CHESS

CENTER FOR HYBRID & EMBEDDED SOFTWARE SYSTEMS

COLLEGE OF ENGINEERING, UC BERKELEY

Prospector: A Sample Code Search Engine for Complex APIs

David Mandelin (UCB)
Rastislav Bodik (UCB)
Lin Xu (UCB)
Doug Kimelman (IBM)

http://chess.eecs.berkeley.edu

Visit the Prospector web site
Pre-alpha download, functional web demo
http://snobol.cs.berkeley.edu/prospector

Why Prospector?

- Software development practice is centered on code **reuse** enabled by **frameworks** and **libraries** – usually with large, complex APIs.
  - In Java: J2SE, J2EE, Eclipse, Jakarta, Colt, …
- Access to **system resources** and **external devices** is provided through complex APIs.
  - E.g., SWT (Java GUIs), Win32, …
- Reuse has great potential for productivity gains, and APIs for system access are essential, but often the **potential of reuse is not realized**, because programmers end up spending hours or days searching documentation for seemingly simple functionality.
  - Example:
    - The authors wanted to use Eclipse (IBM's Java IDE and rich client framework) to parse a Java file.
    - In fact, parsing Java is well supported in Eclipse, requiring only **two lines of code**.
    - Yet, it took **two days** to figure out:

```
ICompilationUnit icu =
    JavaCore.createCompilationUnitFrom(file);
CompilationUnit cu  =
    AST.parseCompilationUnit(icu, false);
```

    - **With Prospector, a programmer can write this code in two seconds.**

Solving API Problems

Prospector is both a source code **search** engine and a source code **synthesis** engine.

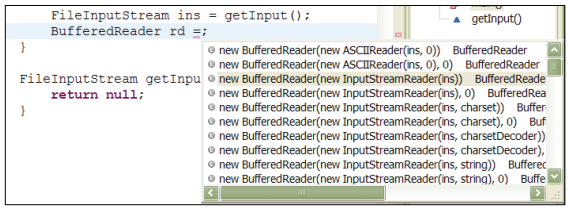
Prospector finds code snippets to solve problems of the form "I **have** an object of type A, but I **need** an object of type B."

- Parsing example: "I have an IFile, I need an ASTNode."

Code snippets are synthesized from method signatures and examples in existing code and presented as **immediately usable Java code**.

Coding with Prospector

- Our prototype integrates Prospector's search technology with Eclipse Java Content Assist (a.k.a. code completion).
- With Prospector, Content Assist offers **complete code snippets**, which can be quite complex if necessary: they may contain multiple method calls, field accesses, array accesses, and downcasts



Prospector is integrated with Eclipse Java Content Assist

Examples

★ Example 1. How do I make Eclipse Java editor show a message box?

MessageDialog.showInformation() shows a message box, but needs a Shell argument. Prospector can show how to get from IJavaEditor to Shell:

```
editor.getSite().getShell()
```

★ Example 2. How do I find a view object in Eclipse?

A view is represented by IViewPart. We can guess that the workbench, represented by IWorkbench, is a good starting point for finding a view. Prospector gives us the answer:

```
IWorkbenchPage page =
    workbench.getActiveWorkbenchWindow().getActivePage();
IViewPart viewPart = page.showView(string);
```

★ Example 3. How do I read lines from a FileInputStream?

There is a readLine() method in BufferedReader, so we just need code to wrap a FileInputStream in a BufferedReader. Prospector says:

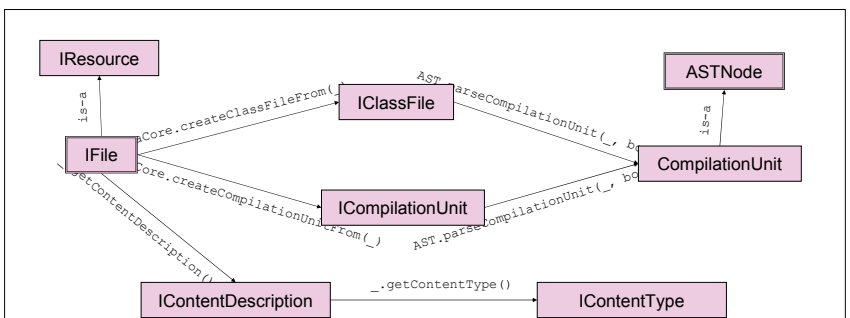
```
new BufferedReader(new FileInputStreamReader(is))
```

★ Example 4. How do I open a Java NIO FileChannel?

FileChannel has no constructor, so it's hard to see how to make one. We can guess that there *is* a way to get from File to FileChannel, and Prospector will give it to us:

```
new FileInputStream(file).getChannel()
```

Prospector Search Technology



The **jungloid graph**: a model of the API

The **jungloid graph** models the API.

A **jungloid** is an expression that connects a sequence of types. Think of a twisty path through the object jungle.

Nodes represent classes.

Edges represent instance methods, static methods, fields, extends relationships, and implements relationships

The jungloid graph is initially constructed from class declarations and method signatures.

Source code examples are extracted and added as new paths to the graph for hard-to-handle constructs such as downcasts.

To answer a Prospector query, simply find all paths between the input and output classes and convert them to Java!