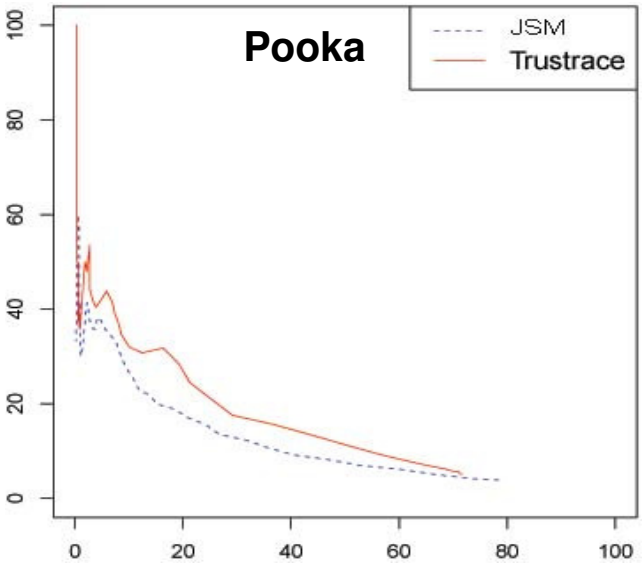


X Axis Shows Recall and Y Axis Shows Precision Values

JSM

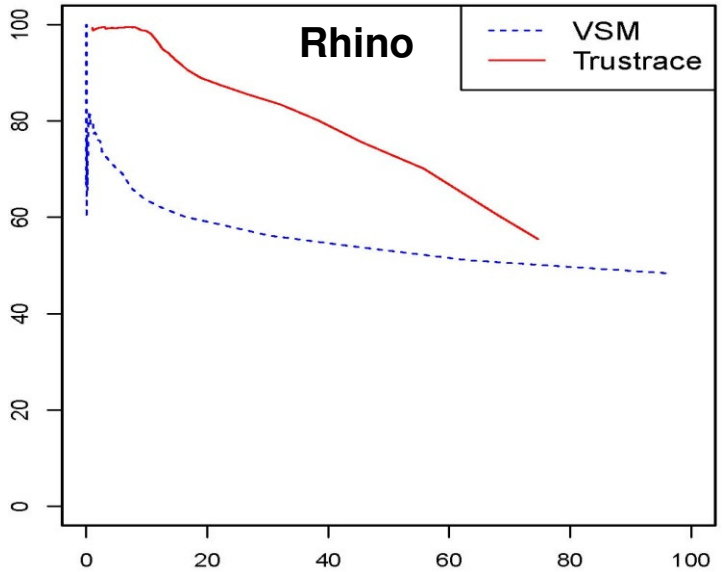


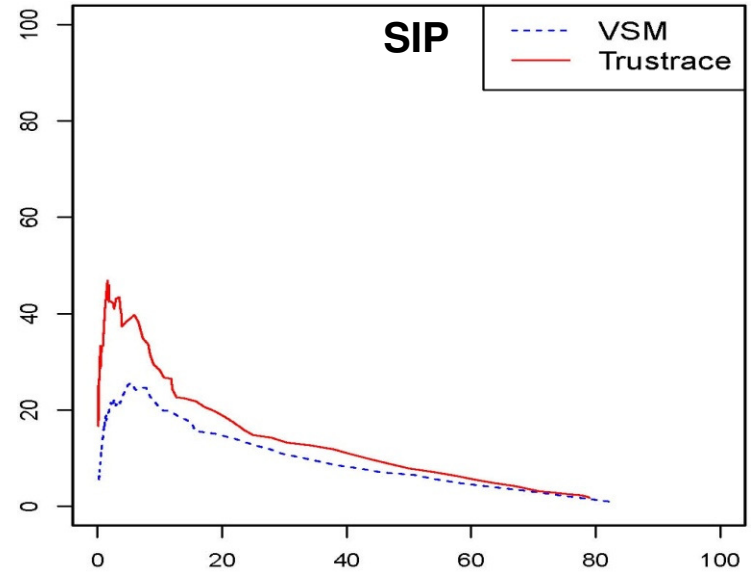
P<0.05



RQ1 – **Histrace** Provides Better **Accuracy** Than VSM and JSM

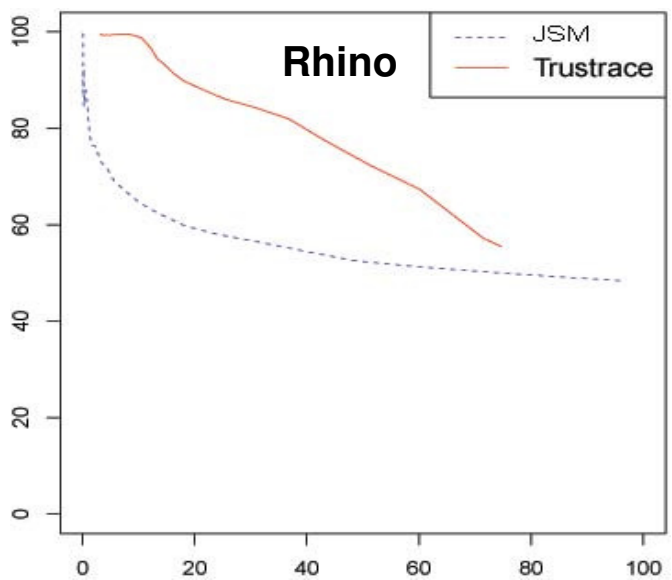
VSM

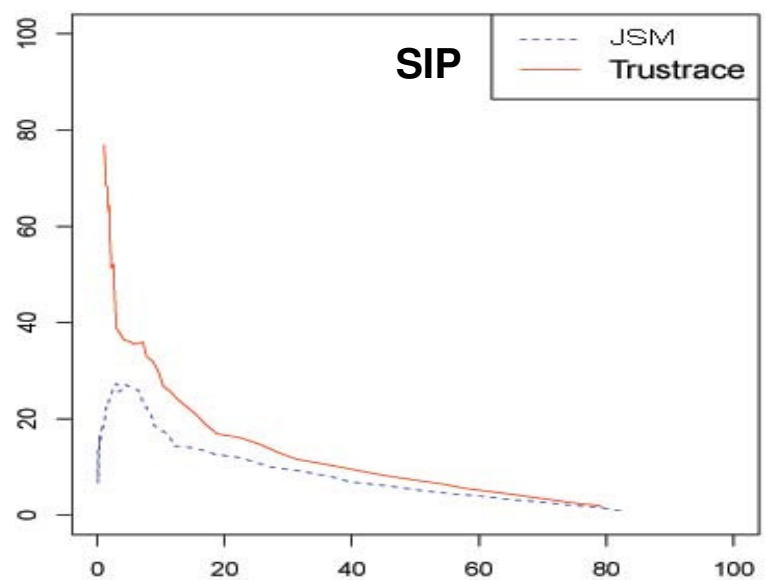




X Axis Shows Recall and Y Axis Shows Precision Values

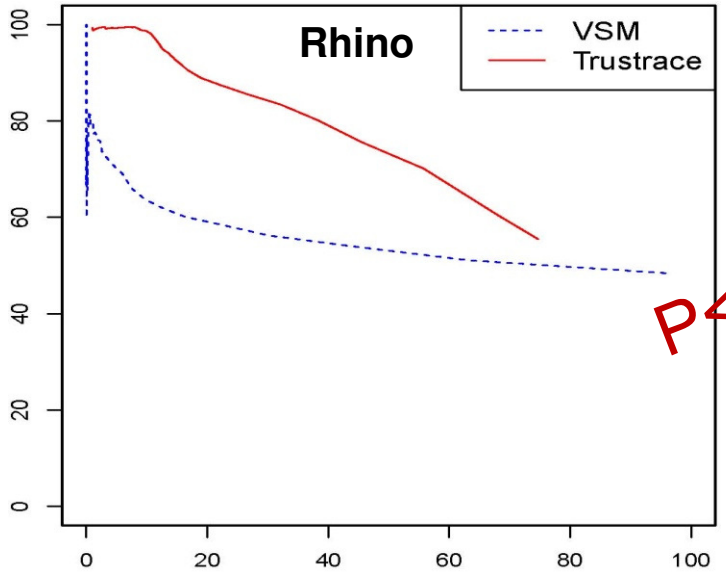
JSM



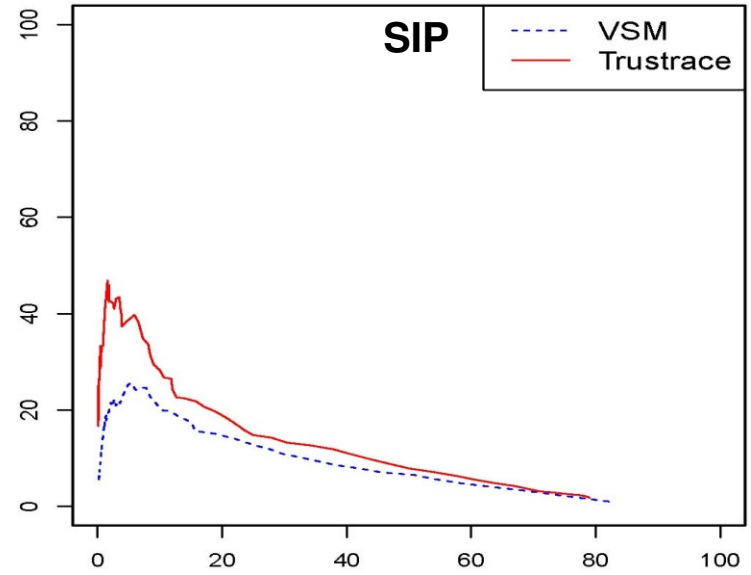


RQ1 – **Histrace** Provides Better **Accuracy** Than VSM and JSM

VSM

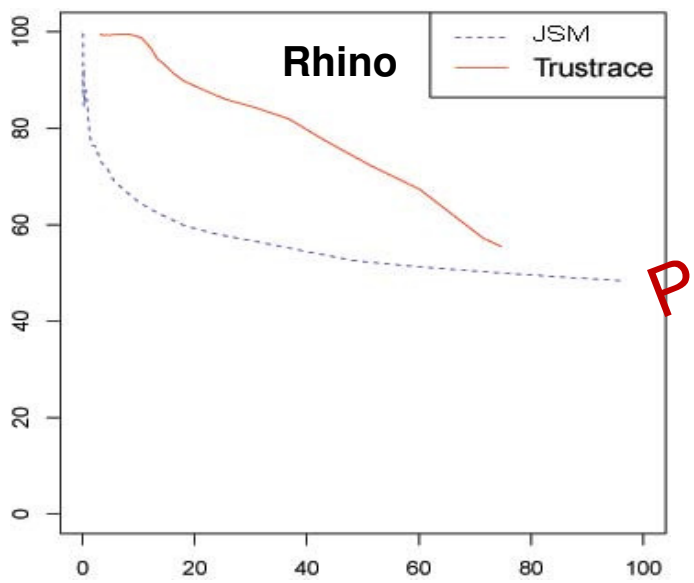


P<0.05

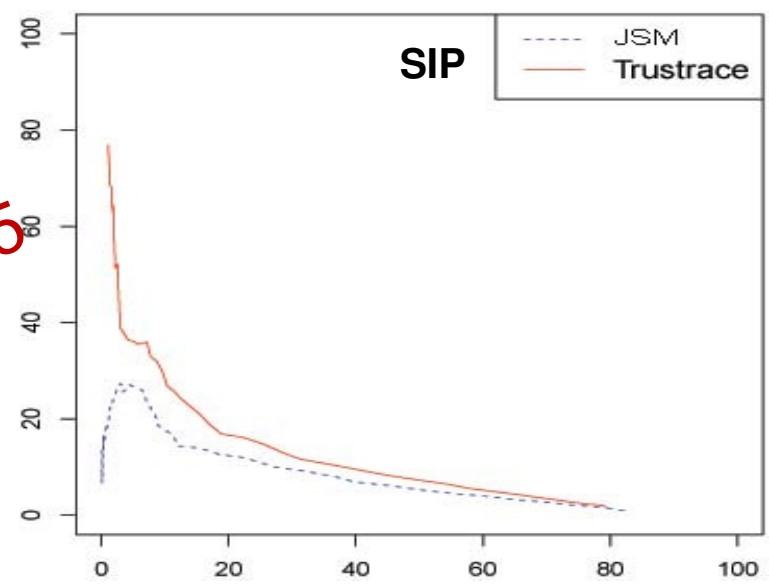


X Axis Shows Recall and Y Axis Shows Precision Values

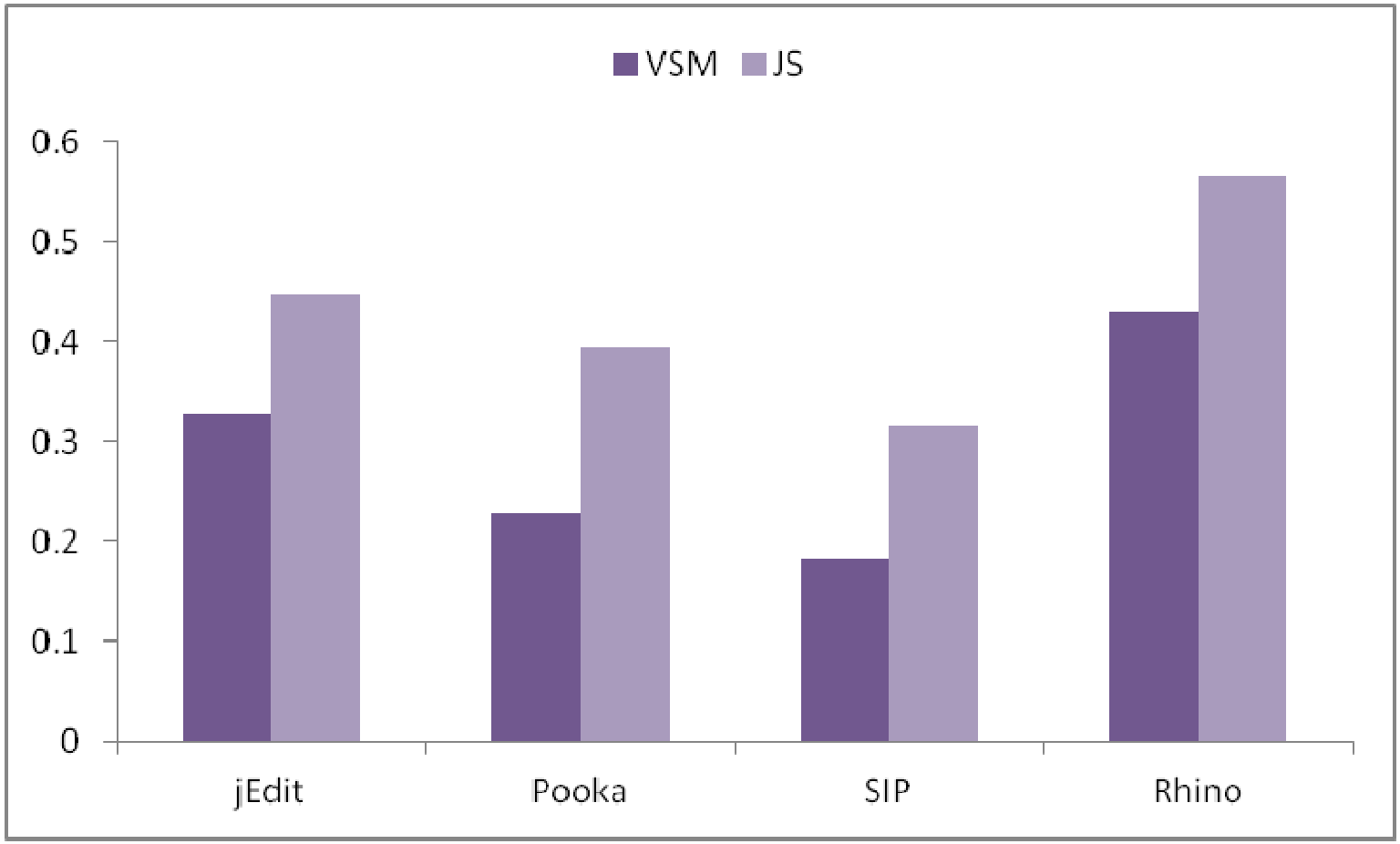
JSM



P<0.05

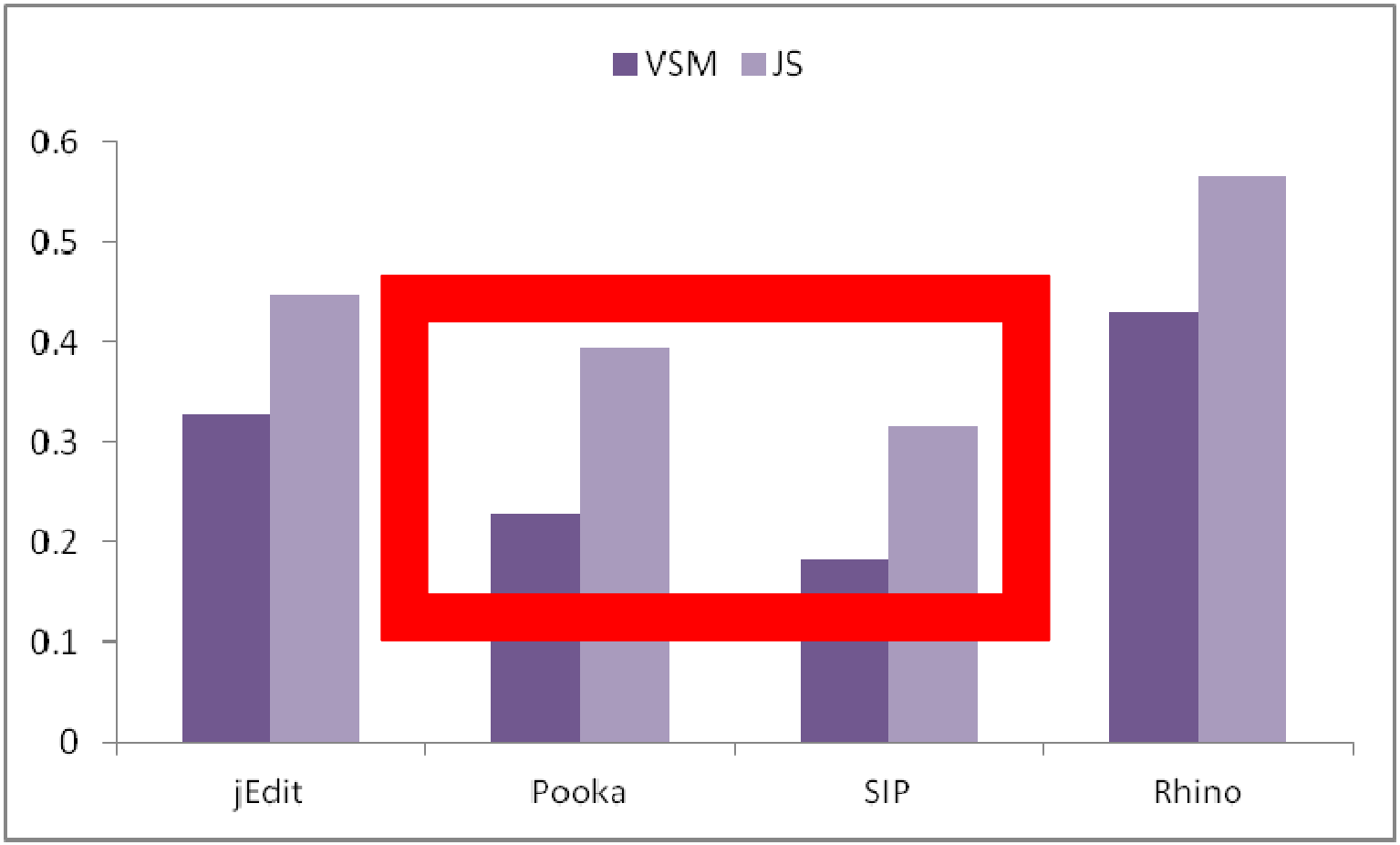


Datasets' Quality Analysis



Y axis shows the % of similarity between requirements and source code

Datasets' Quality Analysis



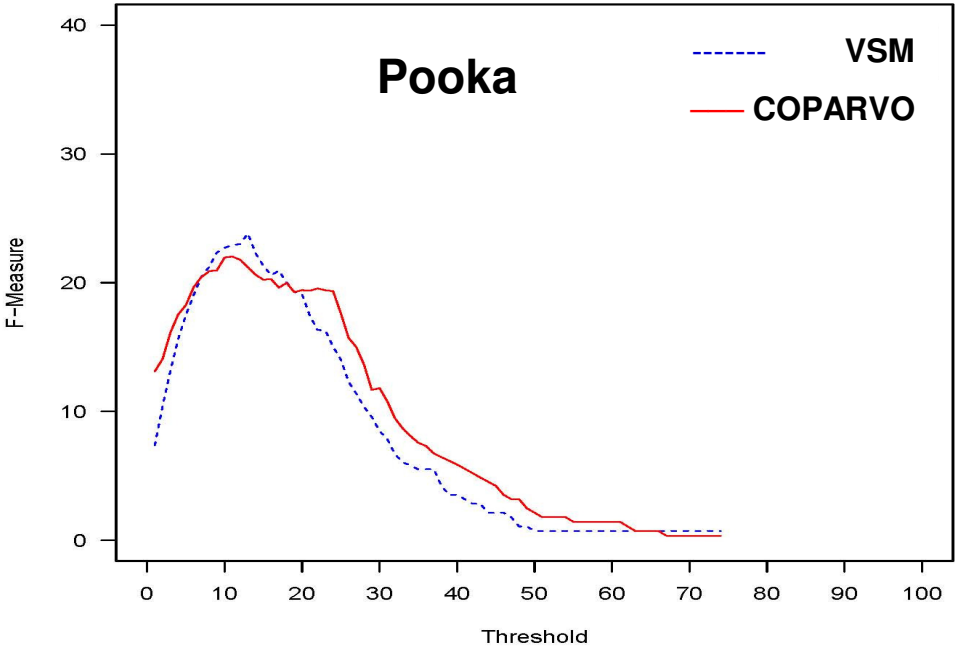
Y axis shows the % of similarity between requirements and source code

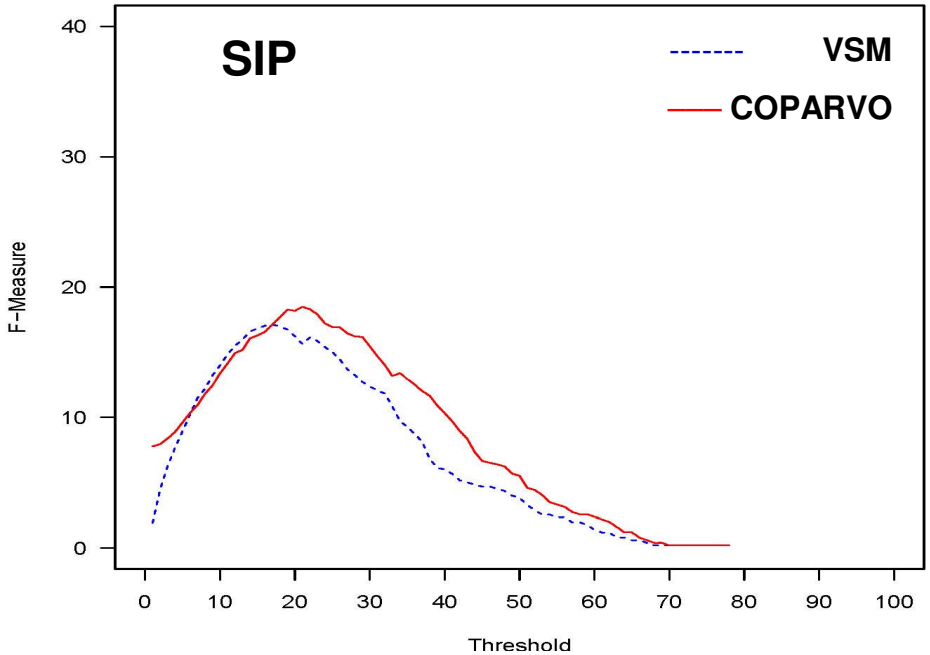
Empirical Evaluation

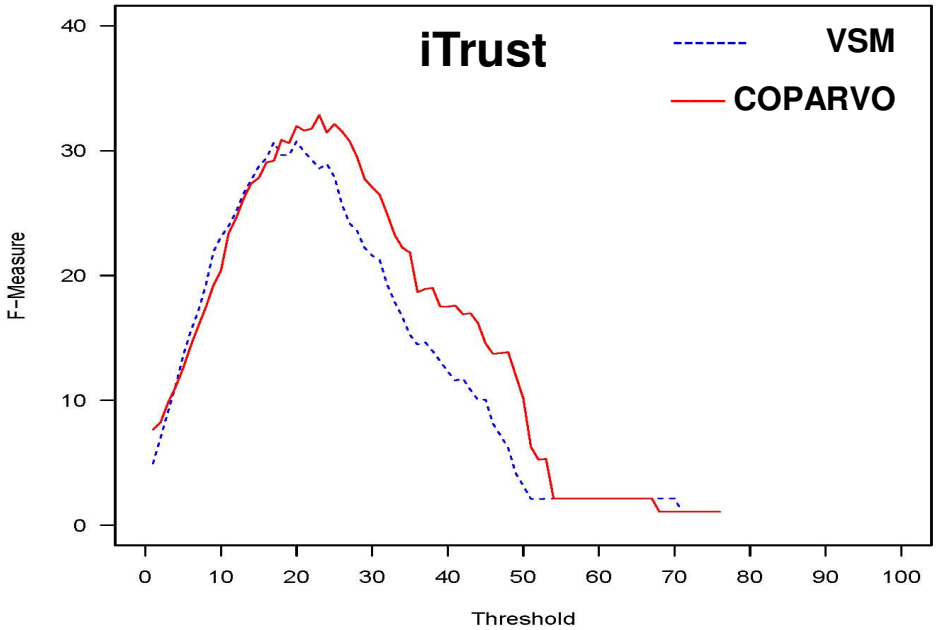
	Trumo	DynWing	Trumo (Ranker)	Static Weight	PCA-based Weights	Voting	JSM	LSI	VSM
Histrace	✓	✓		✓	✓		✓		✓
Partrace						✓			✓
BCRTrace			✓	✓				✓	✓

COPARVO

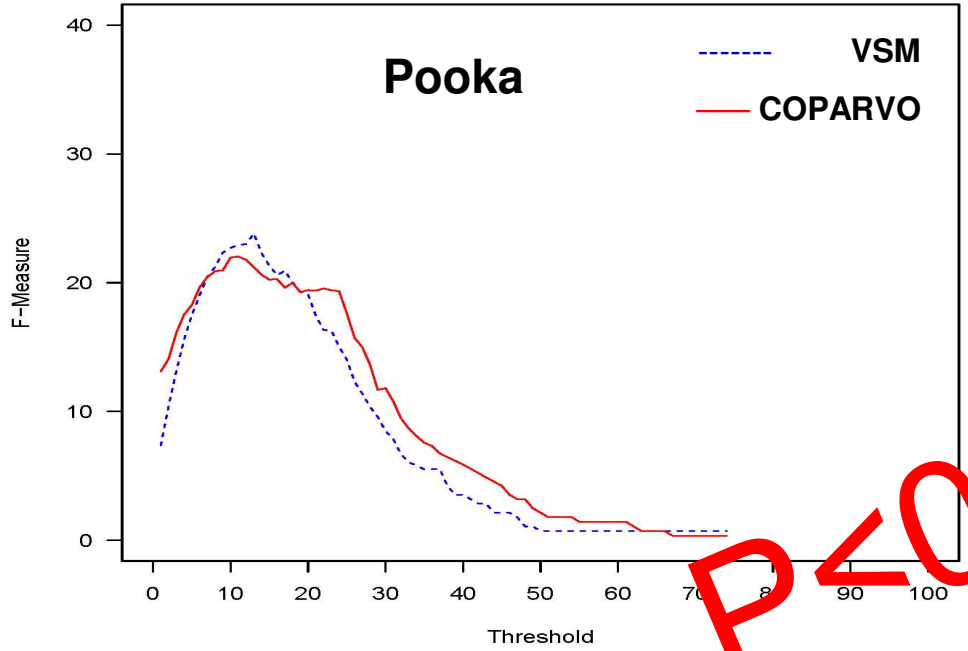
RQ1 – Partrace Improves the Accuracy at Almost All Threshold Points

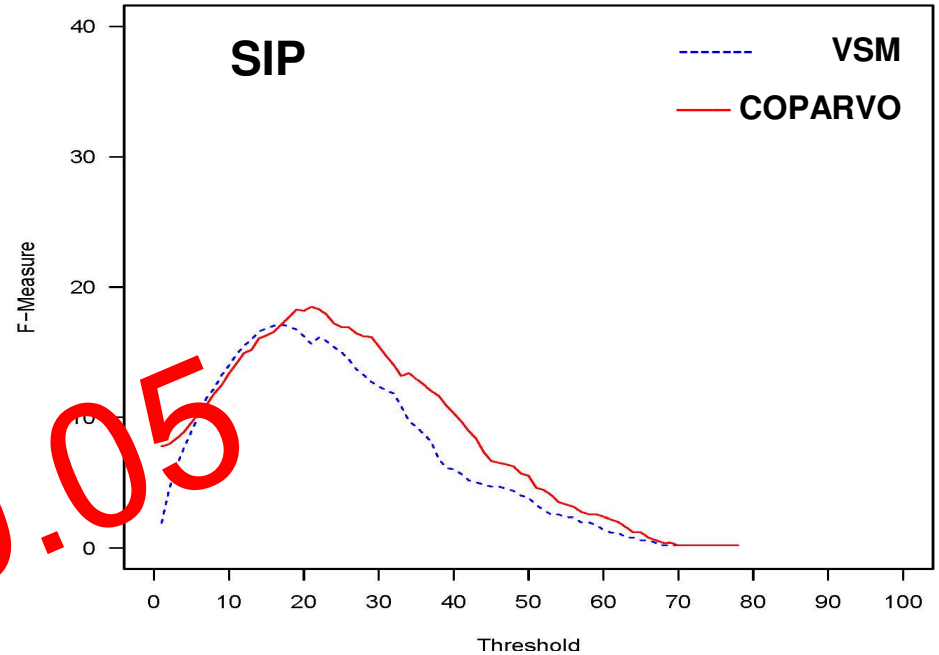


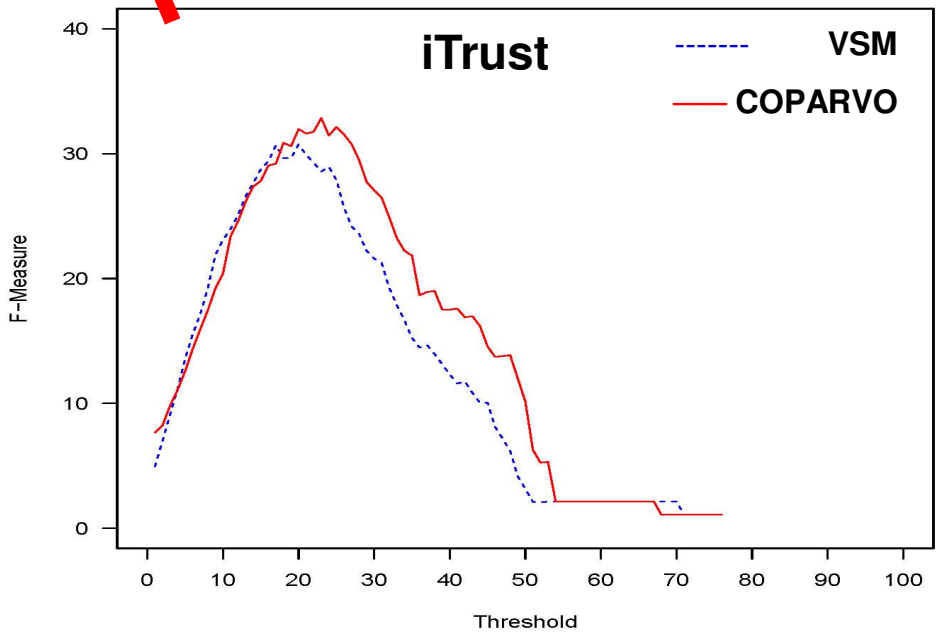




RQ1 – Partrace Improves the Accuracy at Almost All Threshold Points

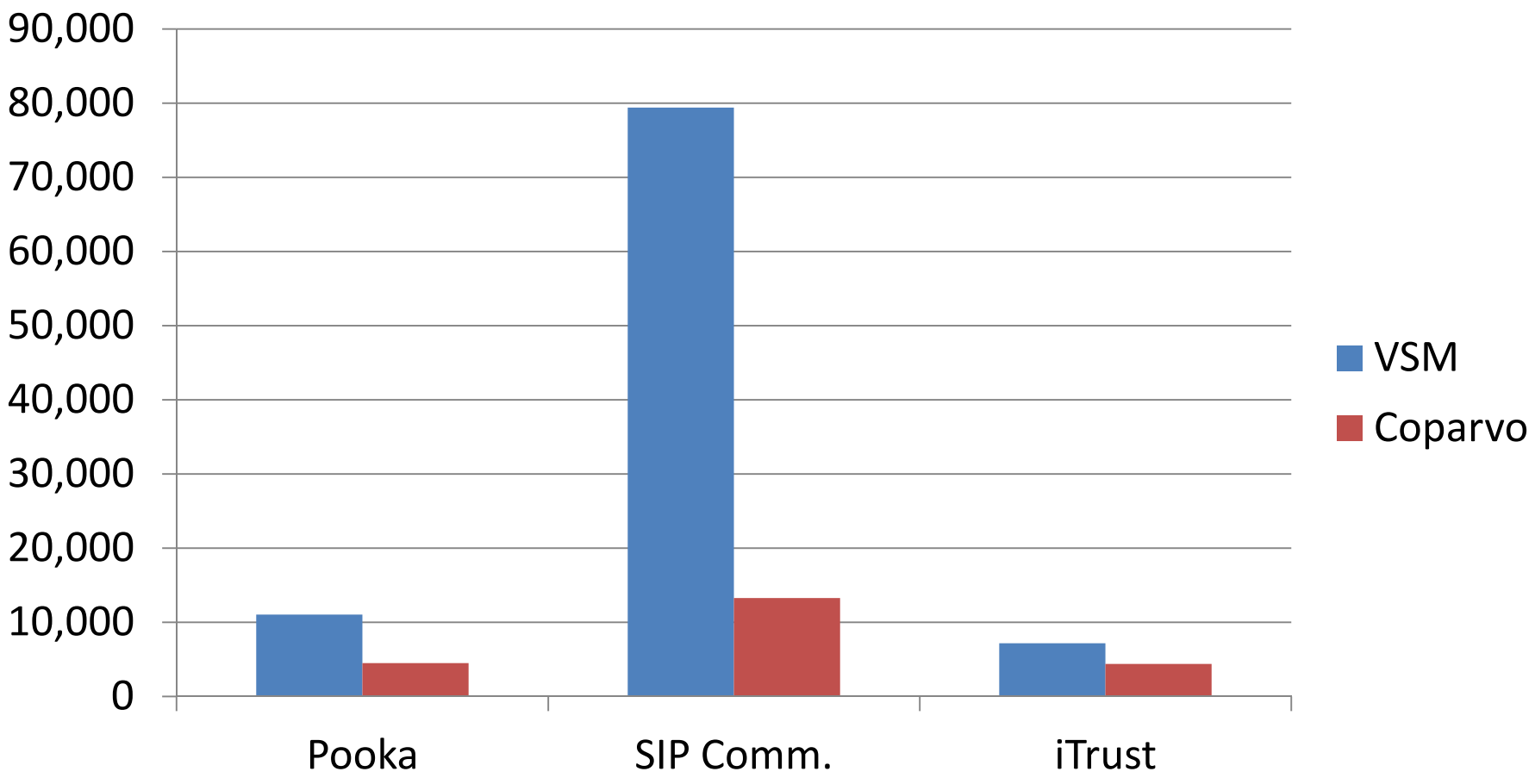






$P < 0.05$

Effort Analysis

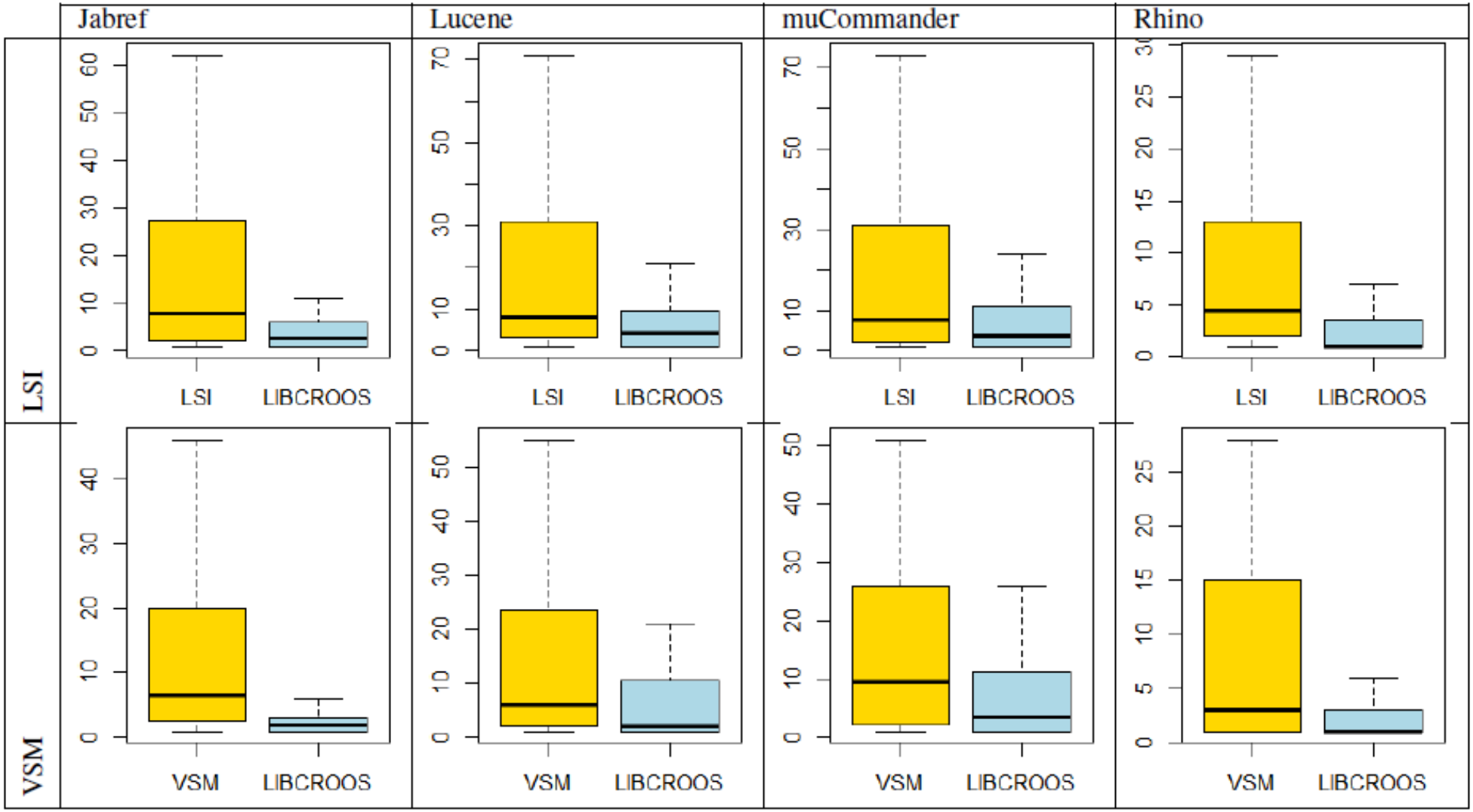


Empirical Evaluation

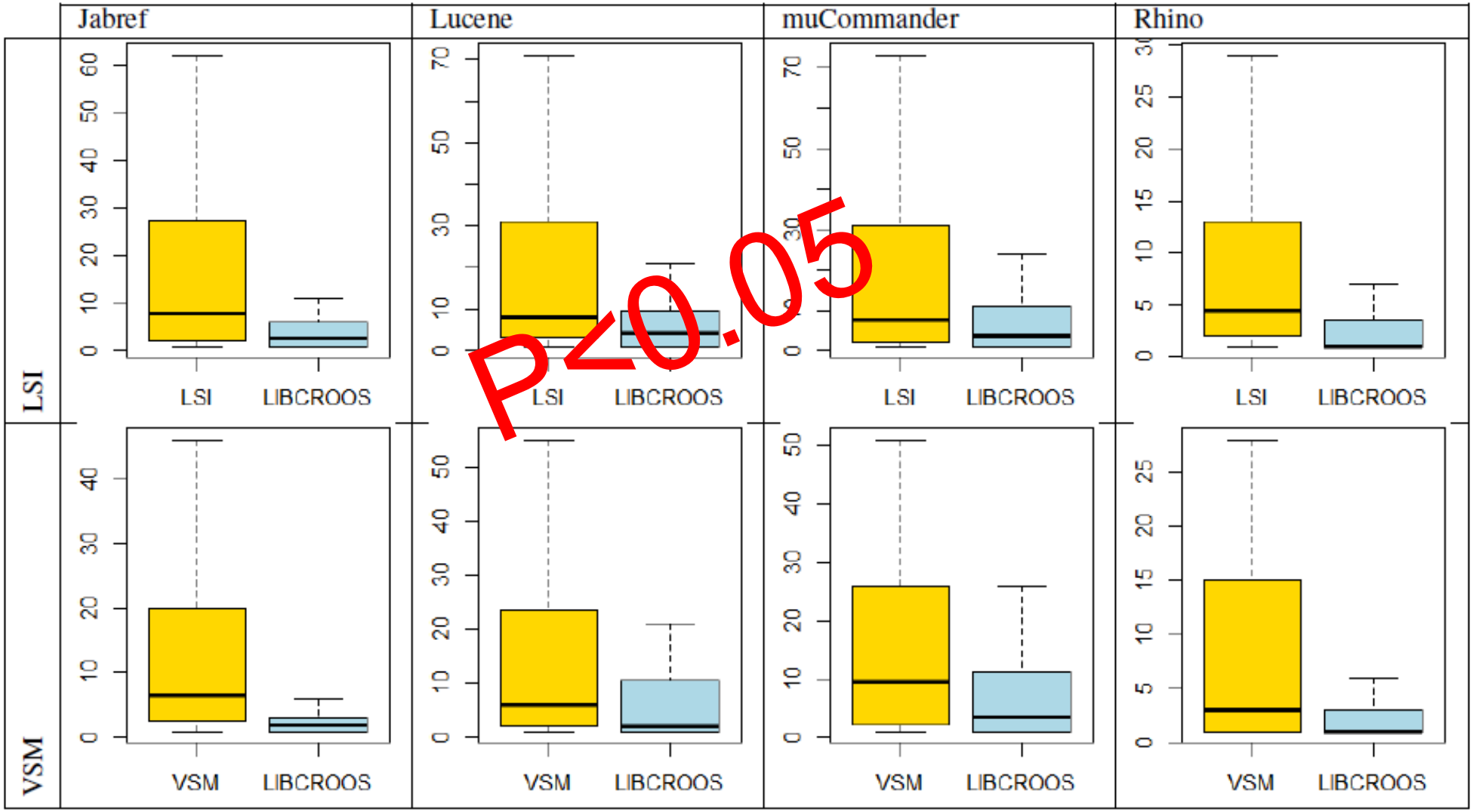
	Trumo	DynWing	Trumo (Ranker)	Static Weight	PCA-based Weights	Voting	JSM	LSI	VSM
Histrace	✓	✓		✓	✓		✓		✓
Partrace						✓			✓
BCRTrace			✓	✓				✓	✓

LIBCROOS

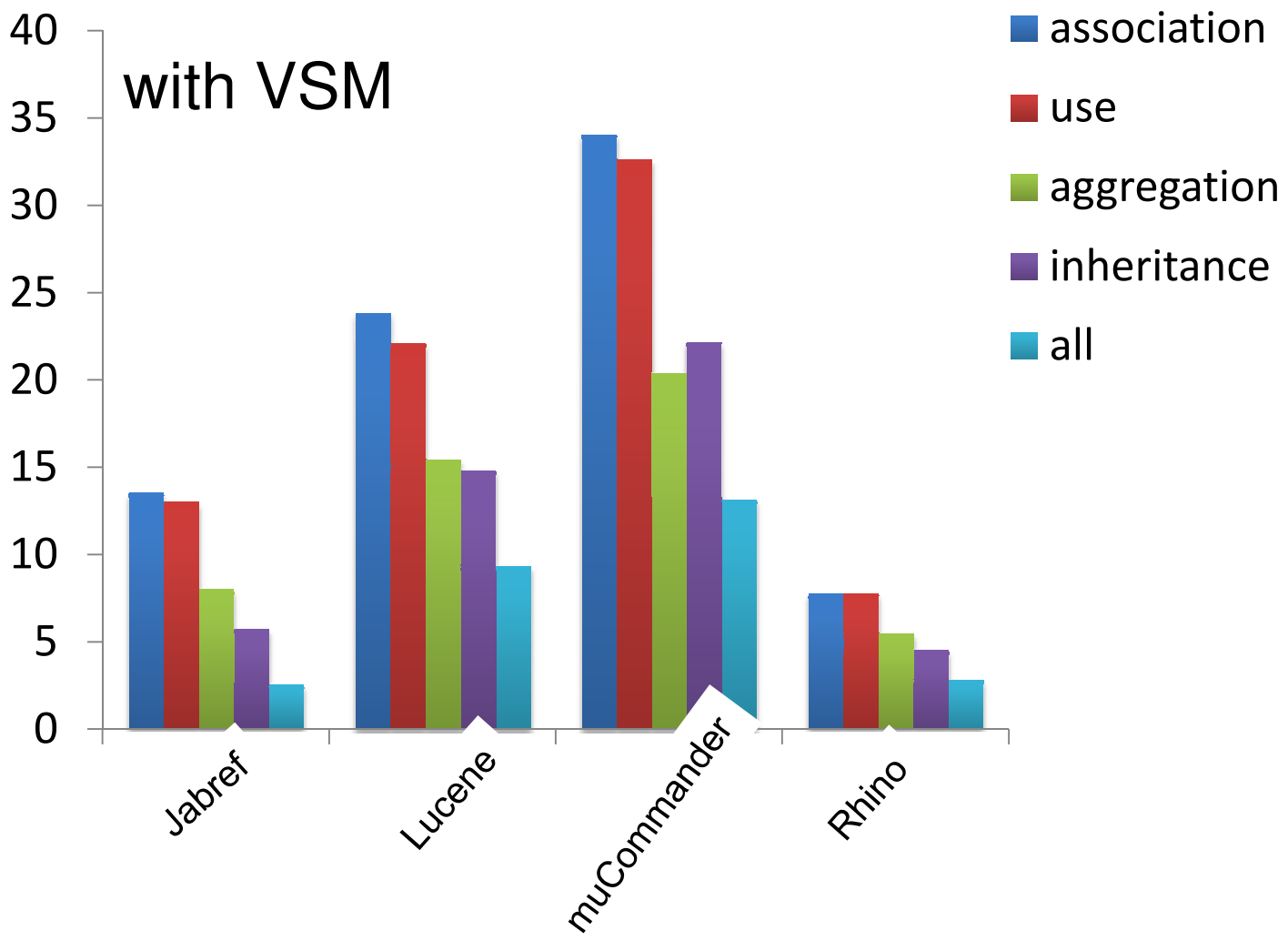
RQ2: BCRTrace **Consistently** ranks the Buggy Classes **Lower**



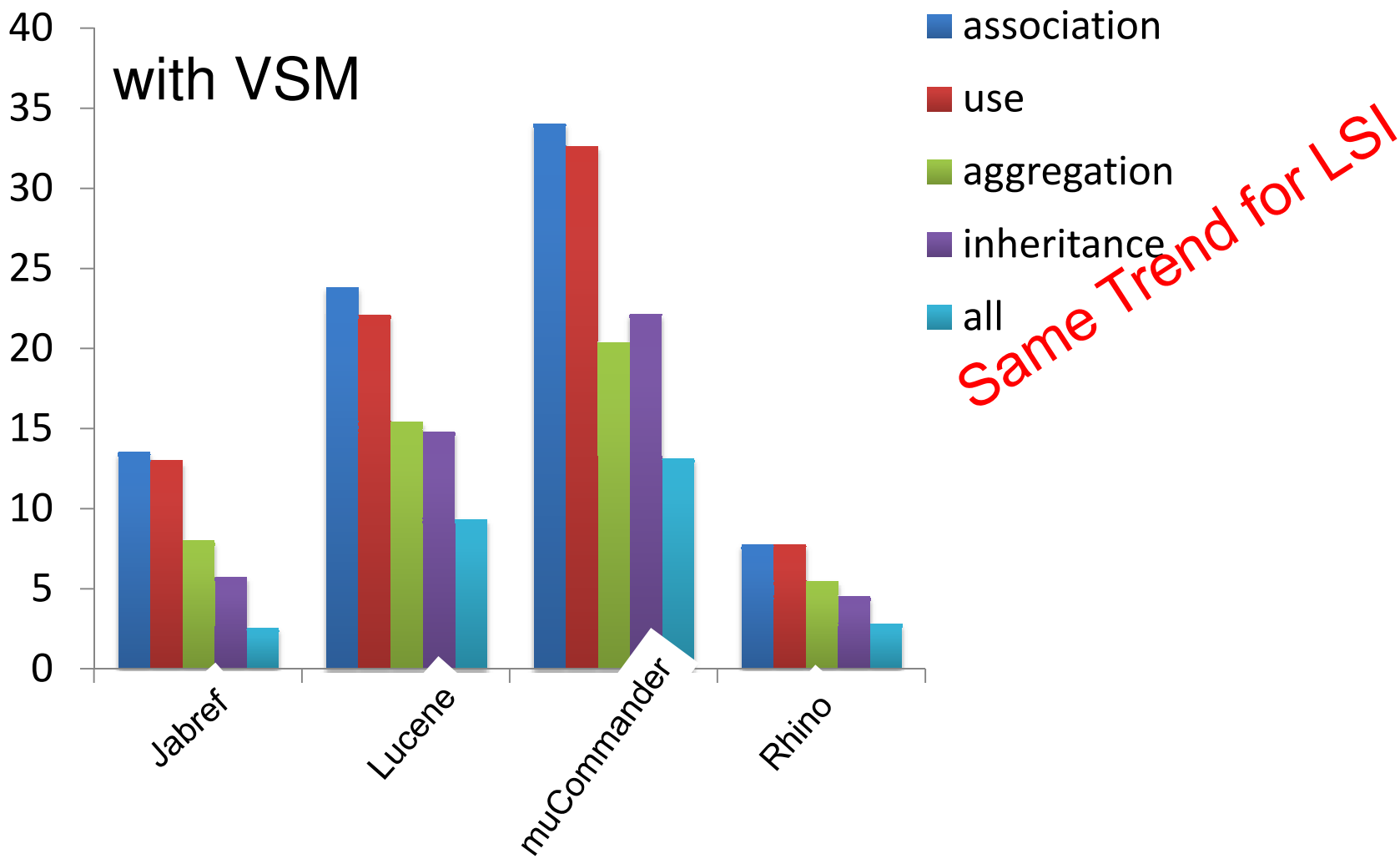
RQ2: BCRTrace **Consistently** ranks the Buggy Classes **Lower**



Static Relationships Analysis

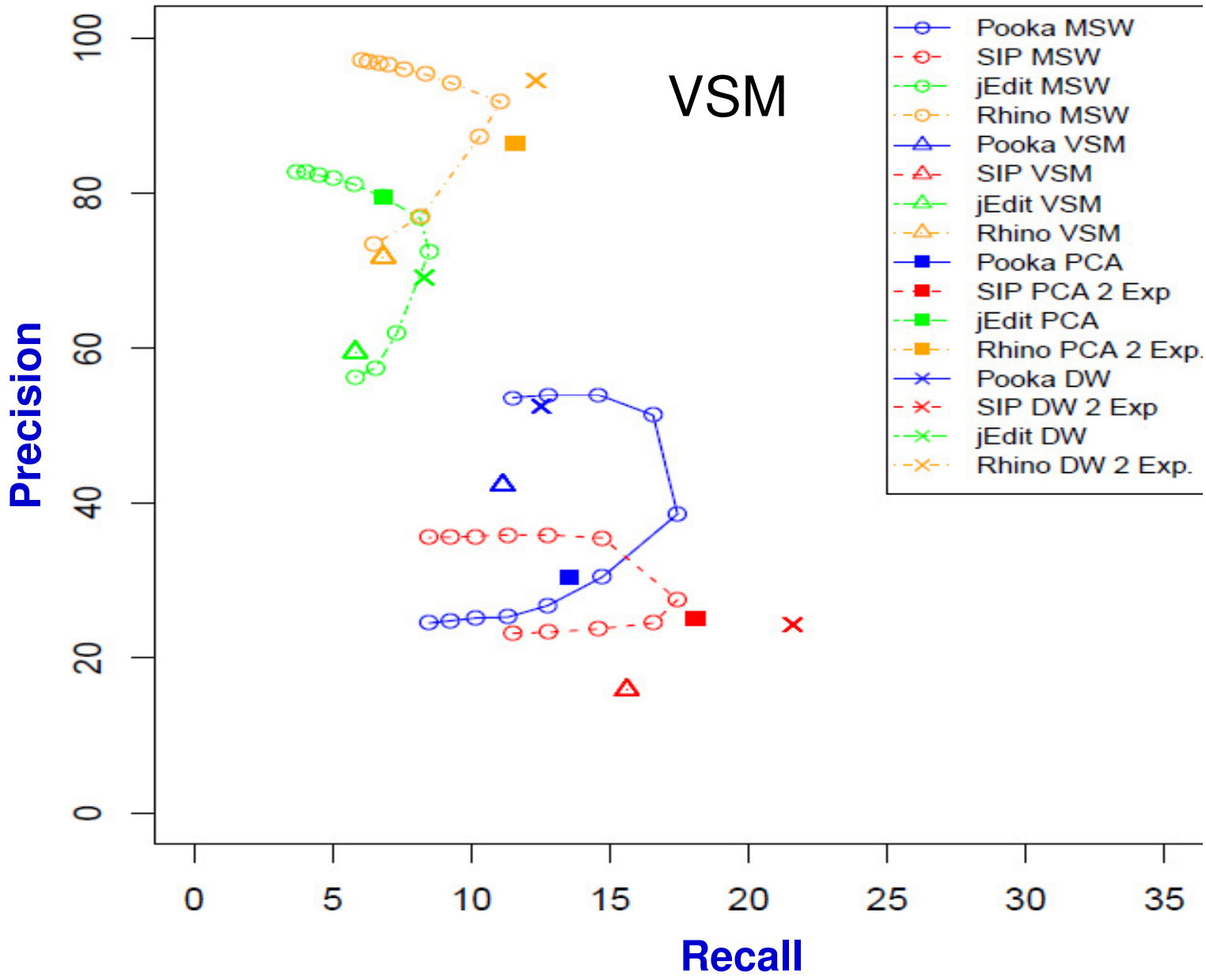


Static Relationships Analysis

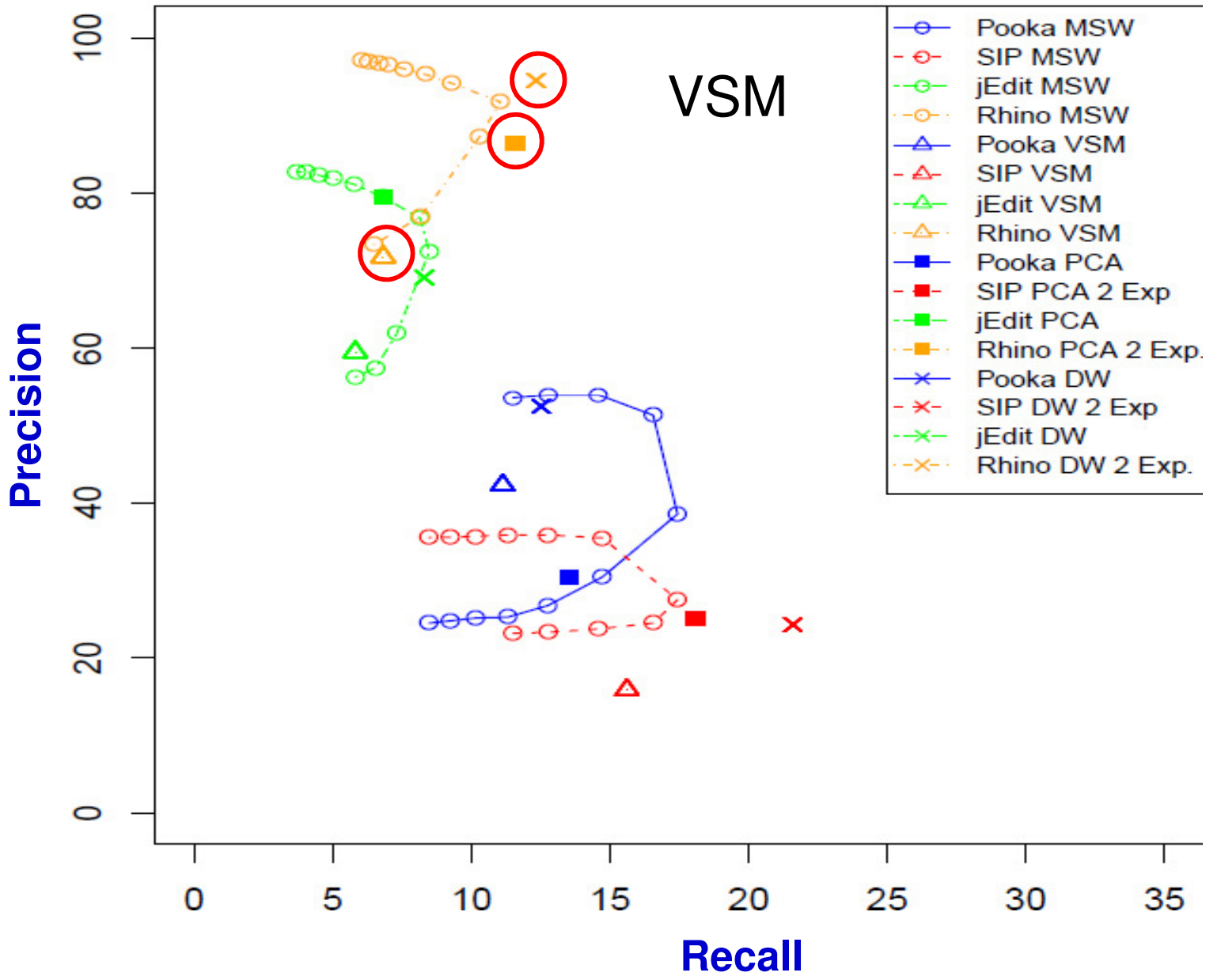


RQ3 – DynWing **Automatically Assigns Weights** to Different Experts

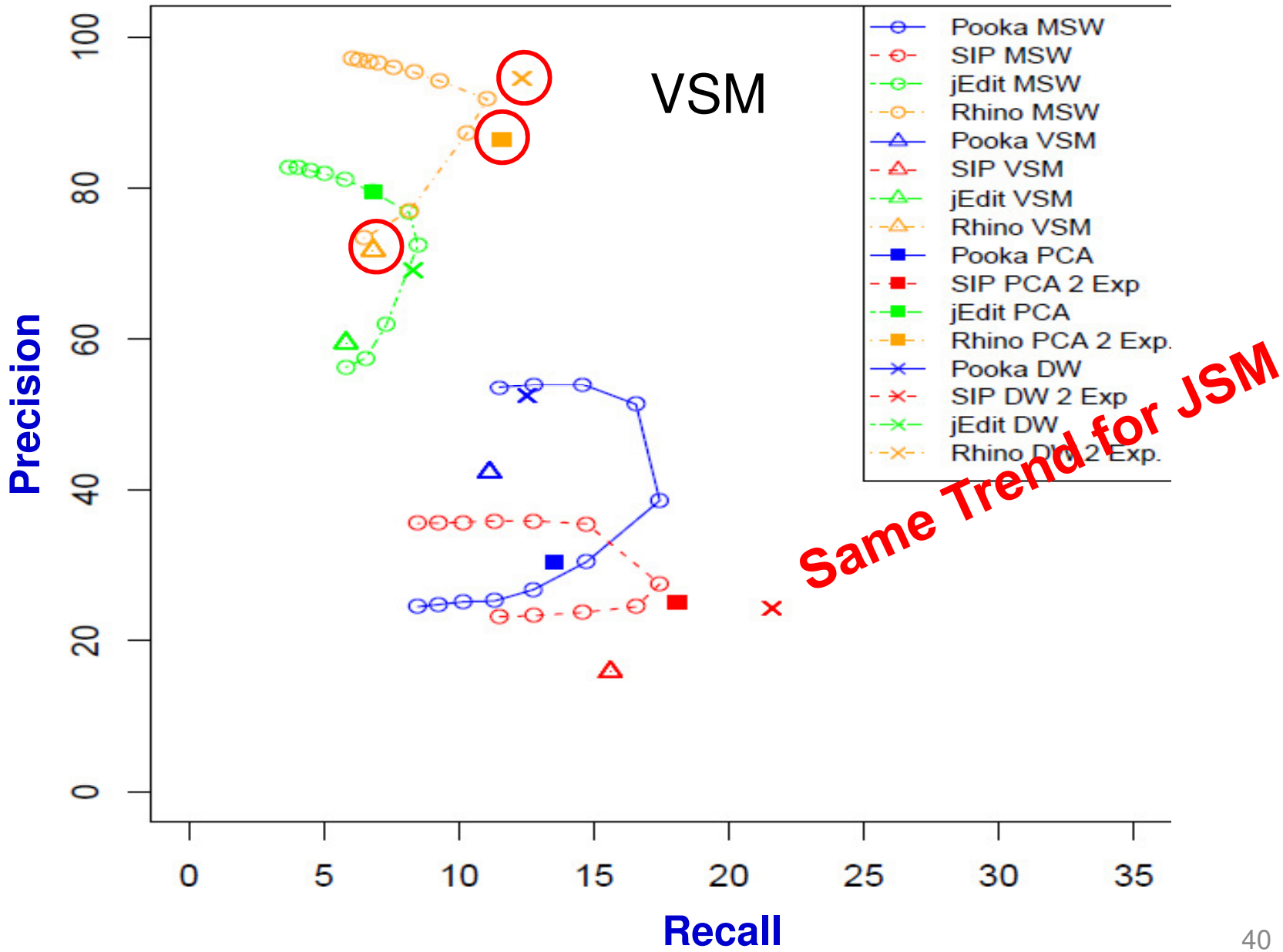
RQ3 – DynWing **Automatically Assigns Weights** to Different Experts



RQ3 – DynWing **Automatically Assigns Weights** to Different Experts



RQ3 – DynWing **Automatically Assigns Weights** to Different Experts



Empirical Studies' Results

- Trumo Model is a general model and can be used for other software maintenance tasks, e.g., bug location
- DynWing automatically assign weights
- Combining experts with weights provide better results than without, i.e., voting

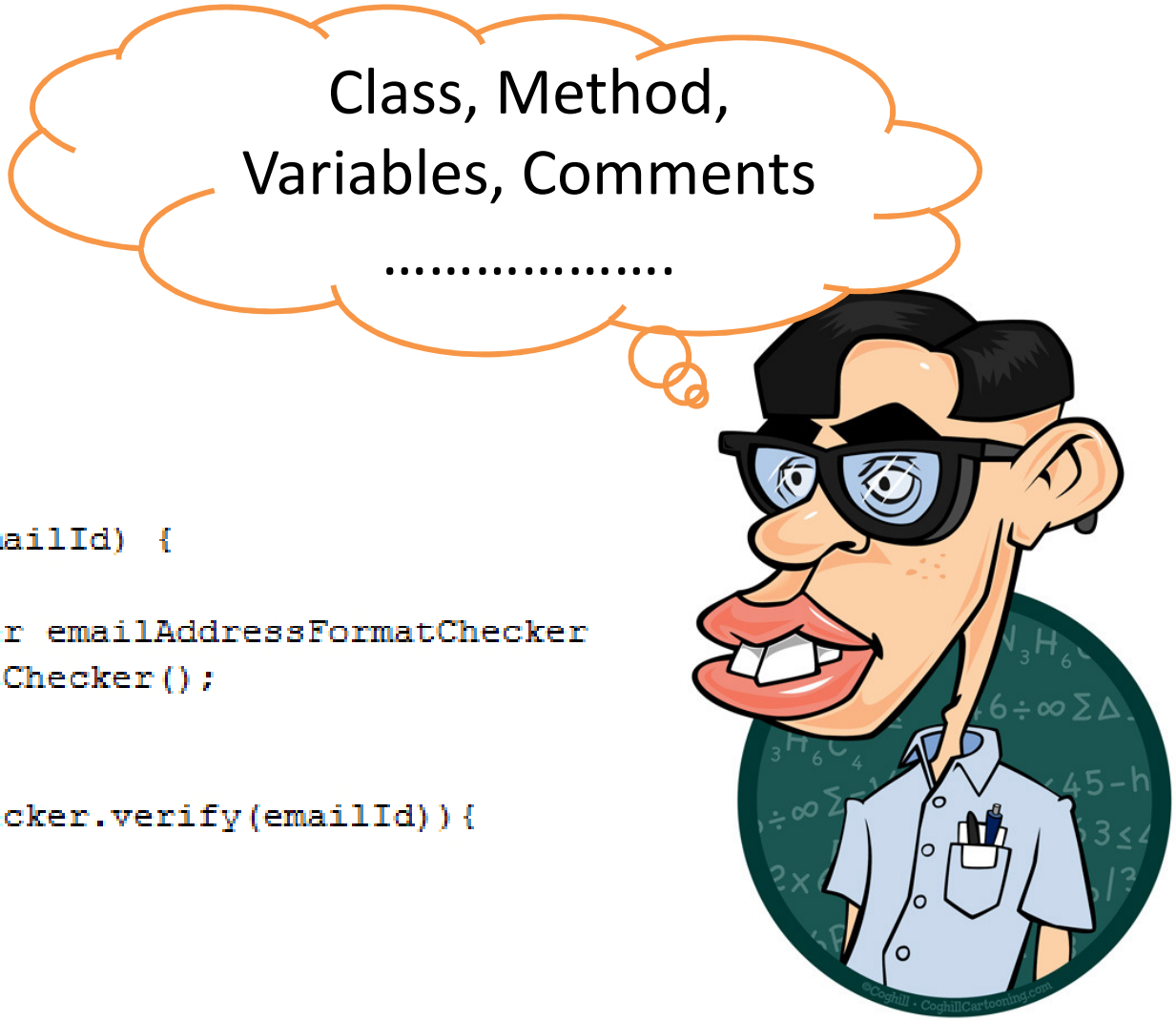
Alert!

- What if we do not have:
  - Software repositories
  - All source code partitions
  - Static Class Relationships

Developers' Knowledge

What the Developer Really Saw

```
public class SendEmail {

    SendEmail (){
        ...................
        ...................
    }

    public sendOutEmail(Sring emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker
        = new EmailAddressFormatChecker();
        int status;

        if(emailAddressFormatChecker.verify(emailId)){
            ...................
            ...................
        }

    }

}
```



Developer

Observing Developers Using Eye-Tracker

- Facelab by Seeing Machine
  - Built-in cameras
  - Infrared pad
  - Monitor screen

Eye-Tracker Study

- RQ1: What Source Code Entities (SCEs) do Developers Value the Most?

- RQ2: Can we Improve IR Techniques by making them Aware of the Developers' Interests?

Eye-Tracker Study Design

Statistics	
Total Subjects	26
Requirements	6
Total Source Code Snippet	6

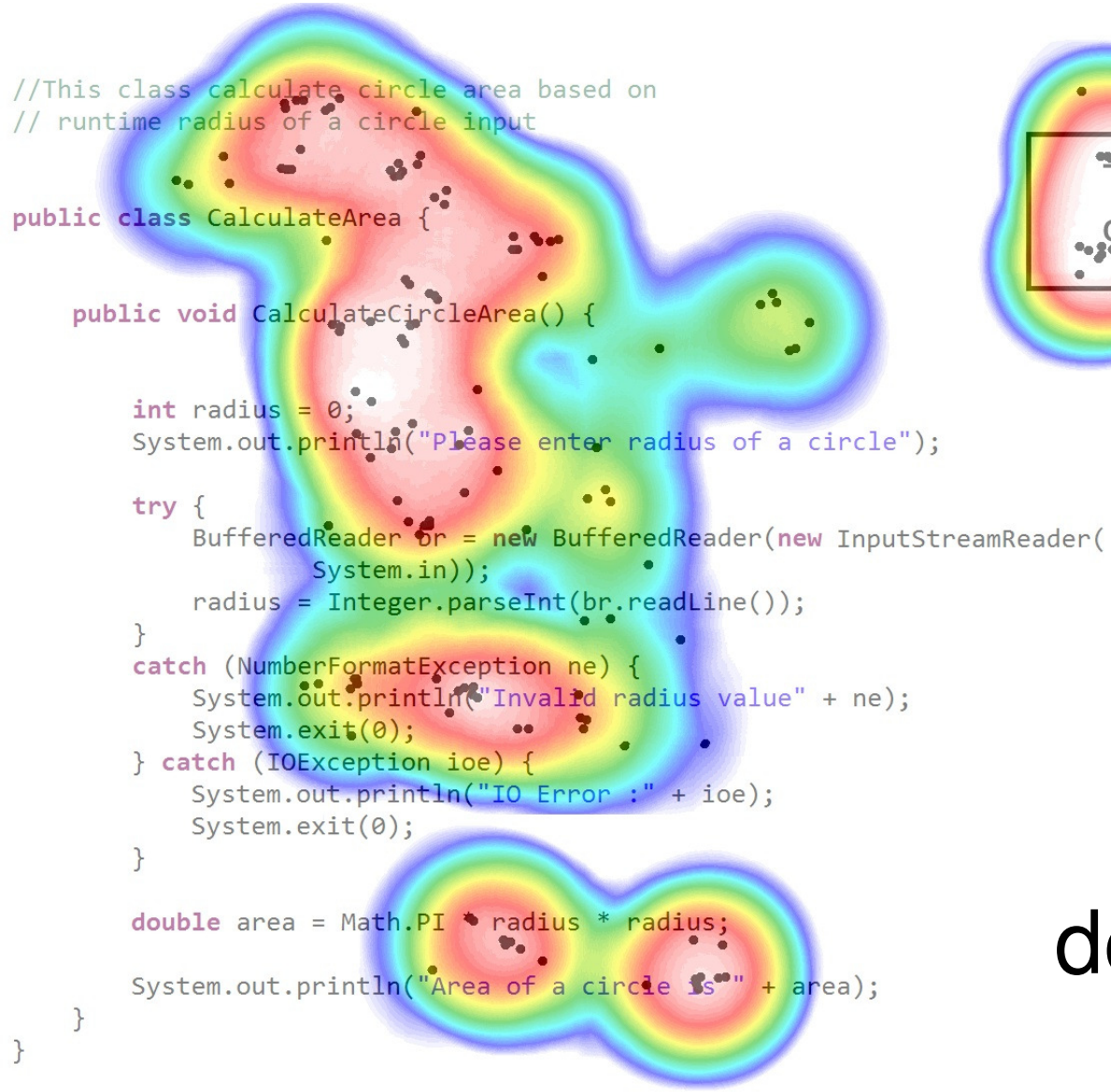
Example Task

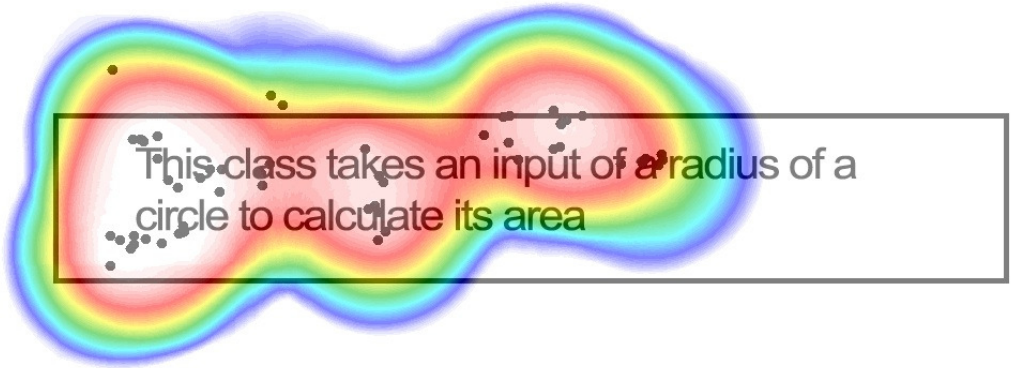
```
//This class calculate circle area based on
// runtime radius of a circle input


public class CalculateArea {


    public void CalculateCircleArea() {


        int radius = 0;
        System.out.println("Please enter radius of a circle");

        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(
                    System.in));
            radius = Integer.parseInt(br.readLine());
        }
        catch (NumberFormatException ne) {
            System.out.println("Invalid radius value" + ne);
            System.exit(0);
        } catch (IOException ioe) {
            System.out.println("IO Error :" + ioe);
            System.exit(0);
        }

        double area = Math.PI * radius * radius;

        System.out.println("Area of a circle is " + area);
    }
}
```

This class takes as input the radius of a circle to calculate its area.

Source Code Sample

Output of Eye-Tracker





eye fixation

==

developer importance

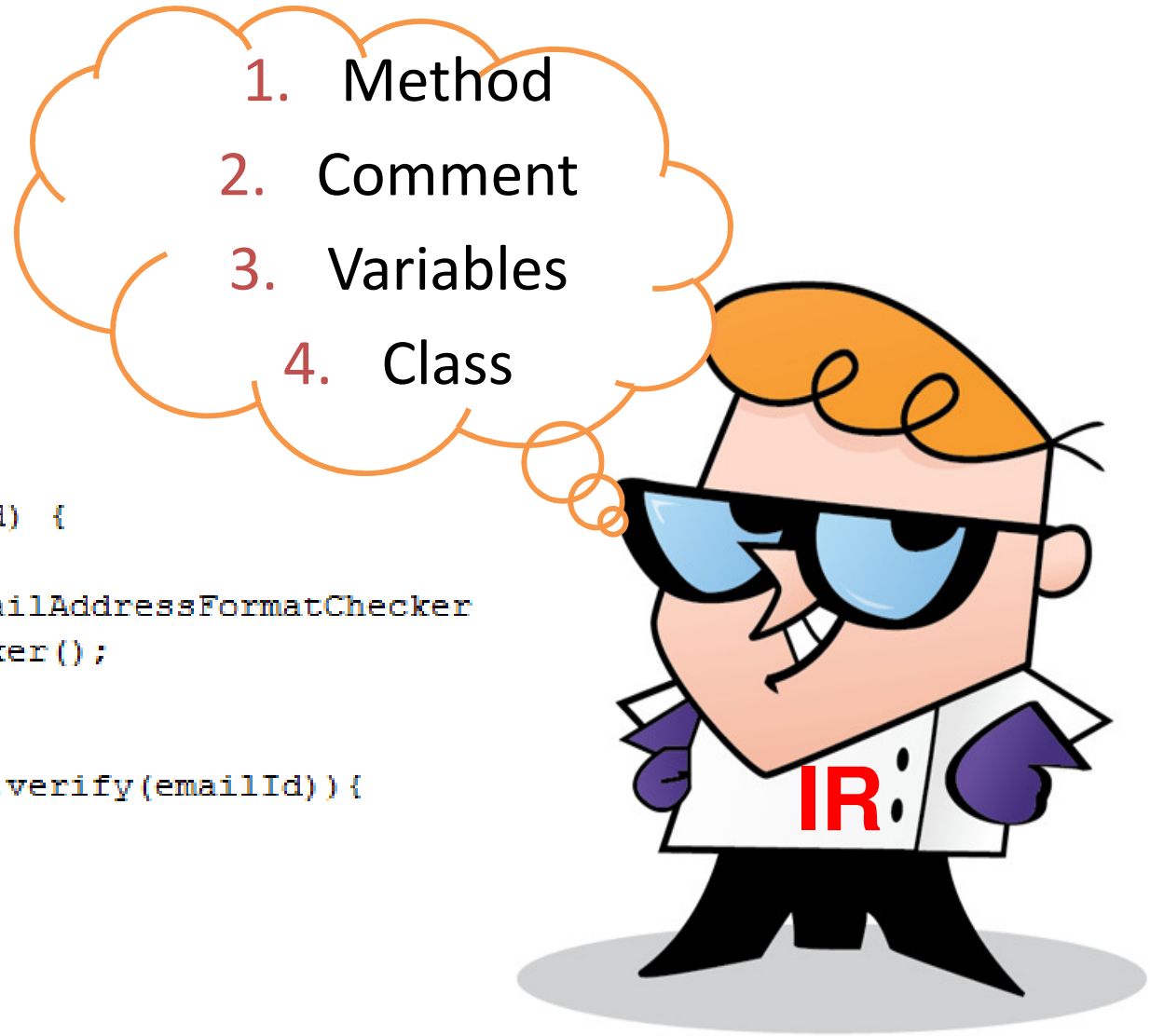
RQ1: What Source Code Entities (SCEs) do Developers Value the Most?

RQ1: What Source Code Entities (SCEs) do Developers Value the Most?

Physical	Conceptual
1. Method Name 2. Comments 3. Variable Name 4. Class Name	1. Domain Related Terms 2. Application Related Terms

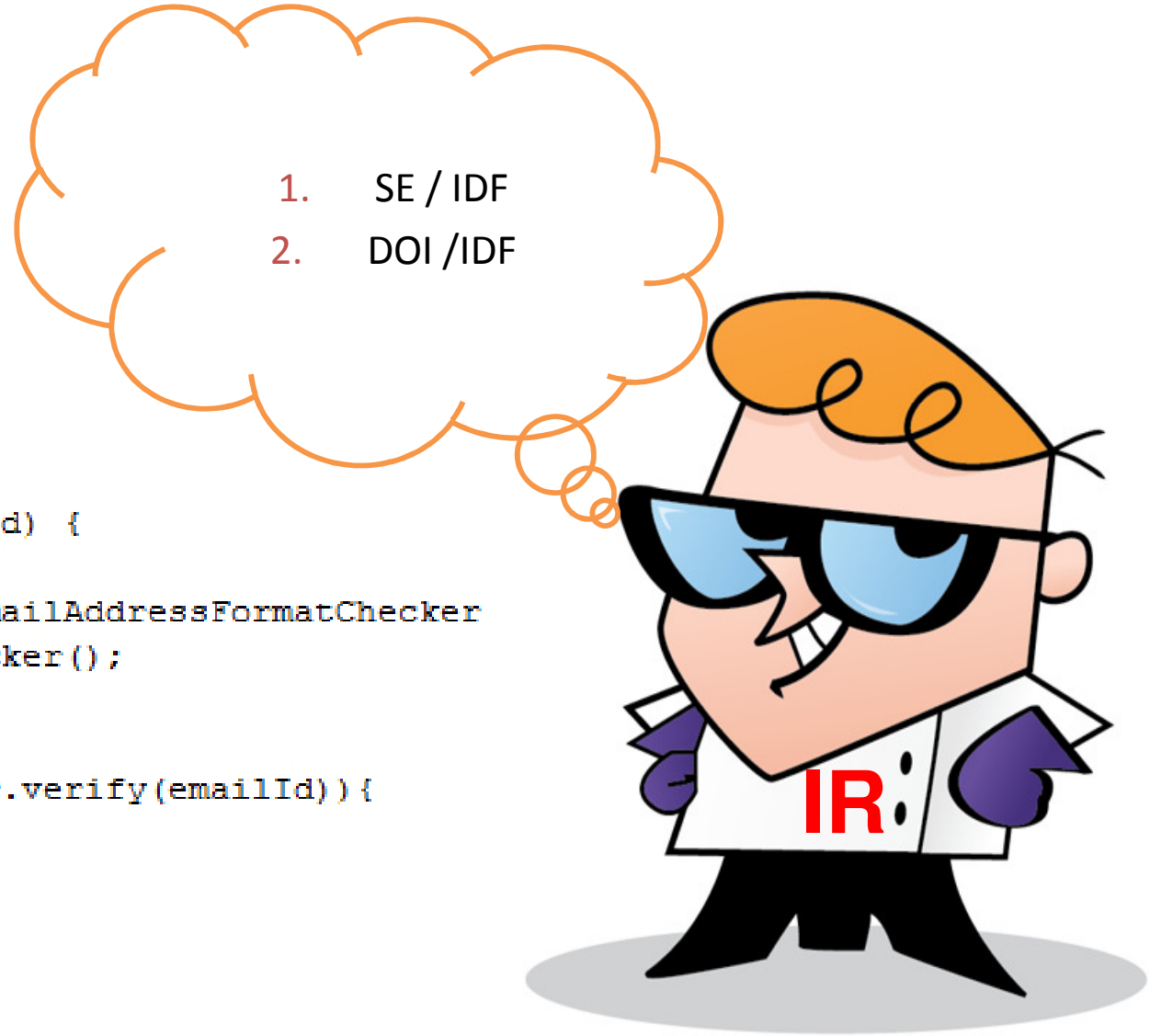
RQ2: Can we Improve IR Techniques by making them Aware of the Developers' Interests?

```
public class SendEmail {

    SendEmail (){
        ..................
        ..................
    }

    public sendOutEmail(Sring emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker
        = new EmailAddressFormatChecker();
        int status;

        if(emailAddressFormatChecker.verify(emailId)){
            ..................
            ..................
        }

    }

}
```



RQ2: Can we Improve IR Techniques by making them Aware of the Developers' Interests?

```
public class SendEmail {

    SendEmail (){
        ...................
        ...................
    }

    public sendOutEmail(Sring emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker
        = new EmailAddressFormatChecker();
        int status;

        if(emailAddressFormatChecker.verify(emailId)){
            ...................
            ...................
        }

    }

}
```

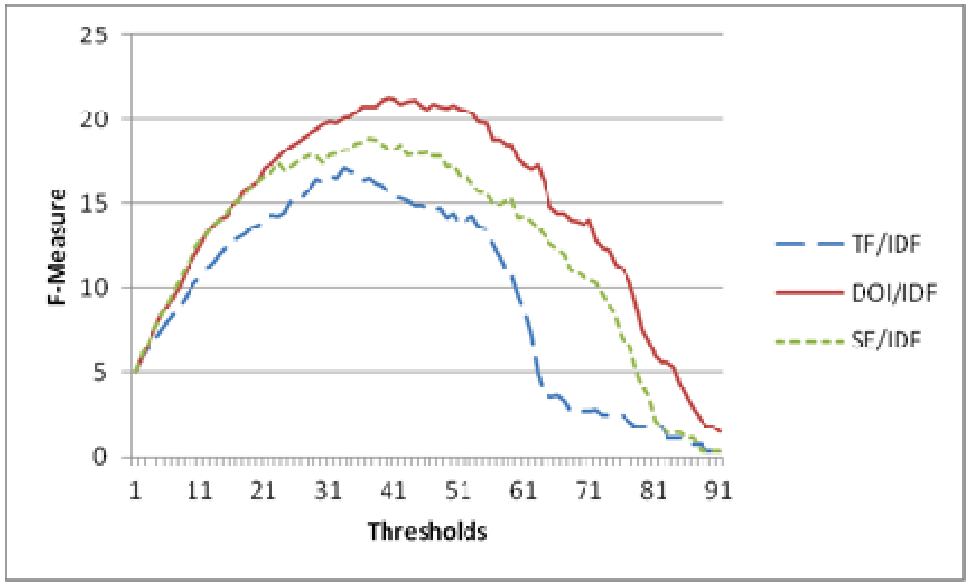


Weighting Scheme

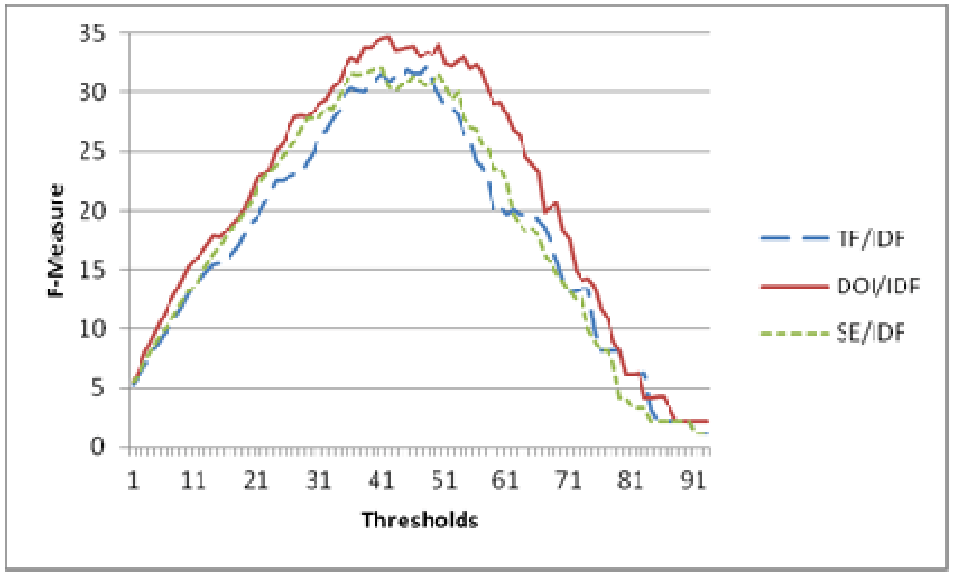
- SE (Source Code Entities): It assigns different weights to all source code entities, e.g., method and class name

- DOI (Domain or Implementation terms): It assigns different weights to domain and implementation

RQ2: Making IR Techniques aware of the Developers' Interests Improves the Accuracy



Pooka



iTrust

Outline

- Introduction
- Related Work
- Creation of Experts
- Combining and Usage of Experts' Opinions
- Assigning Weights to Experts
- Empirical Evaluation
- Conclusion and Future Work

Conclusion

- Using more sources of information improves the accuracy of IR techniques
- Trumo helps to combine the opinions' of experts
- Using experts reduces developers' effort and improves the accuracy of IR techniques
- Adding external information, i.e., software repositories, provides better results than internal information, i.e., source code partitions

Future Work (Short Term)

	Trumo	DynWing	Trumo (Ranker)	Static Weight	PCA-based Weights	Voting	JSM	LSI	VSM
Histrace	✓	✓	✓	✓	✓	✓	✓	✓	✓
Partrace	✓	✓	✓	✓	✓	✓	✓	✓	✓
BCRTrace	✓	✓	✓	✓	✓	✓	✓	✓	✓

- Combine Histrace, BCTrace, and Partrace
- Analysing other sources of information, e.g., mailing lists and MyLyn logs, to create experts
- Using Trumo for other software maintenance tasks, e.g., anti-pattern detection

Future Work (Long Term)

- Updating traceability links during software evolution tasks
- Combining design pattern detection and IR techniques to trace non-functional requirements
- Analysing the impact of anti-pattern on IR-based traceability techniques

Publications

Articles in journal and book chapter

- Nasir Ali, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links, IEEE Transactions on Software Engineering (**TSE**), to appear, 2013

- Nasir Ali, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Factors Impacting the Inputs of Traceability Recovery Approaches, chapter 7. Springer, September 2011

Publications

Conference articles

- Nasir Ali, Zohreh Shara, Yann-Gaël Guéhéneuc, and Giuliano Antoniol, ***An Empirical Study on Requirements Traceability Using Eye-Tracking***. In proceedings of the 28th International Conference on Software Maintenance (**ICSM**), September 2012. IEEE Computer Society Press (**Invited to a special issue of the Journal of Empirical Software Engineering (EMSE)**)

- Nasir Ali, Aminata Sabane, Yann-Gaël Guéhéneuc, and Giuliano Antoniol, **Improving Bug Location Using Binary Class Relationships**. In proceedings of the 12th International Working Conference on Source Code Analysis and Manipulation (**SCAM**), September 2012. IEEE Computer Society Press

- Nasir Ali, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. **Requirements Traceability for OO systems by Partitioning Source Code**. In Proceedings of the 18th Working Conference on Reverse Engineering (**WCRE**), October 17-20, 2011. IEEE Computer Society Press

Publications

Conference articles

- Nasir Ali, Wei Wu, Giuliano Antoniol, Massimiliano Di Penta, Yann-Gaël Guéhéneuc, and Jane H. Hayes. **MoMS: Multi-objective Miniaturization of Software**. In proceedings of the 27th International Conference on Software Maintenance (**ICSM**), September 2011. IEEE Computer Society Press

- Nasir Ali, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. **Trust-based Requirements Traceability**. In Proceedings of the 19th International Conference on Program Comprehension (**ICPC**), 22 - 24 June, 2011. IEEE Computer Society Press

- Nasir Ali. **Trustrace: Improving Automated Trace Retrieval Through Resource Trust Analysis**. In Proceedings of the 19th International Conference on Program Comprehension (**ICPC**), 22 - 24 June, 2011. IEEE Computer Society Press

Thesis

Adding more sources of information and combining them with IR techniques could improve the accuracy of IR techniques for requirements traceability

Contributions

Mining Software Repositories

Trust Model

Binary Class Relationships

Partitioning Source Code

Using Developers' Knowledge

Dynamic Weights Calculator

Trumo

$$R2CT_{i,r_j,t_k} = \{(r_j, c_s, \sigma'_i(r_j, t_k)) | c_s \in \delta_{T_i}(t_k) \ \& \ t_k \in T_i\} \quad (1)$$

$$\begin{aligned} Tr = \{(r_j, c_s, \sigma'_i(r_j, t_k)) \quad & | \\ \exists \ t_k \in T_i : (r_j, c_s) \quad & \in \quad \alpha(R2CT_{i,r_j,t_k}) \\ & \& \quad (r_j, c_s) \in \alpha(R2C)\} \end{aligned} \quad (2)$$

Trumo

$$\sigma_i^*(r_j, c_s) = \frac{\sigma(r_j, c_s) + \sum_{l \in TC_i(r_j, c_s)} \phi(l)}{1 + |TC_i(r_j, c_s)|} \qquad (3)$$

$$\begin{aligned} \psi_{r_j, c_s}(Tr) &= \left[\sum_{i=1}^{P} \lambda_i(r_j, c_s) \sigma_i^*(r_j, c_s) \right] \\ &+ \lambda_{P+1}(r_j, c_s) \frac{|Tr(r_j, c_s)|}{\max_{n,m} |Tr(r_n, c_m)|} \end{aligned} \qquad (4)$$

Eye-tracking Experiment Results

Source Code Entities	Average Fixation Time (ms) of All Subjects
Method Name(s)	5701.10
Comments	4542.41
Variable Name(s)	3181.81
Class Name(s)	2317.25

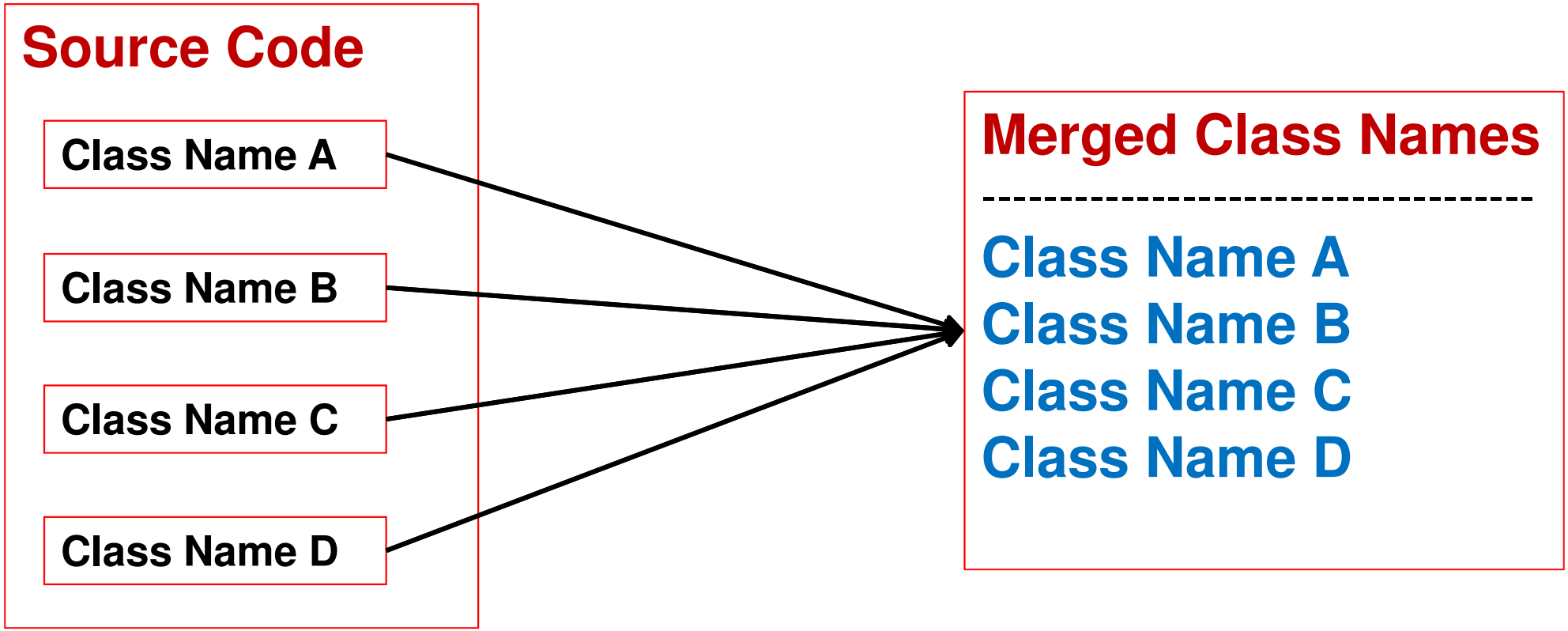
	Average Fixation Time (ms) of All Subjects
Domain (48% of all terms)	4865.30
Implementation (52% of all terms)	1729.80

Creation of Partrace Expert

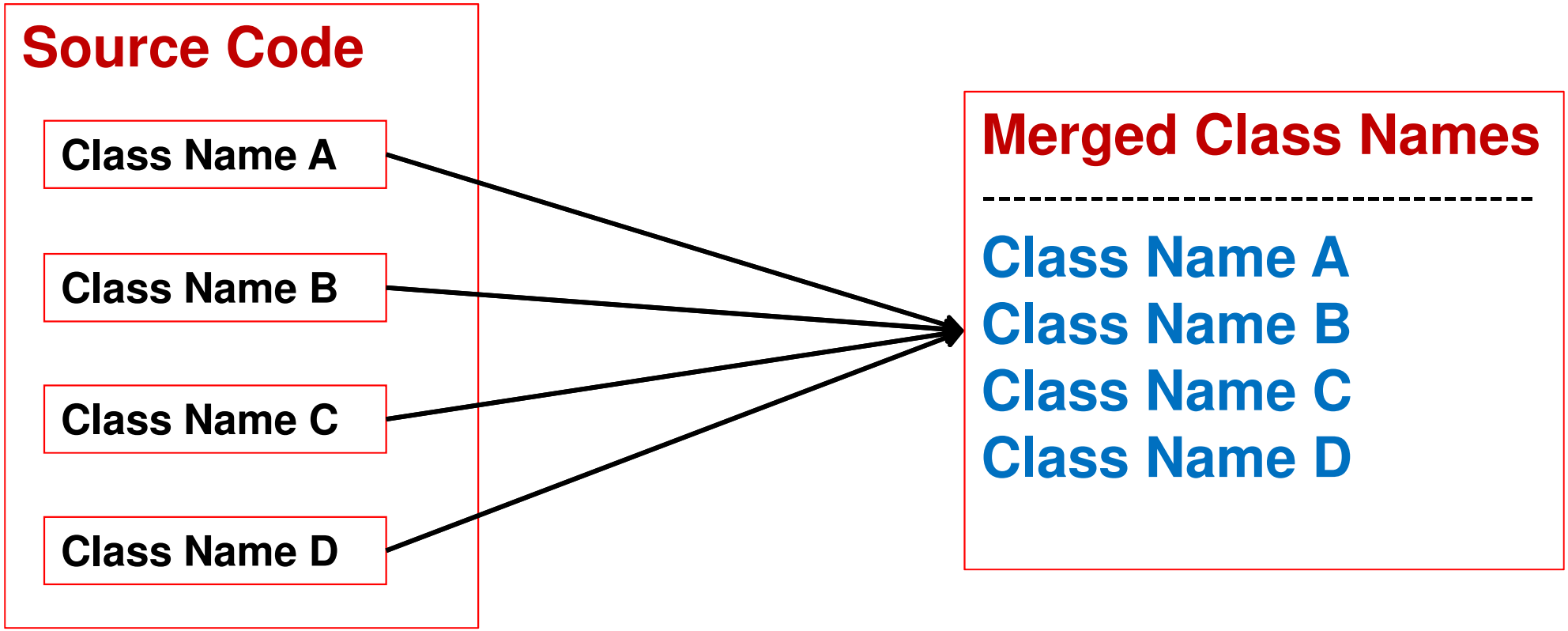
Source Code

- Class Name A
- Class Name B
- Class Name C
- Class Name D

Creation of Partrace Expert



Creation of Partrace Expert



Performed same step for method, variable names, comments, and requirements

Creation of Partrace Expert

Merged Class Names

Merged Method Names

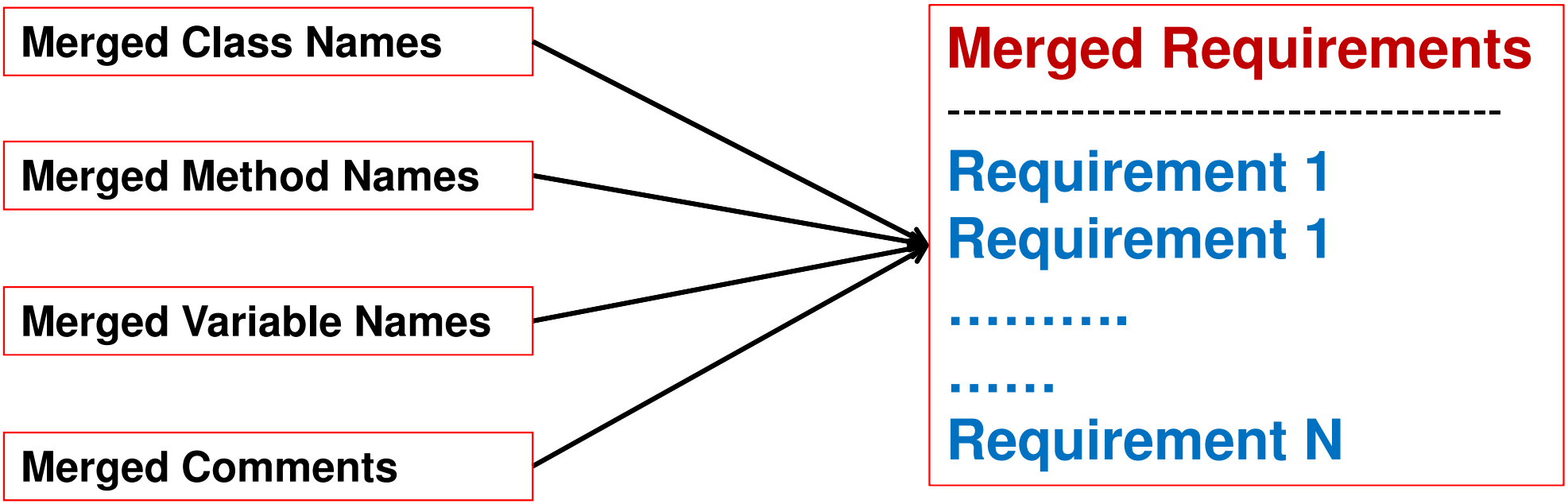
Merged Variable Names

Merged Comments

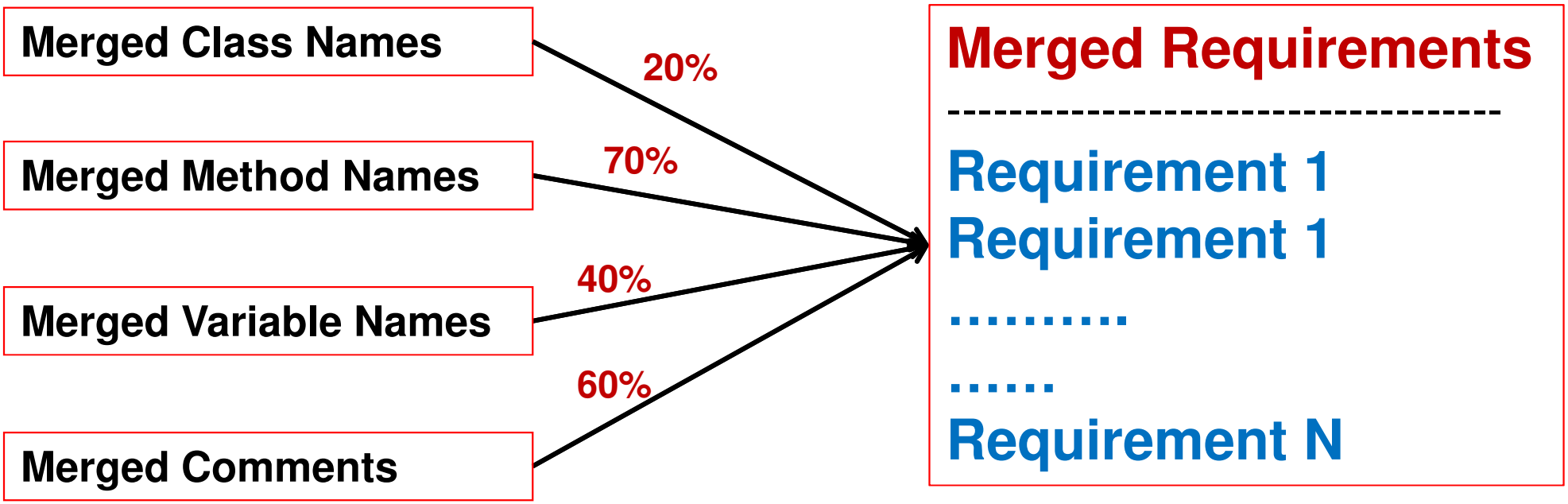
Merged Requirements

Requirement 1
Requirement 1
.....
.....
Requirement N

Creation of Partrace Expert

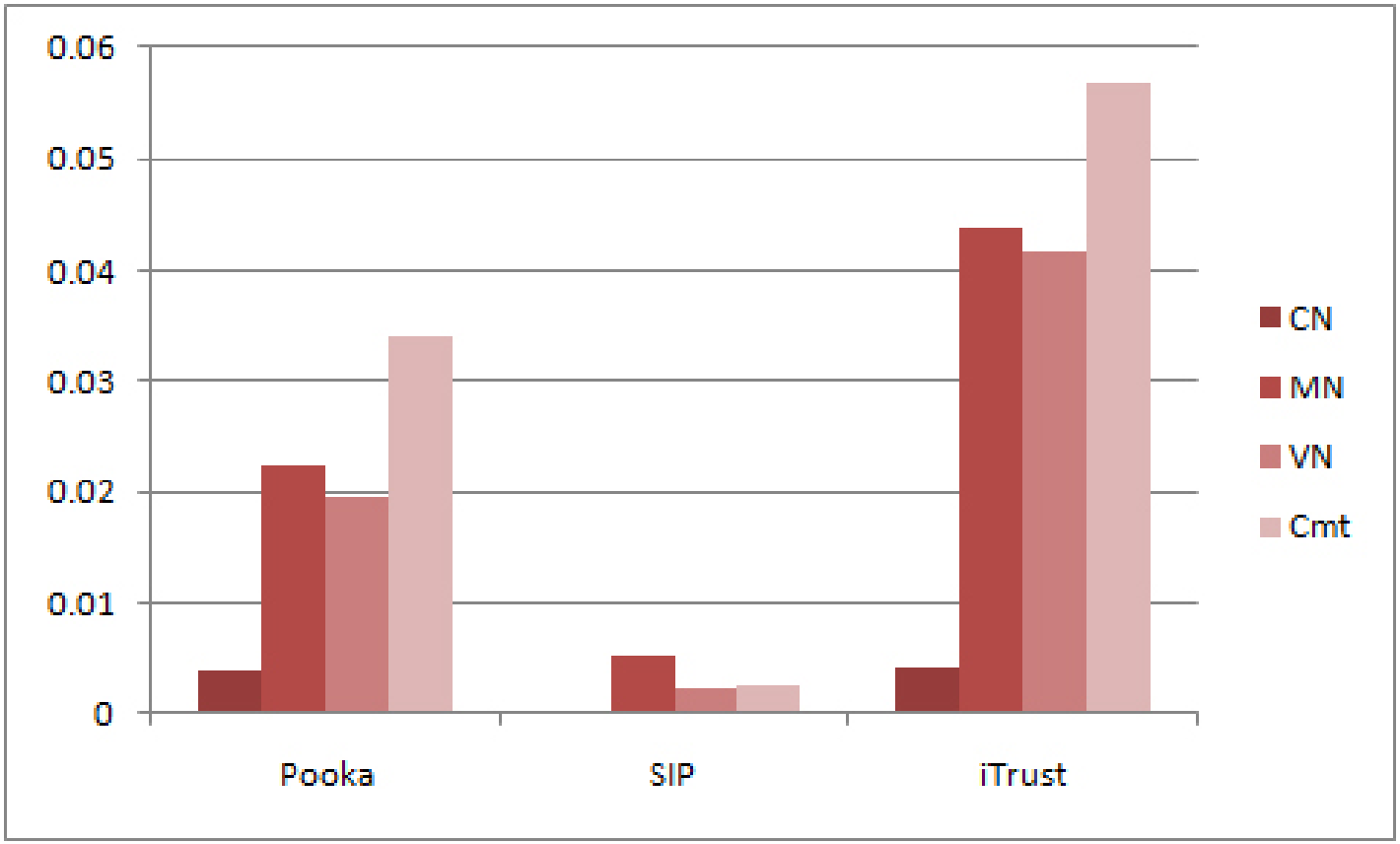


Creation of Partrace Expert

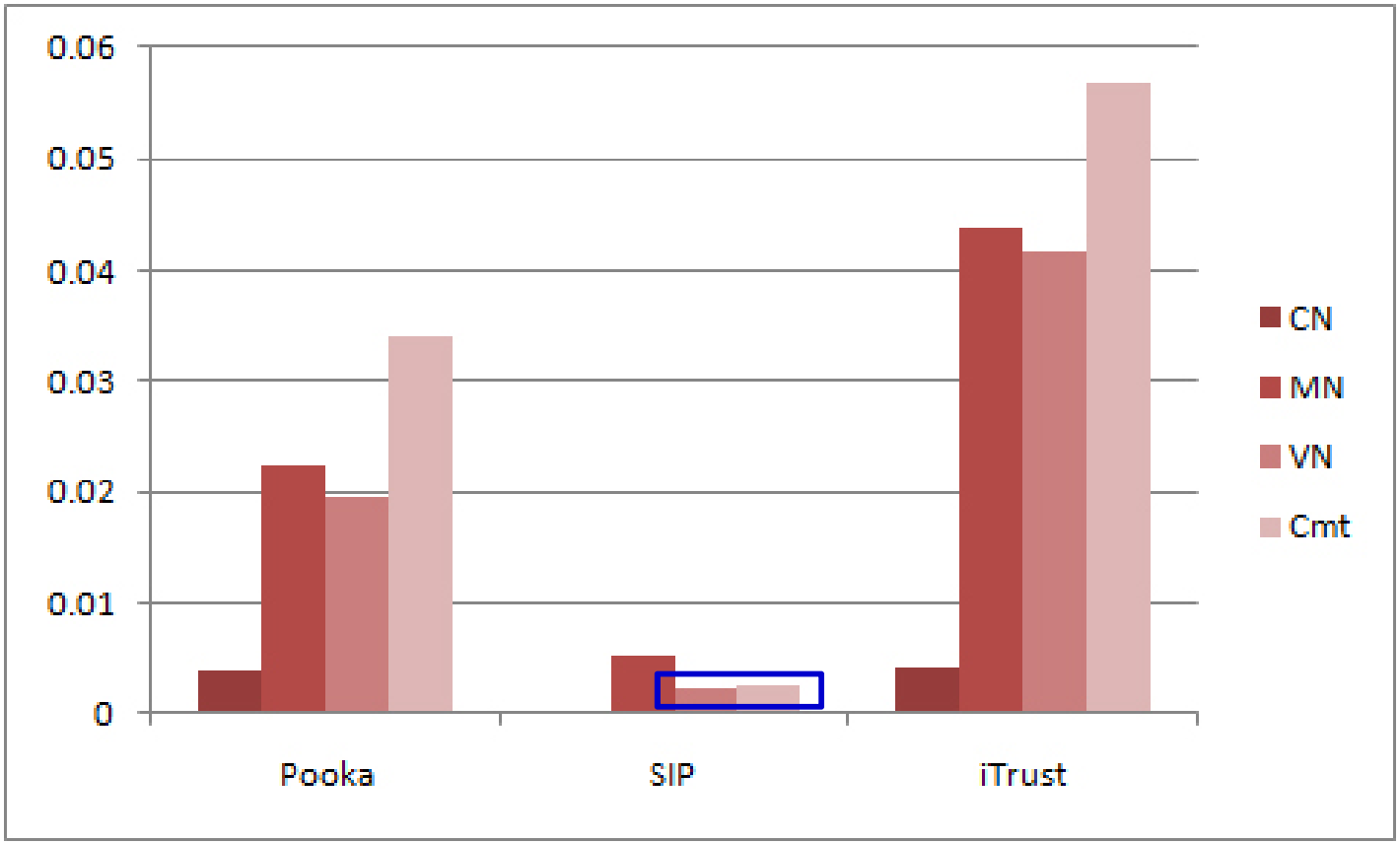


Usage of Partrace Expert

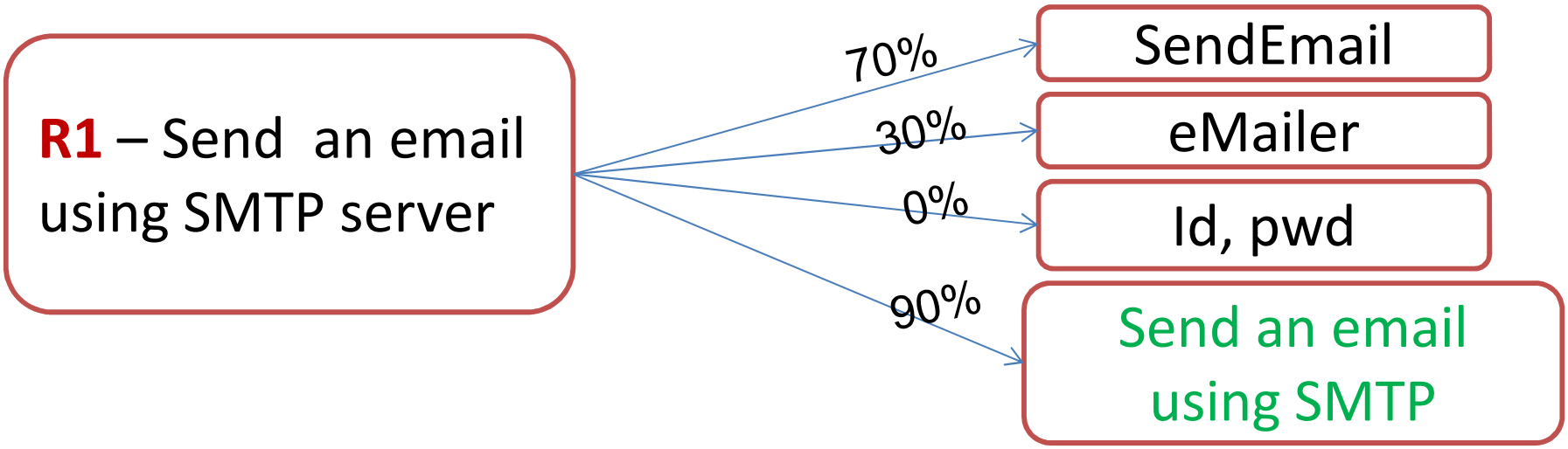
Usage of Partrace Expert



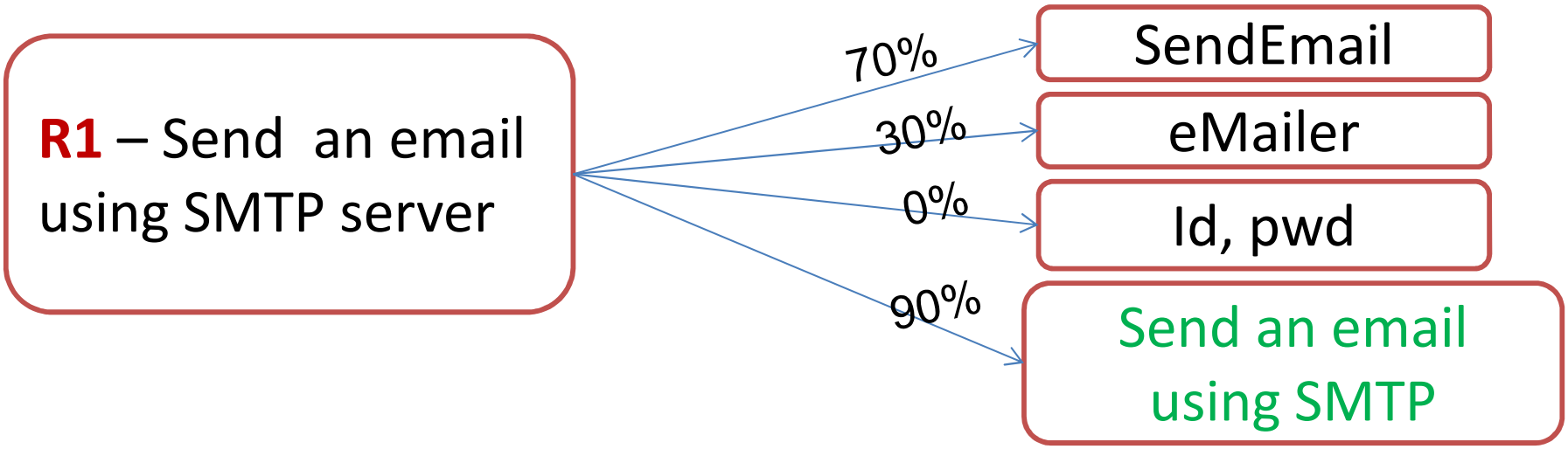
Usage of Partrace Expert

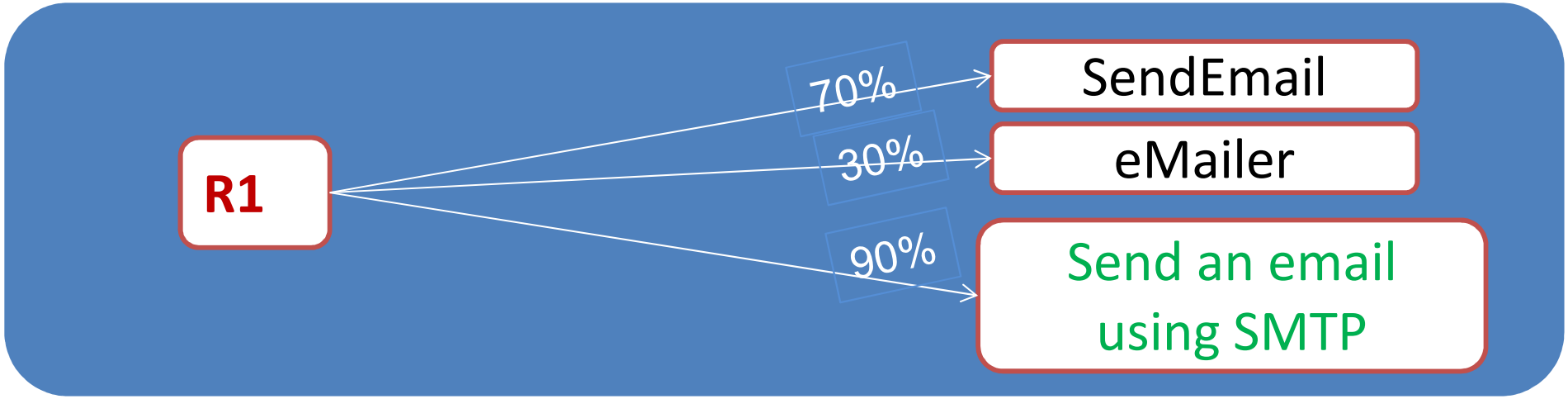


Creation of Partrace Expert

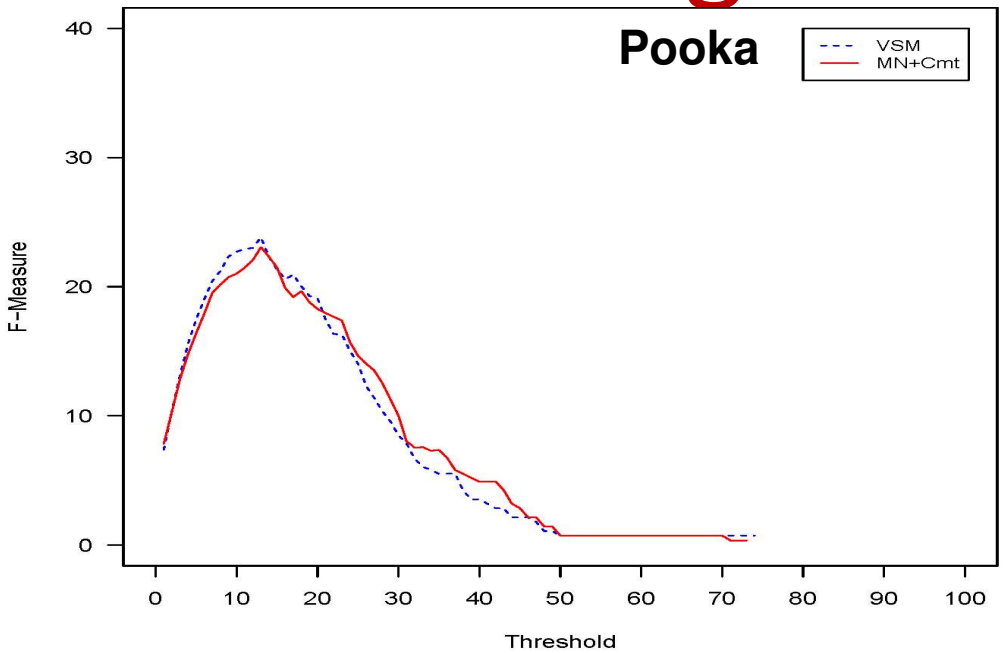


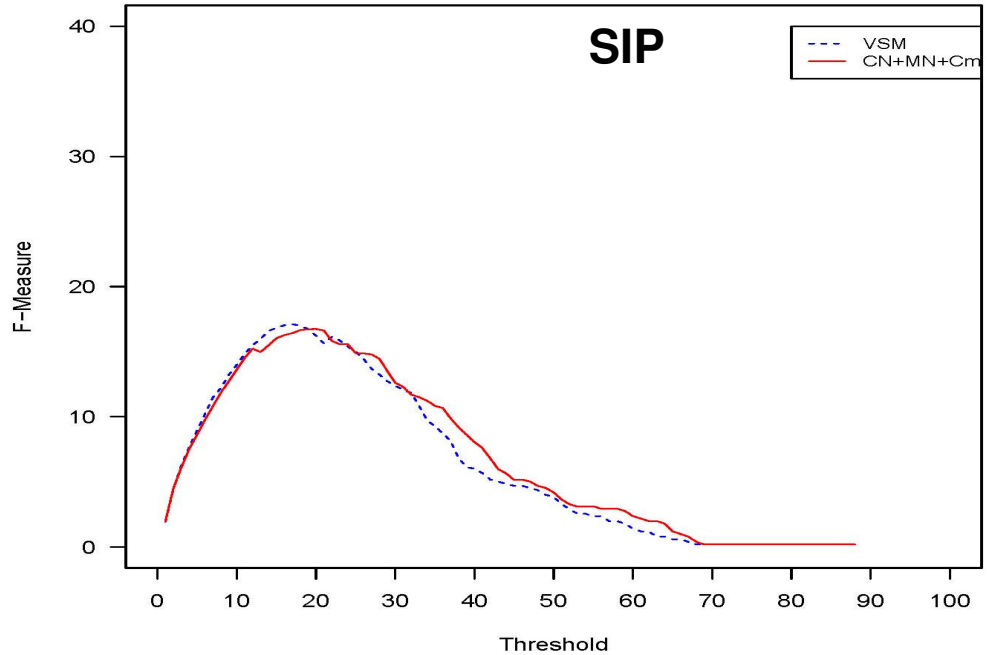
Creation of Partrace Expert

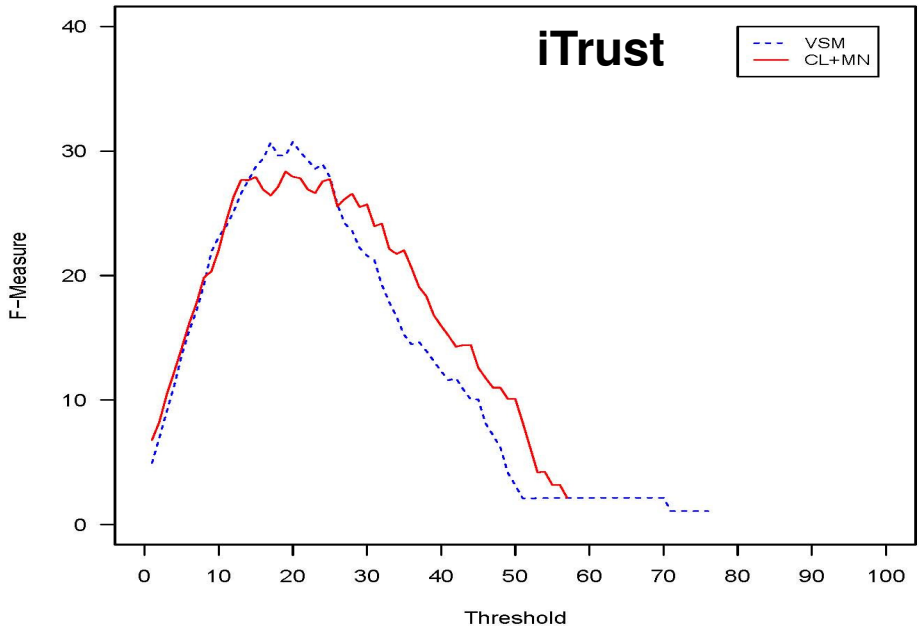




Voting vs. Combination







Alternative Weighting Scheme 1: SE/IDF

$$scetf_{i,j} = \begin{cases} tf_{i,j} \times \boxed{\alpha} \; if \; t_i & \text{class name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\alpha} \times \boxed{\lambda_1}) \; if \; t_i & \text{class name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \boxed{\beta} \; if \; t_i & \text{method name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\beta} \times \boxed{\lambda_2}) \; if \; t_i & \text{method name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \boxed{\gamma} \; if \; t_i & \text{variable name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\gamma} \times \boxed{\lambda_3}) \; if \; t_i & \text{variable name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \boxed{\delta} \; if \; t_i & \text{comment } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\delta} \times \boxed{\lambda_4}) \; if \; t_i & \text{comment } \textbf{in} \text{ requirements} \end{cases}$$

Alternative Weighting Scheme 1: SE/IDF

$$scetf_{i,j} = \begin{cases} tf_{i,j} \times \boxed{\alpha} \; if \; t_i & \text{class name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\alpha} \times \boxed{\lambda_1}) \; if \; t_i & \text{class name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \boxed{\beta} \; if \; t_i & \text{method name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\beta} \times \boxed{\lambda_2}) \; if \; t_i & \text{method name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \boxed{\gamma} \; if \; t_i & \text{variable name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\gamma} \times \boxed{\lambda_3}) \; if \; t_i & \text{variable name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \boxed{\delta} \; if \; t_i & \text{comment } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\boxed{\delta} \times \boxed{\lambda_4}) \; if \; t_i & \text{comment } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \Psi \; if & t_i \text{ requirement } \textit{not in} \text{ code} \end{cases}$$

Alternative Weighting Scheme 1: SE/IDF

$$scetf_{i,j} = \begin{cases} tf_{i,j} \times \alpha \ if \ t_i & \text{class name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\alpha \times \lambda_1) \ if \ t_i & \text{class name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \beta \ if \ t_i & \text{method name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\beta \times \lambda_2) \ if \ t_i & \text{method name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \gamma \ if \ t_i & \text{variable name } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\gamma \times \lambda_3) \ if \ t_i & \text{variable name } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \delta \ if \ t_i & \text{comment } \textit{not in} \text{ requirements} \\ tf_{i,j} \times (\delta \times \lambda_4) \ if \ t_i & \text{comment } \textbf{in} \text{ requirements} \\ tf_{i,j} \times \Psi \ if & t_i \text{ requirement } \textit{not in} \text{ code} \end{cases}$$

$$SE/IDF_{i,j} = scetf_{i,j} \times IDF_i$$

Alternative Weighting Scheme 2: DOI/IDF

$$DOITF_{i,j} = \begin{cases} tf_{i,j} \times \boxed{\Upsilon}\, if\ t_i & \text{domain term} \\ tf_{i,j} \times \boxed{\Phi}\, if\ t_i & \text{implementation term} \end{cases}$$

Alternative Weighting Scheme 2: DOI/IDF

$$DOITF_{i,j} = \begin{cases} tf_{i,j} \times \boxed{\Upsilon}\, if\ t_i & \text{domain term} \\ tf_{i,j} \times \boxed{\Phi}\, if\ t_i & \text{implementation term} \end{cases}$$

$$DOI/IDF_{i,j} = (SE/IDF_{i,j} + DOITF_{i,j}) \times IDF_i$$