My Response to How Semantics-Preserving Transformation of Code

Might Change the Way that Software is Developed

In Dr. Roberts’ lecture on Monday, Dec 1st, Dr. Roberts discussed his ongoing work of translating archaic code from Delphi Pascal to a more modern and supported C#. Rather than re-writing the code from scratch -- an endeavor that would be incredibly tedious, economically unfeasible, and bug-prone, Dr. Roberts is employing a method of source-to-source transformation. Using software that he and his colleague developed, Dr. Roberts is attempting to “translate” one language into the other, much like translation is done from one natural language (for example, English), into another (for example, French).

I think that the source-to-source translation is a powerful and very original technique. Even in my own experience, I have found that I have often written some code similar to what I need for a given project, only to discover that it was written in Visual Basic or Java instead of the required C++. I am sure that, in the global community of computer scientists, who write (or have written) in many dozens of languages, some of which are incredibly archaic at this point, a source-to-source transformation of code could bring back some of that “dead” code back to life. After all, it is wasteful to have to “reinvent the wheel” with each advance in a language, and it is also impractical to cling on to dead languages with a faltering amount of support and libraries, just because a large amount of code has been written in those language. Source-to-source transformation would allow code to be re-used without the painstaking process of re-writing the programs from scratch (which would not only be costly, but would also disappoint consumers
who are waiting for new features or bug fixes, and would potentially even introduce new bugs along the way).

Source-to-source translation of code could even be used to translate messy code into a more readable rendition of that code in that same language. That way, auto-generated code (a very common occurrence in GUI-building, for instance) can be cleaned up by the computer before presenting it to the bewildered user. A similar technique could even be used to go through code and remove unnecessary assignments and other clutter that builds up over time: for example, assigning a value to a temporary variable, only to assign it to another variable, and then promptly passing it to another function. The above would also be a nice supplement to the source-to-source transformation between languages: for example, after translating everything from Delphi to C#, the mechanism could go and clean up the C# code to look more like actual C#, versus a C#-rendition-of-formerly-Delphi-code.

I believe that Dr. Roberts’ work provides a very powerful technique both for re-using archaic code, reusing current code that was written in another language, and even cleaning up code that was auto-generated or that has accumulated patches and various clutter over the course of time. The lecture was indeed very interesting; and I’d be very curious to see further research in the field, along with actual consumer products for code translation.