Exploring the Neighborhood with Dora to Expedite Software Maintenance

Emily Hill, Lori Pollock, K. Vijay-Shanker
University of Delaware
Program Exploration for Maintenance
Program Exploration for Maintenance

You are here
Program Exploration for Maintenance

You are here

Exploring the Neighborhood with Dora

ASE 2007

Emily Hill • University of Delaware
Running Example Scenario

eBay auction sniping (bidding) program has bug in add auction event trigger
Running Example Scenario

eBay auction sniping (bidding) program has bug in add auction event trigger

- Exploration Task: Locate code related to ‘add auction’ trigger
- Starting point: DoAction() method, from prior knowledge
Exploring with only structural information

Looking for: ‘add auction’ trigger
Exploring with only structural information

Looking for: ‘add auction’ trigger

- **DoAction()** has 40 callees
Exploring with only structural information

Looking for: ‘add auction’ trigger

- `DoAction()` has 40 callees

Only 2/40 methods relevant
Exploring with only structural information

Looking for: ‘add auction’ trigger

- `DoAction()` has 40 callees

- 

  `DoAdd()`

  `DoPasteFromClipboard()`

  Only 2/40 methods relevant

- Locates locally relevant items
Exploring with only structural information

Looking for: ‘add auction’ trigger

- **DoAction()** has 40 callees

- **DoAdd()**
- **DoPasteFromClipboard()**

Only 2/40 methods relevant

+ Locates locally relevant items
- But too many irrelevant
Exploring with only structural information

Looking for: ‘add auction’ trigger
- `DoAction()` has 40 callees

- `DoAdd()`
- `DoPasteFromClipboard()`

Only 2/40 methods relevant

+ Locates locally relevant items
- But too many irrelevant

And what if you wanted to explore more than one edge away?
Alternative: Exploring with only lexical information
Alternative: Exploring with only lexical information

Looking for: ‘add auction’ trigger in 1902 methods (159 files, 23KLOC)
Alternative: Exploring with only lexical information

Looking for: ‘add auction’ trigger in 1902 methods (159 files, 23KLOC)

- Use lexical information from comments & identifiers
Alternative: Exploring with only lexical information

Looking for: ‘add auction’ trigger in 1902 methods (159 files, 23KLOC)

- Use lexical information from comments & identifiers
- Search with query ‘add*auction’
Alternative: Exploring with only lexical information

Looking for: ‘add auction’ trigger in 1902 methods (159 files, 23KLOC)

- Use lexical information from comments & identifiers
- Search with query ‘add*auction’

- 91 query matches in 50 methods
**Alternative: Exploring with only lexical information**

Looking for: ‘add auction’ trigger in 1902 methods (159 files, 23KLOC)

- Use lexical information from comments & identifiers
- Search with query ‘add*auction’

- 91 query matches in 50 methods
- Only 2/50 methods are relevant
Alternative: Exploring with only lexical information

Looking for: ‘add auction’ trigger in 1902 methods (159 files, 23KLOC)

- Use lexical information from comments & identifiers
- Search with query ‘*add*auction’
- 91 query matches in 50 methods
- Only 2/50 methods are relevant

+ Locates globally relevant items
- But too many irrelevant
Dora gets it right…

Looking for: ‘add auction’ trigger

DoAction()
Dora gets it right...

Looking for: ‘add auction’ trigger

✓ Structural: guide exploration from starting point
Dora gets it right...

Looking for: ‘add auction’ trigger

✓ **Structural**: guide exploration from starting point

✓ **Lexical**: prunes irrelevant edges
Software Maintenance:
Dora to the rescue
Software Maintenance: Dora to the rescue

- Developers spend more time finding and understanding code than actually fixing bugs [Kersten & Murphy 2005, Ko et al. 2005]
Software Maintenance: Dora to the rescue

- Developers spend more time finding and understanding code than actually fixing bugs [Kersten & Murphy 2005, Ko et al. 2005]

- Critical need for automated tools to help developers explore and understand today’s large & complex software
Software Maintenance: Dora to the rescue

- Developers spend more time finding and understanding code than actually fixing bugs [Kersten & Murphy 2005, Ko et al. 2005]

- Critical need for automated tools to help developers explore and understand today’s large & complex software

→ **Key Contribution:** Automated tools can use program structure *and* identifier names to save the developer time and effort
Dora the Program Explorer

Dora

*Dora comes from exploradora, the Spanish word for a female explorer.*
Dora the Program Explorer

Natural Language Query
- Maintenance request
- Expert knowledge
- Query expansion

*Dora comes from *exploradora*, the Spanish word for a female explorer.
Dora the Program Explorer

Natural Language Query
- Maintenance request
- Expert knowledge
- Query expansion

Program Structure
- Representation
- Current: call graph
- Seed starting point

* Dora comes from exploradora, the Spanish word for a female explorer.
Dora the Program Explorer*

Natural Language Query
- Maintenance request
- Expert knowledge
- Query expansion

Program Structure
- Representation
- Current: call graph
- Seed starting point

Dora

* Dora comes from exploradora, the Spanish word for a female explorer.
Dora the Program Explorer

Natural Language Query
- Maintenance request
- Expert knowledge
- Query expansion

Program Structure
- Representation
- Current: call graph
- Seed starting point

Dora

Relevant Neighborhood
- Subgraph relevant to query

* Dora comes from *exploradora*, the Spanish word for a female explorer.
The Dora Approach

Prune irrelevant structural edges from seed

1. Obtain set of methods one call edge away from seed
2. Determine each method’s relevance to query
   - Calculate lexical-based relevance score
3. Prune low-scored methods from neighborhood, using threshold
4. Recursively explore
The Dora Approach

Prune irrelevant structural edges from seed

1. Obtain set of methods one call edge away from seed
2. Determine each method’s relevance to query
   - Calculate lexical-based relevance score
3. Prune low-scored methods from neighborhood, using threshold
4. Recursively explore
Calculating Relevance Score: Term Frequency

- Score based on number of occurrences of query terms in the method
- **Intuition:** The more query terms in a method, the more likely it is relevant
Calculating Relevance Score: Term Frequency

- Score based on number of occurrences of query terms in the method
- **Intuition:** The more query terms in a method, the more likely it is relevant

Query: ‘add auction’
Calculating Relevance Score: Term Frequency

- Score based on number of occurrences of query terms in the method
- **Intuition**: The more query terms in a method, the more likely it is relevant

```java
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add";
    endResult = promptString(src, prompt, "Adding");
    // They closed the window or cancelled.
    if (endResult == null) return;
    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_AUCTION + endResult");
}
```
Calculating Relevance Score: Term Frequency

- Score based on number of occurrences of query terms in the method
- Intuition: The more query terms in a method, the more likely it is relevant

```java
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter theauction number to add:");
    endResult = promptString(src, prompt, Adding);
    // They closed the window or cancelled.
    if (endResult == null) return;
    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue(ADD_AUCTION + endResult);
}
```
Calculating Relevance Score: Term Frequency

- Score based on number of occurrences of query terms in the method
- Intuition: The more query terms in a method, the more likely it is relevant

```java
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add:";
    endResult = promptString(src, prompt, "Adding");

    // They closed the window or cancelled.
    if (endResult == null) return;

    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue();
}

private void DeleteComment(AuctionEntry ae) {
    if (ae == null) {
        ErrorManagement.logMessage("Auction selected to delete");
        return;
    }

    ae.setComment("");
    FilterManager.getInstance().redrawEntry(ae);
}
```
Calculating Relevance Score: Term Frequency

- Score based on number of occurrences of query terms in the method
- **Intuition:** The more query terms in a method, the more likely it is relevant

**Query:** ‘add auction’

```
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter the number to add: 
    endResult = promptString(src, prompt, Adding);

    // They closed the window or cancelled.
    if (endResult == null) return;

    endResult = endResult.trim();
    MQFactory.getConcrete("user").enque
}
```

6 query term occurrences

```
private void DeleteComment(AuctionEntry ae) {
    if(ae == null) {
        ErrorManagement.logMessage('Auction selected to delete return;
    }
}
```

Only 2 occurrences
Calculating Relevance Score: Inverse Document Frequency

- What about terms that appear all over the program?
Calculating Relevance Score: Inverse Document Frequency

- What about terms that appear all over the program?
- Use inverse document frequency \((\text{idf})\)
  - Intuition: Highly weight terms that appear in few documents/methods
    - Terms appearing all over program not good discriminators
    - Don’t separate relevant from irrelevant methods
  - \(\text{Number of methods} \div \text{number of methods containing the term}\)
Calculating Relevance Score: Inverse Document Frequency

- What about terms that appear all over the program?
- Use inverse document frequency (idf)
  - Intuition: Highly weight terms that appear in few documents/methods
    - Terms appearing all over program not good discriminators
    - Don’t separate relevant from irrelevant methods
  - Number of methods / number of methods containing the term

```java
private void Add(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add";
    endResult = promptString(src, prompt, 'Adding');
    // They closed the window or cancelled.
    if (endResult == null) return;
    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_AUCTION" + endResult);
}
```
Calculating Relevance Score: Inverse Document Frequency

- What about terms that appear all over the program?
- Use inverse document frequency (idf)
  - Intuition: Highly weight terms that appear in few documents/methods
    - Terms appearing all over program not good discriminators
    - Don’t separate relevant from irrelevant methods
  - Number of methods / number of methods containing the term

```java
private void D_add(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add:
    endResult = promptString(src, prompt,
    // They closed the window or cancelled.
    if (endResult == null) return;
    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_ADD_"
```
Calculating Relevance Score: TF-IDF

- Score based on method query term frequency \((tf)\)
- Multiplied by natural log of inverse document frequency \((idf)\)
Calculating Relevance Score: TF-IDF

- Score based on method query term frequency (tf)
- Multiplied by natural log of inverse document frequency (idf)

```java
private void Add(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add:"
    endResult = promptString(src, prompt, 'Adding');
    // They closed the window or cancelled.
    if (endResult == null) return;

    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_AUCTION" + endResult);
}
```
Calculating Relevance Score: TF-IDF

- Score based on method query term frequency \( (tf) \)
- Multiplied by natural log of inverse document frequency \( (idf) \)

```java
private void Add(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add: "
    endResult = promptString(src, prompt, "Adding!");

    // They closed the window or cancelled.
    if (endResult == null) return;

    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_AUCTION + endResult");
}
```

Query: ‘add auction’

6 query term occurrences

\[ \text{tf-idf} = 4 \cdot \ln(7.37) + 2 \cdot \ln(4.58) = 11.03 \]
Calculating Relevance Score: TF-IDF

- Score based on method query term frequency \((tf)\)
- Multiplied by natural log of inverse document frequency \((idf)\)

```java
private void Add(Component src) {
    String endResult;
    String prompt = "Enter the auction number to 
        Add":
    endResult = promptString(src, prompt, [\textcolor{green}{\textit{Adding}}]);
    // They closed the window or cancelled.
    if (endResult == null) return;
    endResult = endResult.trim();
    MQFactory.getConcrete("user").enq
}
```

6 query term occurrences

- Add: 4 \(\cdot \ln(7.37)\)
- Auction: 2 \(\cdot \ln(4.58)\)

\[ \text{tf-idf} = 4 \cdot \ln(7.37) + 2 \cdot \ln(4.58) = 11.03 \]
Calculating Relevance Score: TF-IDF

- Score based on method query term frequency \((tf)\)
- Multiplied by natural log of inverse document frequency \((idf)\)

Query: ‘add auction’

```
private void Add(Component src) {
    String endResult;
    String prompt = "Enter the \textcolor{green}{auction} number to add:";

    endResult = promptString(src, prompt, \textcolor{red}{Adding});

    // They closed the window or cancelled.
    if (endResult == null) return;

    endResult = endResult.trim();
    
    private void DeleteComment(AuctionEntry ae) {
        MQFactory.getConcrete("user").enqueue(ae);
        if (ae == null) {
            ErrorManagement.logMessage('Auction \textcolor{green}{selected} to delete return;

        } else {
            ae.setComment("\textcolor{red}{Auction}");
            FilterManager.getInstance().redrawEntry(ae);
```
Calculating Relevance Score: What about location?

- Weigh term frequency \((tf-idf)\) based on location:
Calculating Relevance Score: What about location?

- Weigh term frequency \((tf-idf)\) based on location:

```java
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add";
    endResult = promptString(src, prompt, "Adding");
    // They closed the window or cancelled.
    if (endResult == null) return;
    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_AUCTION + endResult");
}
```
Calculating Relevance Score: What about location?

- Weigh term frequency (tf-idf) based on location:
  - Method name more important than body

```
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add";
    endResult = promptString(src, prompt, "Adding");
    // They closed the window or cancelled.
    if (endResult == null) return;
    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_AUCTION" + endResult);
}
```
Calculating Relevance Score:
What about location?

- Weigh term frequency \((tf-idf)\) based on location:
  - Method name more important than body

```java
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add;"
    endResult = promptString(src, prompt, "Adding");

    // They closed the window
    if (endResult == null) return

    endResult = endResult.trim();
    MQFactory.getConcrete("user").enqueue("ADD_AUCTION + auctionId");
}
```

```java
private void DoPasteFromClipboard() {
    String auctionId = getClipboardString();

    // ...
    if(auctionId != null) {
        MQFactory.getConcrete("user").enqueue("ADD_AUCTION + auctionId");
    }
}
```
Calculating Relevance Score: What about location?

- Weigh term frequency \((tf-idf)\) based on location:
  - Method name more important than body
  - Method body statements normalized by length

```java
private void DoAdd(Component src) {
    String endResult;
    String prompt = "Enter the auction number to add:"
    endResult = promptString(src, prompt, "Adding");
    // They closed the window
    if (endResult == null) return
    endResult = endResult.trim();
    MQFactory.getConcrete("use");
}
```

```java
private void DoPasteFromClipboard() {
    String auctionId = getClipboardString();
    // ...
    if(auctionId != null) {
        MQFactory.getConcrete("user").enqueue("ADD_AUCTION + auctionId");
    }
}
```
Dora’s Relevance Score
Dora’s Relevance Score

- Factors
  - $\sum \text{tf-idf}$ for each query term in the method name
  - $\sum \text{tf-idf}$ for each query term in the method body
    - the number of statements in the method
Dora’s Relevance Score

● Factors
  ● $\sum \text{tf-idf}$ for each query term in the method name
  ● $\sum \text{tf-idf}$ for each query term in the method body
    the number of statements in the method

● How to determine weights?
  ● Applied logistic regression
  ● Trained on methods from 9 concerns in previous concern location tool evaluation [Shepherd et al. 2007]
    (A concern is a conceptual unit of the software, such as a feature, requirement, design idiom, or implementation mechanism [Robillard & Murphy 2007].)
Dora’s Relevance Score

- **Factors**
  - $\sum \text{tf-idf}$ for each query term in the method name
  - $\sum \text{tf-idf}$ for each query term in the method body
  - the number of statements in the method

- **How to determine weights?**
  - Applied logistic regression
  - Trained on methods from 9 concerns in previous concern location tool evaluation [Shepherd et al. 2007]
    - A *concern* is a conceptual unit of the software, such as a feature, requirement, design idiom, or implementation mechanism [Robillard & Murphy 2007].

- **For details, see paper**
Example:
Dora explores ‘add auction’ trigger
Example:
Dora explores ‘add auction’ trigger

Scores from DoAction() seed:
Example:
Dora explores ‘add auction’ trigger

Scores from **DoAction()** seed:
- Identified as relevant with 0.5 threshold
  - **DoAdd()** (0.93)
  - **DoPasteFromClipboard()** (0.60)
Example:
Dora explores ‘add auction’ trigger

Scores from \texttt{DoAction()} seed:
- Identified as relevant with 0.5 threshold
  - \texttt{DoAdd()} (0.93)
  - \texttt{DoPasteFromClipboard()} (0.60)
- With only one false positive
  - \texttt{DoSave()} (0.52)
Experimental Evaluation: Research Questions
Experimental Evaluation: Research Questions

- Does an integrated lexical- and structural-based approach outperform a purely structural approach?
Experimental Evaluation: Research Questions

- Does an integrated lexical- and structural-based approach outperform a purely structural approach?

- Is a sophisticated lexical scoring technique required, or are naïve lexical scoring techniques sufficient to identify the relevant neighborhood?
Experimental Evaluation: Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
Experimental Evaluation: Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare**: 4 exploration techniques: 1 structural, 3 lexical + structural
Experimental Evaluation: Design

- **Gold Set:** 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare** 4 exploration techniques: 1 structural, 3 lexical + structural
  - **Structural:** Suade [Robillard 2005]
Experimental Evaluation: Design

- **Gold Set:** 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare** 4 exploration techniques: 1 structural, 3 lexical + structural
  - **Structural:** Suade [Robillard 2005]
    - Automatically generates exploration suggestions from seed set
    - Elements that have few connections outside the seed set are more relevant
    - Uses caller/callee & field def-use information to make recommendations
Experimental Evaluation:
Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare** 4 exploration techniques: 1 structural, 3 lexical + structural
  - **Structural**: Suade [Robillard 2005]
    - Automatically generates exploration suggestions from seed set
    - Elements that have few connections outside the seed set are more relevant
    - Uses caller/callee & field def-use information to make recommendations
  - **Lexical + Structural**: Dora (sophisticated)
  - **Lexical + Structural**: boolean AND (naive)
  - **Lexical + Structural**: boolean OR (naive)
Experimental Evaluation: Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare**: 4 exploration techniques: 1 structural, 3 lexical + structural
Experimental Evaluation: Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare**: 4 exploration techniques: 1 structural, 3 lexical + structural
- **Measures**: Precision (P), Recall (R), & F Measure (F)
Experimental Evaluation: Design

- **Gold Set:** 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare** 4 exploration techniques: 1 structural, 3 lexical + structural
- **Measures:** Precision (P), Recall (R), & F Measure (F)
  
  - \( P = \frac{TP}{TP+FP} \) (Are the results returned actually relevant?)
  
  - \( R = \frac{TP}{TP+FN} \) (How close are the returned results to the gold set?)
  
  - \( F = \frac{2PR}{P+R} \) (High when P & R are similarly high)
Experimental Evaluation: Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare**: 4 exploration techniques: 1 structural, 3 lexical + structural
- **Measures**: Precision (P), Recall (R), & F Measure (F)
Experimental Evaluation: Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare**: 4 exploration techniques: 1 structural, 3 lexical + structural
- **Measures**: Precision (P), Recall (R), & F Measure (F)
- **Methodology**
  - For each exploration technique $t$
    - For each method $m$ in the gold set
      - Score each caller & callee of $m$ with $t$
      - Calculate P, R, & F for $m$ with $t$
Experimental Evaluation: Design

- **Gold Set**: 8 concerns from 4 Java programs, manually mapped by 3 independent developers [Robillard et al. 2007]
- **Compare** 4 exploration techniques: 1 structural, 3 lexical + structural
- **Measures**: Precision (P), Recall (R), & F Measure (F)
- **Methodology**
  - For each exploration technique $t$
  - For each method $m$ in the gold set
    - Score each caller & callee of $m$ with $t$
    - Calculate P, R, & F for $m$ with $t$
- 160 seed methods, 1885 call edges (with overlap)
Results: All Concerns
Results: All Concerns

- Dora outperforms Suade with statistical significance ($\alpha = 0.05$)
Results: All Concerns

- Dora outperforms Suade with statistical significance ($\alpha = 0.05$)
- Dora, OR, and Suade perform significantly better than AND
Results: All Concerns

- Dora outperforms Suade with statistical significance ($\alpha = 0.05$)
- Dora, OR, and Suade perform significantly better than AND
- Dora and Suade not significantly different from OR ($\alpha = 0.05$)
  - OR > Suade, $p = 0.43$
  - Dora > OR, $p = 0.033$
  - Dora > Suade, $p = 0.0037$
Results: All Concerns

- Dora outperforms Suade with statistical significance ($\alpha = 0.05$)
- Dora, OR, and Suade perform significantly better than AND
- Dora and Suade not significantly different from OR ($\alpha = 0.05$)
  - OR > Suade, $p = 0.43$
  - Dora > OR, $p = 0.033$
  - Dora > Suade, $p = 0.0037$
- Dora achieves 100% P & R for 25% of the data—more than any other technique
Results: By Concern

- Overall trend also seen for most concerns
Results: By Concern

- Overall trend also seen for most concerns
- **Exceptions: 9 & 12**
  - AND had much higher precision
  - Relevant methods contained both query terms
Experimental Evaluation: Result Summary

- Does an integrated lexical- and structural-based approach (*Dora*) outperform a purely structural approach (*Suade*)?

- Is a sophisticated lexical scoring technique required, or are naïve lexical scoring techniques sufficient to identify the relevant neighborhood?
Experimental Evaluation: Result Summary

- Does an integrated lexical- and structural-based approach (*Dora*) outperform a purely structural approach (*Suade*)?  
  \[ \Rightarrow \text{Dora outperforms Suade with statistical significance (} \alpha = 0.05) \]

- Is a sophisticated lexical scoring technique required, or are naïve lexical scoring techniques sufficient to identify the relevant neighborhood?
Experimental Evaluation:
Result Summary

- Does an integrated lexical- and structural-based approach (*Dora*) outperform a purely structural approach (*Suade*)?
  - Dora outperforms Suade with statistical significance ($\alpha = 0.05$)

- Is a sophisticated lexical scoring technique required, or are naïve lexical scoring techniques sufficient to identify the relevant neighborhood?
  - Although not statistically significant, Dora outperforms OR
  - Dora, Suade, & OR outperform AND ($\alpha = 0.05$)
Experimental Evaluation: Result Summary

- Does an integrated lexical- and structural-based approach (*Dora*) outperform a purely structural approach (*Suade*)?
  - Dora outperforms Suade with statistical significance (\(\alpha = 0.05\))

- Is a sophisticated lexical scoring technique required, or are naïve lexical scoring techniques sufficient to identify the relevant neighborhood?
  - Although not statistically significant, Dora outperforms OR
  - Dora, Suade, & OR outperform AND (\(\alpha = 0.05\))

- *Integrated lexical- and structural-based approaches can outperform purely structural, but not all lexical scoring mechanisms are sufficient to do so*
Related Work
Automated Program Exploration
Related Work

Automated Program Exploration

- Using program structure from seed starting element
  - Suade [Robillard 2005]
Related Work

Automated Program Exploration

- Using program structure from seed starting element
  - Suade [Robillard 2005]
- Using lexical information in comments and identifiers
  - Regular expressions: grep, Eclipse Search
  - Advanced IR: FindConcept [Shepherd et al. 2007], LSI [Marcus et al. 2004], Google Eclipse Search [Poshyvanyk et al. 2006]
Related Work
Automated Program Exploration

● Using program structure from seed starting element
  ● Suade [Robillard 2005]

● Using lexical information in comments and identifiers
  ● Regular expressions: grep, Eclipse Search
  ● Advanced IR: FindConcept [Shepherd et al. 2007], LSI [Marcus et al. 2004], Google Eclipse Search [Poshyvanyk et al. 2006]

● Additional work in paper
Future Work

- Automatically find starting **seeds**
- Use more **sophisticated lexical information**
  - **Synonyms**, topic words (currency, price related to bidding)
  - **Abbreviation** expansion
- Evaluate on **slicing**
Conclusion

→ *Integrated lexical- and structural-based approaches outperform purely structural ones*

www.cis.udel.edu/~hill/dora

This work was supported by an NSF Graduate Research Fellowship and Award CCF-0702401.
Dora’s Relevance Score
Dora’s Relevance Score

- Developing the relevance score
  - Used logistic regression: predicts values between 0 and 1
  - Logistic regression outputs ‘x’ of the score

\[ score = \frac{e^x}{1 + e^x} \]
Dora’s Relevance Score

- Developing the relevance score
  - Used logistic regression: predicts values between 0 and 1
  - Logistic regression outputs ‘x’ of the score
- Training the model
  - Used methods from 9 concerns in previous concern location tool evaluation [Shepherd 2007]
Dora’s Relevance Score

- Developing the relevance score
  - Used logistic regression: predicts values between 0 and 1
  - Logistic regression outputs ‘x’ of the score

- Training the model
  - Used methods from 9 concerns in previous concern location tool evaluation [Shepherd 2007]

- The model:
  \[ x = -0.5 + -2.5 \times bin + name + 0.5 \times statement \]
Dora’s Relevance Score

- Developing the relevance score
  - Used logistic regression: predicts values between 0 and 1
  - Logistic regression outputs ‘x’ of the score
- Training the model
  - Used methods from 9 concerns in previous concern location tool evaluation [Shepherd 2007]

- The model: \( x = -0.5 + -2.5 * \text{bin} + \text{name} + 0.5 * \text{statement} \)
- Where...
  - \( \text{bin} \) = binary (1 if java file exists, 0 otherwise)
  - \( \text{name} = \sum \text{tf-idf} \) for each query term in the method name
  - \( \text{statement} = \sum \text{tf-idf} \) for each query term in a method statement

\[ \text{score} = \frac{e^x}{1 + e^x} \]
Results: Threshold

Figure 3: Precision-Recall Graph. *Suade* and *Dora* were evaluated at various thresholds ranging from 0 to 1 (*AND* and *OR* require no threshold). Each point represents precision and recall averaged over a given threshold, with decreasing threshold values from left to right.