iEx6

Derek Harber

November 14, 2011

Three services that a good programming environment should provide are the following: The first service an environment should include is good API for the language that is being used. For example, Eclipse IDE has excellent API in that it will predict which built in Java methods you might be using and make the appropriate suggestions. This is mainly for efficiency. Instead of searching deep in the Java background code for a certain method, the environment should have information on methods that might be used in that context. The second service is syntax checking. If an environment knows which language the programmer is using, it should be able to check the syntax on it and inform the programmer if there is a mistake in his/her syntax. The third service that a good programming environment should have is a Debugging tool. A programmer could provide print statements throughout the code to debug, but sometimes that can only go so far. Sometimes it is not even possible to include those statements in some languages, so the environment should provide a step-by-step debugging tool so in case invisible errors arise, the programmer can go through step by step to see what the error is.

The following are three things I liked about the Dracula programming environment: First, the check syntax button. Since my computer was not exactly up to speed the entire semester. Taking the time to run the entire code to find a syntax error would have been costly. The check syntax button allowed for checking the syntax without running the entire code. This feature made my coding much more efficient. Second I liked the Debug tool. Littering the ACL2 code with print statements was not feasible and would have taken too long compared to other languages. Having a debug tool in place to go step-by-step through the code helped a lot when trying to find run-time errors though this functional code. Lastly, I liked the parentheses shading that Dracula utilized. In a language like ACL2, parentheses are quite important. Shading from an opening parenthesis to its corresponding closing parenthesis helps tremendously when trying to find where a syntax error is, especially with larger and the more intricate methods.

The following are three things I did not like about the Dracula programming environment: First, the lack of API. Considering the intricacy of ACL2, and the specifics of ACL2, Dracula needs to have a better API system for the programmer. Second, I was not a fan of the error messages that Dracula produces when an error arose. Often times and error would come up that just said “Bad syntax” or something that was embedded in the background ACL2 code, but every time an error would come up, it would not reference where the error occurred in the actual code, just in the background code. I would take hours to try to find the source of the errors produced because there was no indication of where the error occurred. It became the most frustrating part of programming in Dracula. And Lastly, I did not like the Debug tool used in Dracula. I know I listed this as one of the things I liked about in Dracula, but that was only when it was working correctly. I could get the debug tool to work only a couple of times throughout the semester. It got to the point where every time I would start up the debug tool, it would shut down the whole Dracula instance, saved or not. This cost me a few times and I eventually stopped trying to use it and it became useless. So when it was working it worked well, but it hardly ever worked for me.

THE Organization

123 Made-up Street,

Somewhere, AT 55555

11-15-2011

To: Software Development Team

From: Derek Harber

Subject: PROGRAMMING ENVIRONMENT

As all of you know, we have received a contract to design and implement control software for a new kind of nuclear reactor. One of the main characteristics of the reactor that we should keep in mind when designing this control software is its location. It will be in a densely populated area. We should design software that can keep all these people safe from this impending threat that nuclear reactors present.

One of my tasks I was assigned is to determine a good programming environment that we can all use to develop this software. Keeping many of the requirements in mind and some of the project limitations in mind, I reviewed many different environments and have come to a decision on which environment will suit our group the best.

We will be using Visual Studios for all of our needs in regard to this project. We will use this for the following reasons: Every single one of use have extensive experience with Visual Studios, so there will be virtually no learning curve while trying to implement this code. Next, the API in Visual Studios will allow us to use a plethora of languages, which since we are not sure which language will suit us best for the project, Visual Studios will give us the most flexibility in this aspect. And Finally, Visual Studios provides an easy way to export code and to use it on any platform that we see fit. When choosing the right programming environment, flexibility was a high priority of things to see. By meeting all of my requirements, I hereby say that we will use Visual Studios. If you have any questions about the decision, or if you have any other environments that you think will work better than Visual Studios, let me know and I will consider it.

Derek Harber