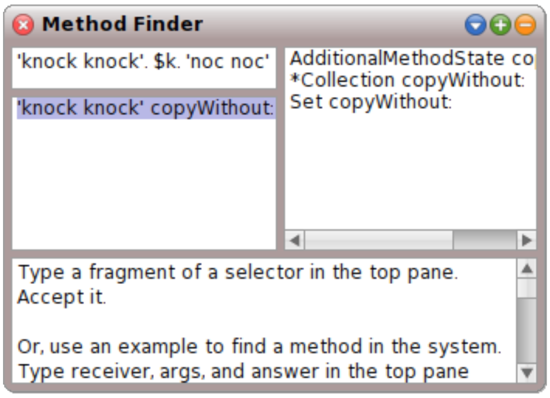
Example-driven Exploration of Dynamically-typed Programming Environments

Bachelor Project Proposal, WS 2014/2015 – SS 2015
Software Architecture Group, Prof. Dr. Robert Hirschfeld

Squeak’s Method Finder

The Message Finder for Squeak allows programmers to type in any fragment of a message name and see a list of all the matching methods in the system whose name contains the fragment. Such facilities are widely available in development environments for text-based programming languages.

The *Method Finder*, in contrast, offers *search by example*. Suppose that you are new to Squeak and you want to copy a string with every occurrence of a letter removed. There probably already is a method that does this, but you don't know its name. The top pane in the screenshot shows what to type into the Method Finder. The string 'knock knock' and the character $k are the inputs. 'noc noc' is the answer you want the method to return. The Method Finder finds “copyWithout:”. It does this even though “copyWithout:” takes a string as an argument, and the example input has a character.

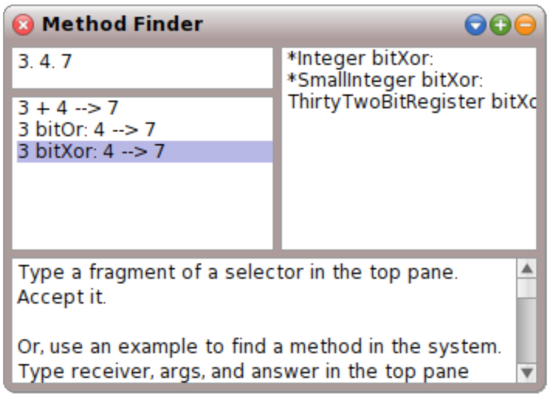


```
'knock knock'. $k. 'noc noc'
'knock knock' copyWithout:
```

```
AdditionalMethodState co
*Collection copyWithout:
Set copyWithout:
```

Type a fragment of a selector in the top pane.
Accept it.

Or, use an example to find a method in the system.
Type receiver, args, and answer in the top pane



```
3. 4. 7
3 + 4 --> 7
3 bitOr: 4 --> 7
3 bitXor: 4 --> 7
```

```
*Integer bitXor:
*SmallInteger bitXor:
ThirtyTwoBitRegister bitXo
```

Type a fragment of a selector in the top pane.
Accept it.

Or, use an example to find a method in the system.
Type receiver, args, and answer in the top pane

Here is how it works: The Method Finder actually executes around 850 of the most common methods in Squeak. It uses the example objects supplied by the user. It tries all permutations of the receiver and arguments, since the user can't be expected to know the proper order. All errors are caught and suppressed, and the search runs quickly.

Project Objectives

Experienced programmers used to be able to know the libraries they rely on by heart. However, with an increasing interest in programming as witnessed from the open source community, the amount of possibly useful code available to the programmers has grown beyond comprehension.

To address this problem, code recommendation and synthesis systems have moved into the focus of active research. These support their users in finding elements of interest in the context of their current programming efforts. To that end they rely on metrics based mostly on static artifacts like source code.

In the context of this project, participants will help to improve code recommendation by following a more dynamic, example-data driven approach. Starting from Squeak's Method Finder, they will

- explore state-of-the-art code recommendation and synthesis systems,
- revisit Method Finder's concepts and implementation, and
- design and implement a next version and its integration in the Squeak environment.

The implementation will involve the use of Squeak's mechanisms for meta-programming and computational reflection. New and modified tool support will take advantage of the improved Method Finder.

Organization

A group of six to eight (6-8) students may participate in the project. Organization and tasks will be determined by the project participants, following an agile development process. The project will be carried out at the Hasso Plattner Institute in Potsdam. Project participants are expected to communicate with our partner on a regular basis. In WS 2014/2015, participants will work on initial design sketches and prototypes. Main steps in design and implementation are to be executed in SS 2015.

Partner & Contact

Ted Kaehler
Viewpoints Research Institute, Los Angeles, California, USA
http://www.vpri.org/, ted@vpri.org

Prof. Dr. Robert Hirschfeld, Tim Felgentreff, Tobias Pape, Marcel Taeumel
Software Architecture Group, Potsdam
http://www.hpi.uni-potsdam.de/swa, hirschfeld@hpi.uni-potsdam.de