Youming Lin

Dr. Rex Page

CS 4263

14 November 2011

iEx6 Programming Environment Survey

1. A good programming environment should provide a clean interface with little clutter that organizes its GUI to emphasize code generation (large editor window, IntelliSense, built-in debugger, etc.). In addition, the programming environment should not hinder a programmer by removing standard features of a programming language. Finally, the programming environment should provide access to commonly used libraries to augment the capabilities of the programming language.
2. Having used the Dracula programming environment for the past semester, I found a few things I liked about Dracula (and DrRacket in general). First, Dracula provides a clean interface with little clutter. The main panel of the GUI displays the code for the program I am working on, with little to no formatting or additional features. The code is exactly what I see in Notepad, with the benefit of color coded syntax. Second, Dracula provides a library with commonly used functions that can be hard or repetitive to code. In addition, the library provides functionalities like file I/O and GUI output that are not found in the default ACL2 functions. Lastly, Dracula provides a feature for proving the logical correctness of the functions I code. This feature is a welcome addition to any code review.
3. While I enjoyed coding in Dracula (or rather, DrRacket), I found a few things I disliked about the programming environment. First, Dracula does not always supply easy to understand error messages, especially the “blame” errors like

"collects\racket\contract\private\blame.rkt". Second, Dracula does not implement all ACL2 features. For example, "(< (complex 2 3) (complex 3 4))" returns an error even though ACL2 allows for the comparison of complex numbers with the "<" operator. Lastly, the programming environment provides little editor functions. While I appreciate the simplicity of the editor in Dracula, I would like it more if it implemented common IDE features like code folding and IntelliSense.

4. Memorandum

To: Control Software Project Manager

From: Youming Lin, Software Engineer

Priority: Medium

Date: 11/14/2011

Subject: Control Software Programming Environment

Dear Project Manager:

After an extensive research on the choice of programming environment for our upcoming control software for a new kind of nuclear reactor, I recommend that we use the Dracula programming environment for software development.

Dracula provides an important feature for code testing: theorem proving. Since our control software will be used in highly sensitive situations (controlling a nuclear reactor in a densely populated area), we need to be able to prove beyond a doubt that our software will work without fail. Dracula provides this function. We can enforce each function we write in Dracula to perform exactly as we want it to, thereby increasing the reliability of our code.

Furthermore, Dracula provides a well-rounded programming environment. It includes common IDE features like debugging, as well additional libraries that provide functionalities

such as file input/output, an AVL tree, and GUI components. Navigating its interface is easy due to Dracula's minimal menus compared with the cluttered interfaces of more popular programming environments like Eclipse IDE or Microsoft Visual Studio.

Based on its incomparable theorem proof feature and its simple and clean interface, I recommend that we use the Dracula programming environment to develop the control software for a new kind of nuclear reactor to be constructed in a densely populated area.