An oracle for object-oriented programmers

October 7 2011, by Larry Hardesty

In the last 40 years, the major innovation in software engineering has been the development of what are called object-oriented programming languages. "Objects" are, effectively, repositories for the computational details of a program, which let the programmer concentrate on the big picture. A complex computer program, with millions of lines of code, can be distilled into some fairly intuitive interactions between objects.

For programmers building a large application from scratch, object-oriented programming is a boon, allowing them to add new functions or make major revisions by changing just a few lines of code. But for a programmer dropped into the middle of a massive development project, trying to navigate the thicket of existing objects can be bewildering. Learning what the objects are and what they do might take days or even weeks.

At the Association for Computing Machinery's SPLASH conference at the end of the month, researchers from MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) will present a new system that automatically determines how objects in a large software project interact, so it can inform latecomers which objects they will need to design certain types of functions. The system could be of particular use to programmers working with open-source software, whose licensing terms require that its underlying code be publicly disclosed. Someone wishing to simply add a function to a common open-source program, for instance, may not want to spend the week it takes to get up to speed on all the program's objects.

"Part of the promise of open source is that if you don't like what it does, you can go in and change it," says Armando Solar-Lezama, the NBX Career Development Assistant Professor of Computer Science and Engineering, who led the work. "But if you have this huge learning curve, then you're not going to be able to do that. You're going to end up with a small group of experts who go and do all the stuff, and everyone else just uses it."

Objectivity

The idea of the object is easiest to understand when the programmer's object — a cluster of data and a set of associated functions — corresponds to a virtual object on-screen. A programmer wishing to add a new window to an application, for instance, simply writes a line of code calling up a new window object; the window comes complete with things like scroll bars and size-adjustment tabs and a display line for text. If the programmer wants to add a button to the window, she calls up a new button object.

But after that, things can get more complicated. To describe the layout for the window, the programmer may have to invoke an object called

Layout; to enable the button to register mouse clicks, she may have to invoke an object called EventListener. These don't appear on-screen as virtual objects, but in the programmer's sense, they're objects nonetheless.

"In some respects, this is a great design," Solar-Lezama says. "It's beautifully engineered to allow you to just take out little pieces of the functionality and replace them without having to go and write lots of code. But the price of that is that you have to know how it works before you can use it."

Solar-Lezama and his students Zhilei Xu and Kuat Yessenov have developed a new system they call Matchmaker, because it takes as input the names of two objects and describes how to get them to interact with each other. To demonstrate how it works, the researchers applied it to an open-source program called Eclipse, which computer scientists use to develop programming tools for new computer languages.

In the Eclipse framework, the window that displays code written in the new language is called an Editor; a function that searches the code for symbols or keywords is called a Scanner. That much a seasoned developer could probably glean by looking over the Eclipse source code. But say you want to add a new Scanner to Eclipse, one that allows you to highlight particular symbols. It turns out that, in addition to your Editor and Scanner objects, you would need to invoke a couple of objects with the unintuitive names of DamageRepairer and PresentationReconciler and then overwrite a function called getPresentationReconciler in yet a third object called a SourceViewerConfiguration.

With Matchmaker, the developer would simply type the words "editor" and "scanner" into the query fields, and the program would return the names of the objects that link them and a description of the modifications required to any existing functions.

Observe and detect

Matchmaker builds up its database of object linkages in a program's source code by monitoring the program's execution. In the case of Eclipse, it noticed that every time a Scanner was invoked, so were the other objects.

On occasion, Matchmaker's inferences may turn out to be wrong. But even in those cases, Solar-Lezama argues, some guidance is better than none. To test that thesis, Solar-Lezama and his colleagues did a user study with eight programmers new to Eclipse. All of them were asked to perform the same task, which required linking up two different types of objects. Four of them were allowed to use Matchmaker, and four weren't. Moreover, the example was specifically chosen so that the instructions provided by Matchmaker were incomplete: They left out one crucial step. Nonetheless, Solar-Lezama says, the programmer who completed the task most quickly without Matchmaker still took longer than the slowest of the programmers who used it.

“I think the user study is the linchpin that makes this seem like it's got real practical implications,” says Jeff Foster, an associate professor at the University of Maryland who's part of the Department of Computer Science's Programming Languages group. “If you can hand the tool to somebody else and they can be better programmers when they're using it, that's a great result.”

Foster points out that Matchmaker can't answer all the questions that a programmer new to an application might have. Matchmaker is useful, he says, “in cases where you can guess that you needed an X and Y, and you could find the names of those components, and you can ask how they're connected. But there's a whole other set of questions where you don't even know what components to use.”

Still, Foster says, in the instances where Matchmaker is applicable, "you're going to get result that's vastly superior. You're going to get a result in seconds or minutes instead of having to search on Google, filter out a lot of bad answers, figure out what people meant when they explained various things — if the tool works for the problem, it's very useful."

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology

Citation: An oracle for object-oriented programmers (2011, October 7) retrieved 26 April 2024 from https://phys.org/news/2011-10-oracle-object-oriented-programmers.html