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

SOURCE CODE
Context

Requirements

RSS option for an instant messenger

SOURCE CODE

IR Techniques
Context

Requirements

RSS option for an instant messenger

SOURCE CODE

20% of similarity

IR Techniques
Outline

• Introduction
• Related Work
• Creation of Experts
• Combining and Usage of Experts' Opinions
• Assigning Weights to Experts
• Empirical Evaluation
• Conclusion and Future Work
<table>
<thead>
<tr>
<th></th>
<th>Single Expert</th>
<th>Multiple Experts</th>
<th>Combining Experts</th>
<th>Automated Weighting</th>
<th>Feature Location</th>
<th>Req. Traceability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gethers et al. (2011)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>De Lucia et al. (2011)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Asuncion et al. (2010)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Maletic and Collard (2009)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abadi et al. (2008)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Mader et al. (2008)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Poshyvanyk et al. (2007)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marcus and Maletic (2003)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antoniol et al. (2002b)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
public class SendEmail {

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

    public sendOutEmail(String emailId) {

        EmailAddressFormatChecker emailAddressFormatChecker = new EmailAddressFormatChecker();
        int status;

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

```
public class SendEmail {
    SendEmail (){}
    ................
    ................
}

public sendOutEmail(String 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
Problem

Gethers et al. (2011)
VSM + JSM + RTM

Requirement
check email address format before storing it in address book
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

IR Technique

R1

30%

SVN/CVS Log Message 1

10%

SVN/CVS Log Message 2
Creation of Histrace Expert

R1 30%  
IR Technique 10%  

SVN/CVS Log Message 1  
Class - A  
Class - B

SVN/CVS Log Message 2  
Class - A  
Class - C
Creation of BCRTrace Expert

R1 – Instant Messenger should support RSS protocol
Creation of BCRTraceExpert

BCR

0.70

0.65

0.59

0.56

0.35

0.34

0.27

0.19

0.12

0.09

0.06

R1 – Instant Messenger should support RSS protocol
Creation of BCRTrace Expert

R1 – Instant Messenger should support RSS protocol
//Send an email using SMTP

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

class SendEmail {
    private void eMailer() {
        String id;
        String pwd;
        ..................
    }
}
Creation of Partrace Expert

R1 – Send an email using SMTP server
Creation of Partrace Expert

**R1** – Send an email using SMTP server

- 70% \rightarrow SendEmail
- 30% \rightarrow eMailer
- 0% \rightarrow Id, pwd
- 90% \rightarrow Send an email using SMTP
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

IR Technique (VSM)

Class - A

30%

Class - B

Class - C

Expert (Histrace)
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 expert provides evidence for a link
Combining Experts’ Opinions

TRUMO – Trust Model

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

IR Technique (VSM)

Class - A

Expert

Class - A

Class - B

Class - A

\[ \lambda_1 0.4\% + \lambda_2 0.5 + \lambda_3 0.2 \]

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

<table>
<thead>
<tr>
<th>Link ID</th>
<th>Expert 1</th>
<th>Expert 2</th>
<th>% of time</th>
<th>Final Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.21</td>
<td>0.35</td>
<td>14/75 = 0.19</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>$\lambda_1$</td>
<td>$\lambda_2$</td>
<td>$\lambda_3$</td>
<td></td>
</tr>
</tbody>
</table>

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

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

$\lambda_1 = 0.2$, $\lambda_2 = 0.5$, $\lambda_3 = 0.3$
# Static Weighting

Manually assign weight to each expert \cite{Poshyvanyk2007}

<table>
<thead>
<tr>
<th>Link ID</th>
<th>Expert 1</th>
<th>Expert 2</th>
<th>% of time</th>
<th>Final Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.21</td>
<td>0.35</td>
<td>14/75 = 0.19</td>
<td>?</td>
</tr>
</tbody>
</table>

\[ \lambda_1 = 0.33, \lambda_2 = 0.33, \lambda_3 = 0.33 \]
Voting

Class A

Requirements
------------------------------------
Email client must support pop3 integration.........
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

Requirements

Email client must support pop3 integration........

Comments of Class A

Method Names of Class A
Email client must support pop3 integration
Voting

Class A

Requirements

Email client must support pop3 integration

Comments of Class A

Method Names of Class A
Voting

Class A

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

<table>
<thead>
<tr>
<th></th>
<th>Trumo</th>
<th>DynWing</th>
<th>Trumo (Ranker)</th>
<th>Static Weight</th>
<th>PCA-based Weights</th>
<th>Voting</th>
<th>JSM</th>
<th>LSI</th>
<th>VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histrace</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Partrace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BCRTrace</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

- ✓: Present
- : Absent
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?

• **RQ3** - How does the accuracy of the traceability links recovered using DynWing compare to that using static weight and PCA?
## Empirical Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Trumo</th>
<th>DynWing</th>
<th>Trumo (Ranker)</th>
<th>Static Weight</th>
<th>PCA-based Weights</th>
<th>Voting</th>
<th>JSM</th>
<th>LSI</th>
<th>VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histrace</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Partrace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>BCRTrace</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

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

- **VSM**
- **JSM**

X Axis Shows Recall and Y Axis Shows Precision Values
RQ1 – **Histrace** Provides Better Accuracy Than VSM and JSM

**X Axis Shows Recall and Y Axis Shows Precision Values**

- **VSM**
- **JSM**

**P<0.05**
RQ1 – Histrace Provides Better Accuracy Than VSM and JSM
RQ1 – Histrace Provides Better Accuracy Than VSM and JSM

X Axis Shows Recall and Y Axis Shows Precision Values

VSM

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

<table>
<thead>
<tr>
<th></th>
<th>Trumo</th>
<th>DynWing</th>
<th>Trumo (Ranker)</th>
<th>Static Weight</th>
<th>PCA-based Weights</th>
<th>Voting</th>
<th>JSM</th>
<th>LSI</th>
<th>VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histrace</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Partrace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>BCRTrace</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**COPARVO**
RQ1 – Partrace Improves the Accuracy at Almost All Threshold Points

- **Pooka**
  - VSM
  - COPARVO

- **SIP**
  - VSM
  - COPARVO

- **iTrust**
  - VSM
  - COPARVO
RQ1 – Partrace Improves the Accuracy at Almost All Threshold Points

P ≤ 0.05
Effort Analysis

- **VSM**
- **Coparvo**

<table>
<thead>
<tr>
<th>Category</th>
<th>VSM</th>
<th>Coparvo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pooka</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP Comm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iTrust</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Empirical Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Trumo</th>
<th>DynWing</th>
<th>Trumo (Ranker)</th>
<th>Static Weight</th>
<th>PCA-based Weights</th>
<th>Voting</th>
<th>JSM</th>
<th>LSI</th>
<th>VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histrace</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Partrace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>BCRTrace</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
RQ2: BCRTrace **Consistently** ranks the Buggy Classes **Lower**
RQ2: BCRTrace **Consistently** ranks the Buggy Classes **Lower**
Static Relationships Analysis

with VSM

- association
- use
- aggregation
- inheritance
- all
Static Relationships Analysis

with VSM

association
use
aggregation
inheritance
all

Same Trend for LSI
RQ3 – DynWing **Automatically Assigns Weights** to Different Experts
RQ3 – DynWing **Automatically Assigns Weights** to Different Experts
RQ3 – DynWing **Automatically Assigns Weights to Different Experts**

![VSM Precision Recall Chart](chart.png)

- VSM
- Precision
- Recall
- Chart showing comparison between different tools and their performance metrics.
RQ3 – DynWing **Automatically Assigns Weights to Different Experts**

[Graph showing a scatter plot with labels for VSM, Precision, and Recall.]

Same Trend for JSM
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(String emailId) {
        EmailAddressFormatChecker emailAddressFormatChecker
            = new EmailAddressFormatChecker();
        int status;

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

<table>
<thead>
<tr>
<th>Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Subjects</td>
<td>26</td>
</tr>
<tr>
<td>Requirements</td>
<td>6</td>
</tr>
<tr>
<td>Total Source Code Snippet</td>
<td>6</td>
</tr>
</tbody>
</table>
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

```java
// 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);
    }
}
```

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?

<table>
<thead>
<tr>
<th>Physical</th>
<th>Conceptual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Method Name</td>
<td>1. Domain Related Terms</td>
</tr>
<tr>
<td>2. Comments</td>
<td>2. Application Related Terms</td>
</tr>
<tr>
<td>3. Variable Name</td>
<td></td>
</tr>
<tr>
<td>4. Class Name</td>
<td></td>
</tr>
</tbody>
</table>
RQ2: Can we Improve IR Techniques by making them Aware of the Developers’ Interests?

```
public class SendEmail {
    SendEmail (){
        ................
        ................
    }

    public sendOutEmail(String 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(String 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

![Graphs showing F-Measure vs Thresholds for Pooka and iTrust](image)

- 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)

<table>
<thead>
<tr>
<th></th>
<th>Trumo</th>
<th>DynWing</th>
<th>Trumo (Ranker)</th>
<th>Static Weight</th>
<th>PCA-based Weights</th>
<th>Voting</th>
<th>JSM</th>
<th>LSI</th>
<th>VSM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Histrace</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Partrace</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>BCRTrace</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- 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


Publications

Conference articles


Conference articles


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
- Binary Class Relationships
- Partitioning Source Code
- Using Developers’ Knowledge
- Dynamic Weights Calculator
- Trust Model
\[ R2CT_{i,r_j,t_k} = \{(r_j, c_s, \sigma'_i(r_j, t_k)) \mid c_s \in \delta_{T_i}(t_k) \& t_k \in T_i\} \]  

(1)

\[ Tr = \{(r_j, c_s, \sigma'_i(r_j, t_k)) \mid \exists t_k \in T_i : (r_j, c_s) \in \alpha(R2CT_{i,r_j,t_k}) \& (r_j, c_s) \in \alpha(R2C)\} \]  

(2)
\[
\sigma^*_i(r_j, c_s) = \frac{\sigma(r_j, c_s) + \sum_{l \in TC_i(r_j, c_s)} \phi(l)}{1 + |TC_i(r_j, c_s)|}
\]  

(3)

\[
\psi_{r_j, c_s}(Tr) = \left[ \sum_{i=1}^{P} \lambda_i(r_j, c_s) \sigma^*_i(r_j, c_s) \right] \\
+ \lambda_{P+1}(r_j, c_s) \frac{|Tr(r_j, c_s)|}{\max_{n,m} |Tr(r_n, c_m)|}
\]  

(4)
## Eye-tracking Experiment Results

<table>
<thead>
<tr>
<th>Source Code Entities</th>
<th>Average Fixation Time (ms) of All Subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Name(s)</td>
<td>5701.10</td>
</tr>
<tr>
<td>Comments</td>
<td>4542.41</td>
</tr>
<tr>
<td>Variable Name(s)</td>
<td>3181.81</td>
</tr>
<tr>
<td>Class Name(s)</td>
<td>2317.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain (48% of all terms)</th>
<th>Average Fixation Time (ms) of All Subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation (52% of all terms)</td>
<td>1729.80</td>
</tr>
</tbody>
</table>
Creation of Partrace Expert

Source Code

Class Name A
Class Name B
Class Name C
Class Name D
Creation of Partrace Expert

Source Code

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

Merged Class Names

- Class Name A
- Class Name B
- Class Name C
- Class Name D
Creation of Partrace Expert

**Source Code**

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

**Merged Class Names**

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

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

Merged Class Names
Merged Method Names
Merged Variable Names
Merged Comments

Merged Requirements

Requirement 1

.....

Requirement N
Creation of Partrace Expert

Merged Class Names
Merged Method Names
Merged Variable Names
Merged Comments

Merged Requirements
---------------------
Requirement 1
Requirement 1
......
Requirement N
Usage of Partrace Expert
Usage of Partrace Expert
Usage of Partrace Expert

![Bar chart showing usage comparison between Pooka, SIP, and iTrust]

- **Pooka**: CN > MN > VN > Cmt
- **SIP**: CN > VN > MN > Cmt
- **iTrust**: CN > VN > MN > Cmt

*Note: CN, MN, VN, and Cmt represent different categories of usage.*
Creation of Partrace Expert

R1 – Send an email using SMTP server

SendEmail

30%
esMailer

Id, pwd

70%
Send an email using SMTP

90%
Creation of Partrace Expert

**R1** – Send an email using SMTP server

- **70%**
  - SendEmail
- **30%**
  - eMailer
- **0%**
  - Id, pwd
  - Send an email using SMTP

- **90%**
  - SendEmail
  - eMailer
  - Id, pwd
  - Send an email using SMTP
Voting vs. Combination

Pooka

SIP

iTrust
Alternative Weighting Scheme 1: SE/IDF

\[
scetf_{i,j} = \begin{cases} 
 tf_{i,j} \times \alpha & \text{if } t_i \text{ class name } \text{not in} \text{ requirements} \\
 tf_{i,j} \times (\alpha \times \lambda_1) & \text{if } t_i \text{ class name } \text{in} \text{ requirements} \\
 tf_{i,j} \times \beta & \text{if } t_i \text{ method name } \text{not in} \text{ requirements} \\
 tf_{i,j} \times (\beta \times \lambda_2) & \text{if } t_i \text{ method name } \text{in} \text{ requirements} \\
 tf_{i,j} \times \gamma & \text{if } t_i \text{ variable name } \text{not in} \text{ requirements} \\
 tf_{i,j} \times (\gamma \times \lambda_3) & \text{if } t_i \text{ variable name } \text{in} \text{ requirements} \\
 tf_{i,j} \times \delta & \text{if } t_i \text{ comment } \text{not in} \text{ requirements} \\
 tf_{i,j} \times (\delta \times \lambda_4) & \text{if } t_i \text{ comment } \text{in} \text{ requirements}
\end{cases}
\]
Alternative Weighting Scheme 1: SE/IDF

\[ scet_{i,j} = \begin{cases} 
  t_{f_i,j} \times \alpha & \text{if } t_i \text{ class name not in requirements} \\
  t_{f_i,j} \times (\alpha \times \lambda_1) & \text{if } t_i \text{ class name in requirements} \\
  t_{f_i,j} \times \beta & \text{if } t_i \text{ method name not in requirements} \\
  t_{f_i,j} \times (\beta \times \lambda_2) & \text{if } t_i \text{ method name in requirements} \\
  t_{f_i,j} \times \gamma & \text{if } t_i \text{ variable name not in requirements} \\
  t_{f_i,j} \times (\gamma \times \lambda_3) & \text{if } t_i \text{ variable name in requirements} \\
  t_{f_i,j} \times \delta & \text{if } t_i \text{ comment not in requirements} \\
  t_{f_i,j} \times (\delta \times \lambda_4) & \text{if } t_i \text{ comment in requirements} \\
  t_{f_i,j} \times \Psi & \text{if } t_i \text{ requirement not in code} 
\end{cases} \]
Alternative Weighting Scheme 1: SE/IDF

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

\[
SE/IDF_{i,j} = scetf_{i,j} \times IDF_i
\]
Alternative Weighting Scheme 2: DOI/IDF

\[ DOIIF_{i,j} = \begin{cases} 
  t f_{i,j} \times \Upsilon & \text{if } t_i \text{ domain term} \\
  t f_{i,j} \times \Phi & \text{if } t_i \text{ implementation term}
\end{cases} \]
Alternative Weighting Scheme 2: DOI/IDF

\[ DOI/IDF_{i,j} = \begin{cases} 
  tf_{i,j} \times \Upsilon & \text{if } t_i \text{ domain term} \\
  tf_{i,j} \times \Phi & \text{if } t_i \text{ implementation term}
\end{cases} \]

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