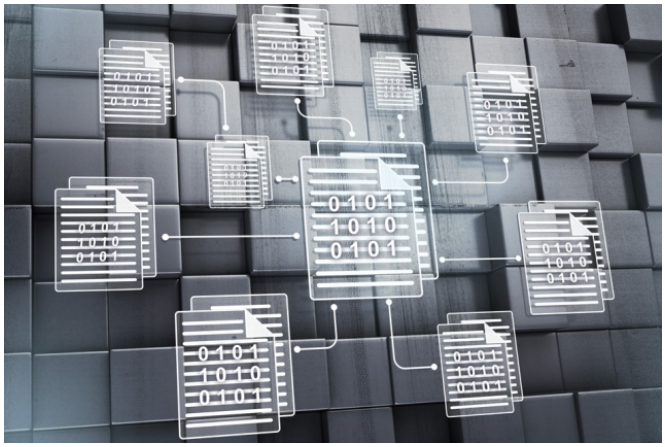
System makes modifications necessary to transplant code from one program into another

20 September 2017, by Larry Hardesty



“CodeCarbonCopy enables one of the holy grails of software engineering: automatic code reuse,” says Stelios Sidiroglou-Douskos, a research scientist at CSAIL. Credit: MIT News

Researchers at MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) have developed a new system that allows programmers to transplant code from one program into another. The programmer can select the code from one program and an insertion point in a second program, and the system will automatically make modifications necessary—such as changing variable names—to integrate the code into its new context.

Crucially, the system is able to translate between "data representations" used by the donor and recipient programs. An image-processing program, for instance, needs to be able to handle files in a range of formats, such as jpeg, tiff, or png. But internally, it will represent all such images using a single standardized scheme. Different programs, however, may use different internal schemes. The CSAIL researchers' system automatically maps the donor program's scheme onto that of the recipient, to import code seamlessly.

The researchers presented the new system, dubbed CodeCarbonCopy, at the Association for Computing Machinery's Symposium on the Foundations of Software Engineering.

"CodeCarbonCopy enables one of the holy grails of software engineering: automatic code reuse," says Stelios Sidiroglou-Douskos, a research scientist at CSAIL and first author on the paper. "It's another step toward automating the human away from the development cycle. Our view is that perhaps we have written most of the software that we'll ever need—we now just need to reuse it."

The researchers conducted eight experiments in which they used CodeCarbonCopy to transplant code between six popular open-source image-processing programs. Seven of the eight transplants were successful, with the recipient program properly executing the new functionality.

Joining Sidiroglou-Douskos on the paper are Martin Rinard, a professor of electrical engineering and computer science; Fan Long, an MIT graduate student in electrical engineering and computer science; and Eric Lahtinen and Anthony Eden, who were contract programmers at MIT when the work was done.

Mutatis mutandis

With CodeCarbonCopy, the first step in transplanting code from one program to another is to feed both of them the same input file. The system then compares how the two programs process the file.

If, for instance, the donor program performs a

series of operations on a particular piece of data and loads the result into a variable named "mem_clip->width," and the recipient performs the same operations on the same piece of data and loads the result into a variable named "picture.width," the system will infer that the variables are playing the same roles in their respective programs.

Once it has identified correspondences between variables, CodeCarbonCopy presents them to the user. It also presents all the variables in the donor for which it could not find matches in the recipient, together with those variables' initial definitions. Frequently, those variables are playing some role in the donor that's irrelevant to the recipient. The user can flag those variables as unnecessary, and CodeCarbonCopy will automatically excise any operations that make use of them from the transplanted code.

New order

To map the data representations from one program onto those of the other, CodeCarbonCopy looks at the precise values that both programs store in memory. Every pixel in a digital image, for instance, is governed by three color values: red, green, and blue. Some programs, however, store those triplets of values in the order red, green, blue, and others store them in the order blue, green, red.

If CodeCarbonCopy finds a systematic relationship between the values stored by one program and those stored by the other, it generates a set of operations for translating between representations.

CodeCarbonCopy works well with file formats, such as images, whose data is rigidly organized, and with programs, such as image processors, that store data representations in arrays, which are essentially rows of identically sized memory units. In ongoing work, the researchers are looking to generalize their approach to file formats that permit more flexible data organization and programs that use data structures other than arrays, such as trees or linked lists.

"In general, code quoting is where a lot of problems in software come from," says Vitaly Shmatikov, a professor of computer science at Cornell Tech, a joint academic venture between Cornell University and Israel's Technion. "Both bugs and security vulnerabilities—a lot of them occur when there is functionality in one place, and someone tries to either cut and paste or reimplement this functionality in another place. They make a small mistake, and that's how things break. So having an automated way of moving code from one place to another would be a huge, huge deal, and this is a very solid step toward having it."

"Recognizing irrelevant code that's not important for the functionality that they're quoting, that's another technical innovation that's important," Shmatikov adds. "That's the kind of thing that was an obstacle for a lot of previous approaches—that you know the right code is there, but it's mixed up with a lot of code that is not relevant to what you're trying to do. So being able to separate that out is a fairly significant technical contribution."

More information: CodeCarbonCopy: people.csail.mit.edu/rinard/pa … 7.codecarboncopy.pdf