Nathan Breaux

CS 4263

iEx6

15 November 2011

Programming Environment Survey

The first service that a good programming environment should provide is effective real-time syntax checking. This saves the programmer time being able to see what they did wrong syntactically before compiling the program. This is especially helpful for languages such as lisp, where it can be easy to make a mistake because of the parenthesis. The second service a good programming environment should provide is code blocks. By this I mean the ability to expand and collapse blocks of code, as well as the ability to jump to specific blocks of code. This makes moving through and editing thousands of lines of code more manageable. The last feature that a good programming environment should provide is a good testing service. I have not used the test driven development method before this course, but it has proved to be extremely helpful in the course of development. It creates solid code and saves future debugging time.

The first thing I enjoy about the Dracula programming environment is the testing feature. Like I stated previously I found the test driven development method to be effective in creating solid, error free code. This proves to save future debugging time and forces you to write better code. The second service of Dracula that I liked was the highlighting feature. When first starting to write in lisp it can be very frustrating trying to line up parenthesis correctly. The highlighting feature proved invaluable in the development process. The third feature of Dracula that I enjoyed was the ability to jump to specific functions in the code from a drop down menu. This saved development time being able to quickly jump to and edit specific blocks of code.

The first service of Dracula that I did not like is the documentation. More than half of the teachpacks documentation is still under construction and the only way to find out the features of a book is to look at the source code. While this was just a minor annoyance for me because I had people to ask, for someone beginning to develop in this environment on their own it could prove to be a major frustration. The second thing about Dracula that I did not like was its lack of real-time error checking. This could have saved me a lot of time when beginning to develop in lisp since most of the entries in my defect logs were syntax errors. The last service of Dracula that I did not like is the errors. Usually when you throw an error while compiling the environment will tell you what line the error occurred on. While the errors where helpful in describing what went wrong, it was sometimes hard to find where it went wrong.