## Objectives

- Tools for Finding Concerns
  - Concepts
  - Google Eclipse Search
  - Find Concept



## Review

- What is the purpose of tools like make, ant, and maven?
- How are they similar to each other?
- How are they different from each other?
- What don't they do for us? What do we still have to do?



## Motivation

60-90% software costs spent on ***reading*** and ***navigating*** code for maintenance*

**Corrective** (emergency fixes, routine debugging)
**Adaptive** (new environment/input form)
**Perfective** (enhancements)

*Why do maintenance tasks require so much comprehension overhead?*

*[Erlikh] Leveraging Legacy System Dollars for E-Business

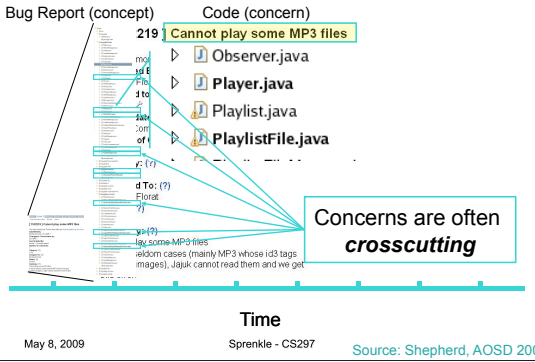
 Source: Shepherd, AOSD 2007

## Concepts/Concerns

- In maintenance, must identify high-level idea (**concept**) **AND** locate, comprehend, and change the concept's implementation (**concern**)
- Example:
  - Concept: Students register for classes
  - Concern: Code that looks up student, determines if student has prerequisite for class, adds student to class list or to a wait list, …



## Motivating Example - Problem

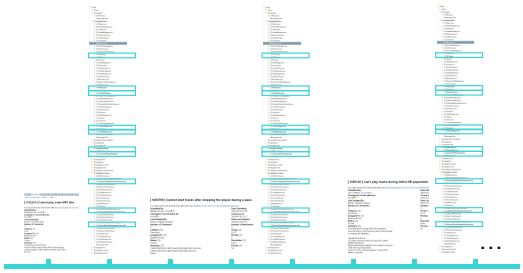



 Source: Shepherd, AOSD 2007

## Motivating Example - Problem


 Source: Shepherd, AOSD 2007

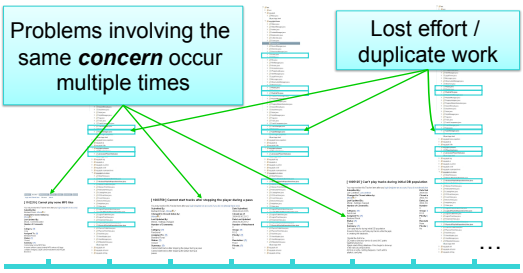
## Motivating Example - Problem



Time


Source: Shepherd, AOSD 2007

## Motivating Example - Problem

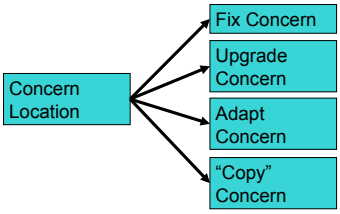



Time


Source: Shepherd, AOSD 2007

## Key Challenge: Concern Location

- Find, collect, and understand all source code related to a particular concept
- Foundation of many software maintenance tasks





Source: Shepherd, AOSD 2007

## GOOGLE ECLIPSE SEARCH (GES)



## "Source Code Exploration with Google"

Denys Poshyvanyk, Maksym Petrenko, Andrian Marcus, XinrongXie, DapengLiu
ICSM 2006

- Problem?
  - Limitations of state of the art
  - Goals



## Limitations of State of the Art

- Regular expressions
  - Less intuitive, strict formulation
- Information retrieval (IR)
  - Formulation of queries with multiple words
  - Problems:
    - Computational efficiency
    - Online re-indexing of software as it changes
    - **Not widespread adoption**
- Google Desktop Search
  - Not project-specific search; searches entire file system
  - Requires an internet browser
    - User has to switch between IDE and the browser

## Goals

- Apply IR search techniques to searching code
  - Needs to be fast
- Solution that is widely adopted



## "Source Code Exploration with Google"

- Problem?
  - Limitations of state of the art
  - Goals
- Approach?
  - Benefit?



## Approach


- Integration of Google Desktop Search + Eclipse Development Environment.
- Plugin: Google Eclipse Search (GES)



## GDS Features

- On-the-Fly preprocessing and indexing of the context
- Continual indexing
  - Maintains and updates content location changes
  - Accurate results
- Immediate response for queries
- History of searches
- Advanced search options
  - Boolean operators
- Sorting of the results
  - Relevance
  - Dates



## Advantages of GDS combined with Eclipse

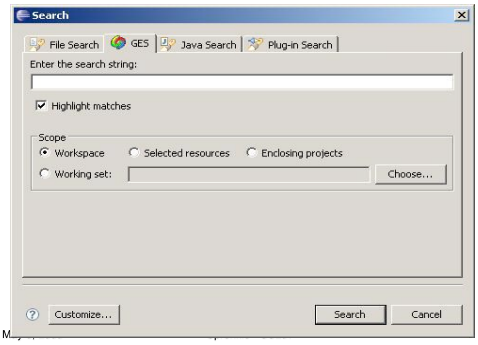
- IR-based searching
  - Multiple term queries
  - Natural language queries
  - Boolean operators
  - ranking of search results
- Scalability & reliability
  - Important for massive file repositories, such as large scale software systems
- Search results in Eclipse
  - Link between search results and source code editor



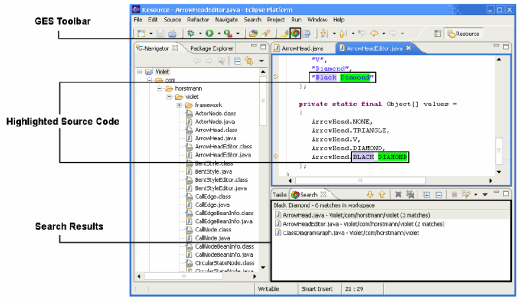
## GES Design & Implementation

- Similar to File Search in Eclipse
- Type a Query into the GES dialogue Box
- Specify the Scope of the search
  - workspace
  - selected resources
  - enclosing projects
  - working sets
- After the query, the search is displayed in GES search Results Tab
- Results can be explored by browsing in the editor

## GES Screenshot




## GES Screenshot






## "Source Code Exploration with Google"

- Problem?
  - Limitations of state of the art
  - Goals
- Approach?
  - Benefit?
- Evaluation?
  - Results?



## Pilot Case Study

- Performed on Violet
  - Cross-platform UML Editor written in Java
  - Has 65 classes + 448 methods + 9000 LOC
- Test with common development task:
  - Request for a new feature: "introduce a user-defined arrow type for the class diagram"



## Queries for PCS-I

- Q1 : "arrow class diagram"
  - No matches
- Q2: "edge class diagrams"
  - Results: 11 Files
  - `UseCaseDiagramGraph, StateDiagramGraph, SequenceDiagramGraph, StateTransitionEdge, ObjectDiagramGraph, NoteNode, ObjectNode, FieldNode, ImplicitParameterNode, ClassDiagramGraph, CallNode`

match



## Analysis of Results

- `ClassDiagramGraph` was relevant to concept "edge class diagrams"
- **But** so were `ArrowHead`, `ArrowHeadEditor`, `ClassDiagramGraphStrings`

Hmm….

## GES vs. Eclipse File Search

- Problem: concept location task in Violet
- Goal: “to locate the place in the source code which specifies the width of the class diagrams”
- File: “value saved in DEFAULT_WIDTH variable”



## Using GES

- Q1: “default width”
  - Gets correct answer



## Using File Search

- Q1: “default width”
  - No results
- Q2: ”default”
  - Closer
- Q3: refine previous results:“width”
  - Much closer
- **BUT**, could have searched for “default*width” for same results as GES
  - Less intuitive, could require more complex query



## Summary of Results

- GES returned good results in the first query
- GES is faster than File Search
- GES returns ranked list of results
- User investigates fewer LOCs with GES
- Developers learn relevant information faster than File Search

Convinced?



## Other Case Studies

- Compare GES and Eclipse search on larger code bases than Violet
- Subjects
  - Art of illusion: Java 3D modeling studio
    - 442 classes , 20 interfaces, 100,838 LOC
  - Eclipse Version 3.1 + complete sources
    - 20000 files
    - 2 million LOC
- Methodology
  - 10 queries were run on each system
  - Measured average response time



## Response Time

**Table 1. Average response time per query for the GES and the Eclipse File Search (in full seconds)**

| | Violet 9 KLOC | AOI 100 KLOC | Eclipse 2 MLOC |
|---|---|---|---|
| GES | 1 s | 1 s | 1 s |
| File Search | 1 s | 9 s | 97 s |

- GES: faster response time
- GES scales well with code size

## "Source Code Exploration with Google"

- Problem?
  - Limitations of state of the art
  - Goals
- Approach?
  - Benefit?
- Evaluation?
  - Results?
- Limitations?
- Conclusions



## Limitations

How much of a limitation is this?

- GES uses GDS
  - GDS is not open-source
  - Requires GDS's background indexing
  - Only when user's computer is idle
  - User has to wait for the (re)-indexing of the file
  - None of the GDS APIs handles this issue
- How **relevant** are the results?
  - Got 11 files but only 1 was relevant
- Small case studies
  - Needs further evaluation to draw conclusions



## Conclusions

- Integrated GDS into Eclipse
  - Improves source code searching
  - Easy-to-adopt approach
- GES allows to perform searches in source code and documentation
- Faster than Eclipse's file search



## GES System Requirements

- Eclipse SDK 3.2 or higher
- Google Desktop Search (GDK) 2.0 or higher
- Java Run-Time Environment (JRE) 1.5 or higher

`http://ges.sourceforge.net/`



## Can We Improve GES?

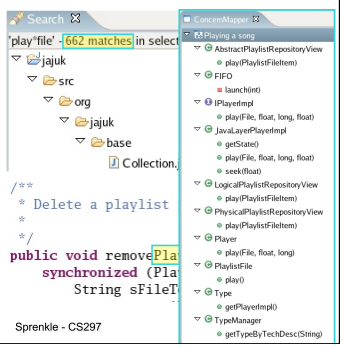
- Think about how we write OO programs and how you figure out how they work
  - Given a problem, what are the objects? What are the objects' methods?
  - Example: recipe writing factory



## Limitations of State of the Art

Source: Shepherd, AOSD 2007

- Return large result sets
- Return irrelevant results
- Return hard-to-interpret result sets



## Motivating Example

**Recipe for Vegetable Soup**

1. Place asparagus and onion in a saucepan with 1/2 cup vegetable broth. Bring the broth to a boil, reduce heat and let simmer until the vegetables are tender.
2. Reserve a few asparagus tips for garnish. Place remaining vegetable mixture in an electric ...

Procedural instructions are easy to read.
Decompose into objects


Source: Pollock talk 2006

## OO System: Beef class

```
//Place the Beef into the pan
place(Pan p){
   checkIfFitsInPan(p);
   actuallyPlaceInPan(p);
}
```

**Bug**: something goes wrong when I place something in the saucepan.

```
/* Make sure the beef fits
in the pan, before Placing
the Beef into the pan */
checkIfItFitsInPan(Pan p)
{ ... }
```

```
//Place the beef into the pan
actuallyPlaceInPan(Pan p) {
 ...
}
```


Source: Pollock talk 2006

## OO System: Beef class

```
//Place the Beef into the pan
place(Pan p){
   checkIfFitsInPan(p);
   actuallyPlaceInPan(p);
}
```

**Bug**: something goes wrong when I place something in the saucepan.

**Problem**: How can we find actions in a noun's world?

```
/* Make sure the beef fits
fore Placing
the pan */
nPan(Pan p)
{ ... }
```

```
//Place the beef into the pan
actuallyPlaceInPan(Pan p) {
 ...
}
```

**Identification problem:** Actions scattered across several methods


Source: Pollock talk 2006

## Key Idea: Users Leave Natural Language Clues in Code

```
//Place the Beef into the pan
Beef.place(Pan p){
   checkIfFitsInPan(p);
   actuallyPlaceInPan(p);
}
```

```
//Place the onion in the pan
Onion.place(Pan p){
...
}
```

```
/* Make sure the beef fits
the pan, before Placing the
Beef into the pan */
Beef.checkIfItFitsInPan(Pan p)
{ ... }
```

```
//Place the beef into the pan
Beef.actuallyPlaceInPan(Pan p)
{
 ...
}
```

**Extract** Clues with Natural Language Processing (NLP)


Source: Pollock talk 2006

"Using Natural Language Program Analysis to Locate and Understand Action-Oriented Concerns"
Shepherd, Fry, Gibson, Pollock, and Vijay-Shanker
AOSD 2007

## FIND CONCEPT



## Find-Concept's Strategy

- Extract *verb-direct object* pairs
- Example: **Place** the **item** in the pan
- Example: **Place** the **beef** in the pan


Source: Pollock talk 2006

## Find-Concept's Strategy

- Extract *verb-direct object* pairs
  - Example: **Place** the **item** in the pan
  - Example: **Place** the **beef** in the pan
- Where to extract
  - Method signatures
  - Comments

```
//Chop the beef into cubes
chop() {
      unwrap();
      slice(CUBES);
}
```

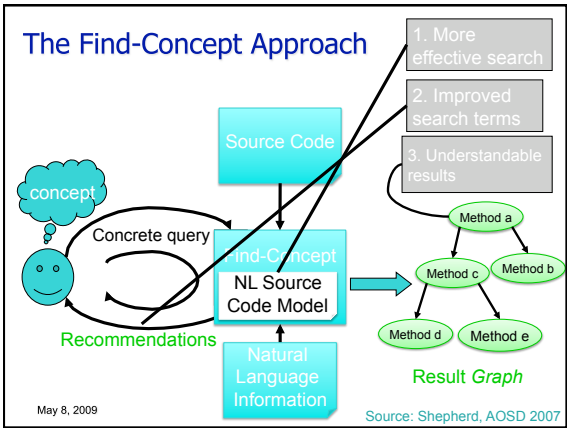
 Source: Pollock talk 2006

## Find-Concept's Strategy

- Extract *verb-direct object* pairs
  - Example: **Place** the **item** in the pan
  - Example: **Place** the **beef** in the pan
- Where to extract
  - Method signatures
  - Comments
- Make recommendations
  - Stemmed/Rooted: *complete, completing*
  - Synonym: *finish, complete*
  - Co-location: *completeWord()*

 Source: Pollock talk 2006

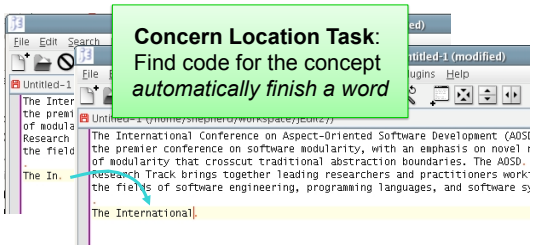
## The Find-Concept Approach




 Source: Shepherd, AOSD 2007

## Example Use of Find-Concept

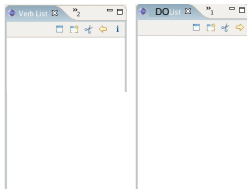
- Target Concern: Automatically finish a word



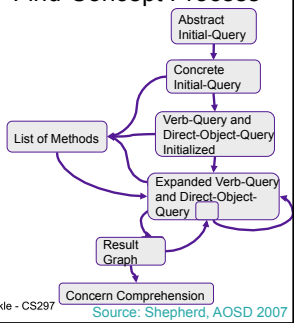

 Source: Shepherd, AOSD 2007

## Example Use of Find-Concept

Query State



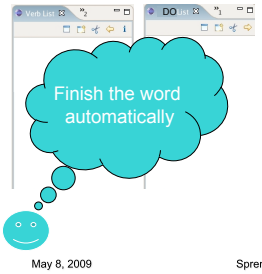
Find-Concept Process



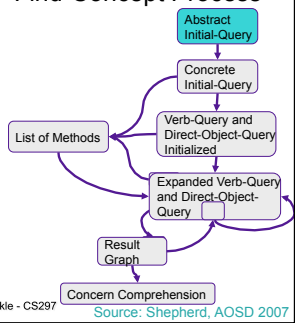

 Source: Shepherd, AOSD 2007

## Forming an Abstract Query

Query State



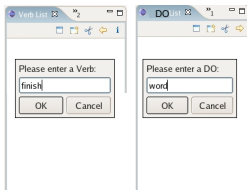

Find-Concept Process



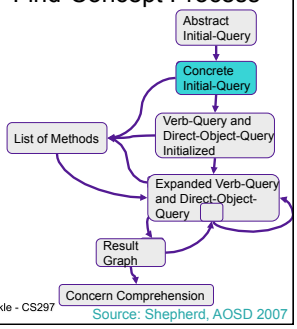

 Source: Shepherd, AOSD 2007

## Forming a Concrete Query

Query State




Find-Concept Process

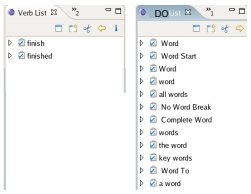




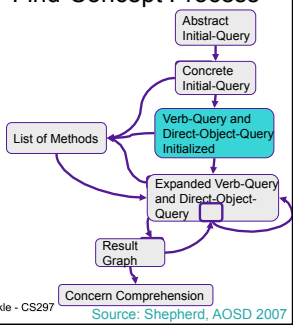
Source: Shepherd, AOSD 2007

## Queries Initialized

Query State



Find-Concept Process

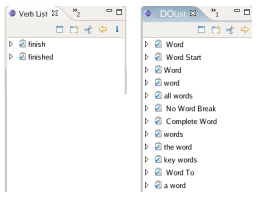




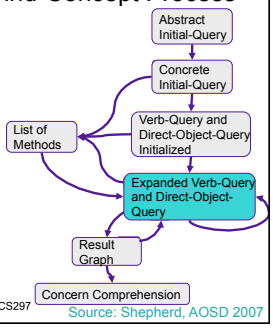
Source: Shepherd, AOSD 2007

## Expand Verb Query

Query State



Find-Concept Process

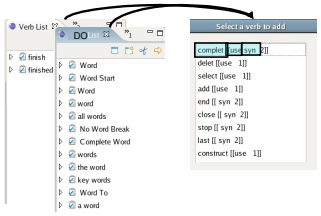




Source: Shepherd, AOSD 2007

## Examine Verb Query Recommendations

Recommendations



Ordered list of verb recommendations

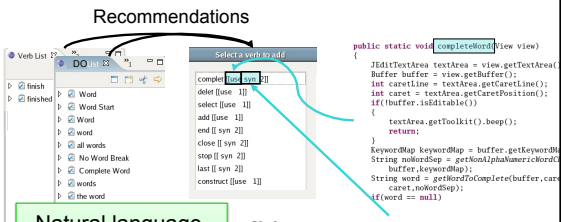
- Stemmed verb
- Summary of reasons


Source: Shepherd, AOSD 2007

Source: Shepherd, AOSD 2007

## Examine Verb Query Recommendations

Recommendations



Natural language **and** program knowledge to make recommendations

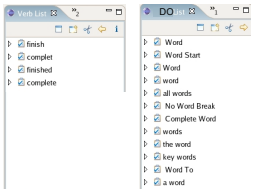
Verb

- S: (v) complete, **finish** (come or bring to a finish or an end) *"He finished the dishes"; "She completed the requirements for her Master's Degree"; "The fastest runner finished the race in just over 2 hours; others finished in over 4 hours"*
- S: (v) finish up, land up, fetch up, end up, wind up, **finish** (finally be or do something) *"He ended up*

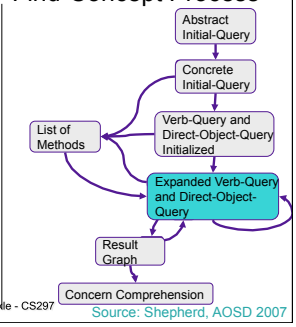


## Expand DO Query

Query State



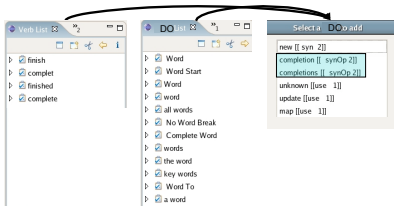
Find-Concept Process





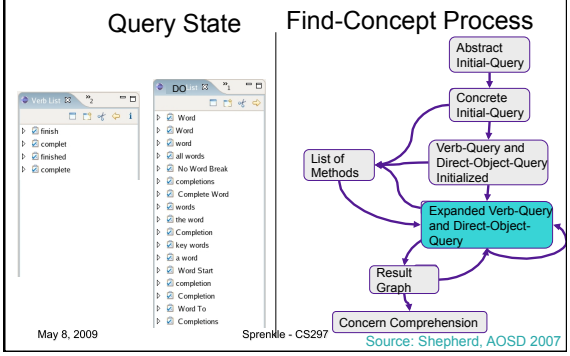
Source: Shepherd, AOSD 2007

## Examine DO Recommendations




Source: Shepherd, AOSD 2007

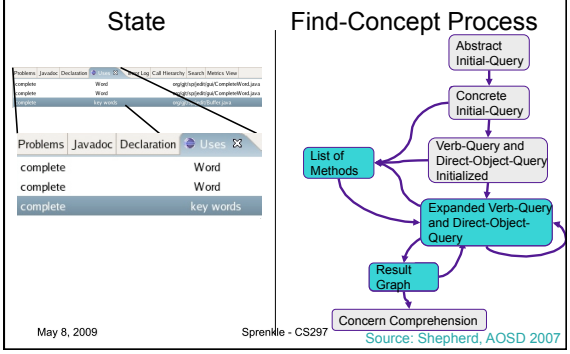
## Examine Intermediate Feedback





Source: Shepherd, AOSD 2007

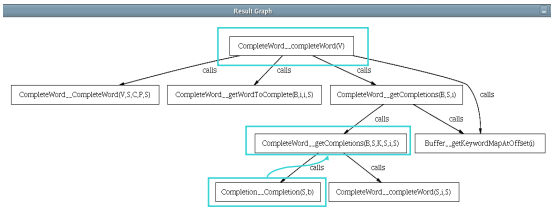
## Examine Intermediate Feedback





Source: Shepherd, AOSD 2007

## Examine Result Graph



- ✓Gain understanding of concern
- ✓Answer common development questions


Source: Shepherd, AOSD 2007

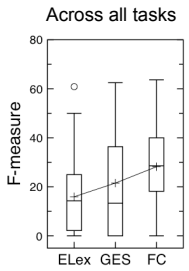
## Experimental Evaluation

- Give users concern location tasks
  - Use Eclipse Lexical Search (ELex), Google Eclipse Search (GES), and Find-Concept to find relevant methods
- Measure query's effectiveness
  - **Precision**: ratio of # of relevant procedures retrieved to total # of procedures retrieved
    - High precision: result set contains *few irrelevant* results
  - **Recall**: ratio of relevant procedures retrieved to the total # of relevant procedures existing in the source application
    - High recall: *most* of *relevant* results are included in the result set
- Measure user effort
  - Measured the amount of time each user required to form a satisfactory query for each task


Source: Shepherd, AOSD 2007

## Experimental Evaluation

- Query Effectiveness
  - FC > Eclipse Search with statistical significance
  - FC >= GES on 7/9 tasks
  - FC is more consistent than GES
- User Effort
  - FC = Eclipse Search = GES




FC is more consistent and more effective in experimental study without requiring more effort


Source: Shepherd, AOSD 2007

## Analysis

- What didn't the authors evaluate that you might be interested in?



## Limitations

- What are Find-Concept's assumptions?
  - How do those assumptions limit its effectiveness?



## Limitations/Assumptions

- Comments: exist, descriptive, accurate, right level
  - May not exist, may be misleading, inaccurate, at a different granularity
- Variable and method names are descriptive, accurate
  - May be misleading



## Next Paper: Dora the Program Explorer

- Use *program structure*
- Unfamiliar terms:
  - Call graphs, method call chains
  - Control flow
  - Variable def-use
  - Type hierarchy
- Due Monday



## Reminder: Midterm Wednesday

- 15% of grade
- Focus on UNIX commands, Bash scripting
  - UNIX philosophy
  - Reading and writing Unix commands
  - Understand purpose of various tools
- Software tools
  - What can they do?
- Tool types we've covered so far
  - Build tools
  - Search/navigation tools
- Bring questions on Monday



## Do Overs

- Submit
  - New assignment on paper and electronically
    - Directory name: <assign#>.doover
  - Original assignment
    - So I can allocate points appropriately
- Due 1 week after returned