Charles Maingi

iEx6 Programming Environment Survey

November 12, 2011

Describe three services that a good programming environment should provide.

In my opinion, one of the most important services that a programming environment should provide is syntax error evaluation as a user is writing the software. It is crucially helpful to be able to correct syntax error as they occur because these errors might affect the structure of a program. If a user waits until compile time to realize that he or she has multiple errors in the code that will change the structure of their program crushes the individuals work enjoyment. A second service that I think is crucial is a good documentation database, this documentation should be required to give simple examples of the functions being explained. If an environment requires an individual to use trial and error for most of its functions, then the individual develops dread of using the environment. A third and last crucial requirement for a programming environment is the debugging function. In software programming, when a program becomes bigger than one thousand lines of code, trying to find a bug in that program becomes a nightmare, therefore a good debugging service is crucial.

Describe three things you like about the Dracula programming environment.

One good thing about the Dracula programming environment is the fact that it is actively checking for errors in a program as an individual is coding, for example, if a user tries to use a function in another file that is not included in the current file, the environment gives an error that the function is not recognized. This is very useful because the programmer can trace the program and not have to dig very deep into the code to find the problem. A second strength for Dracula is the fact that it has a lot of prebuilt functions, this is a good thing but it has its weakness, I will explain the weakness in the next question. The prebuilt functions are numerous and if there is a function that a programmer wants to perform and it seems general, then it is best to look around, chances are it already exists. A third strength of Dracula is the function property checking capability, I have not seen another environment that an individual can test a function for random number of input size. This capability gave me more confidence that my functions will do what I design them to do for untested input size. I am usually constantly having doubts in my function in languages such as Java but the Dracula defproperty function strengthened my confidence.

Describe three things you dislike about the Dracula programming environment.

Without hesitation, my biggest dislike about Dracula was the lack of examples for prebuilt functions, although the array of prebuilt functions was large, the programmer was forced to utilize trial and error to check how the function works. A function such a CW says that it is supposed to print to the comment window, although I needed on multiple instances to print to screen, to this very day I still do not know how to use this function. A second dislike about Dracula is that the use of its debugging function is complicated, debugging should be intuitive and easy to understand. Although I am not a strong user of debugging, the Dracula debugging environment intimidated me earlier on in my use of the environment that I resulted to being comfortable with step by step trial and error debugging. A third dislike of Dracula for me is the fact that a lot of the function are present in modules that are almost impossible to know that they exist, for example, a function such as str->rat which is in the io-utilities teachpack is not easy to know that it exist. I needed to use this function in one of my programs and if I had not seen my teacher use it somewhere in his examples, I would have rewritten the function. Therefore, Dracula user convenience highly depends on who you know and how long you have been coding with the environment.

TO: Reactor Builders.

FROM: Charles Maingi

DATE: November 12, 2011

SUBJECT: Programming Environment Choice.

The Purpose of this memo is to give a recommendation for the programming environment to use in the software development for the control software of a nuclear reactor. The severely serious nature of this recommendation requires serious research into the strength and dependability of as many environments as possible. The programming Environment that is recommended for this task is the Dracula programming environment. Although this environment is not the most convenient and familiar to most programmers, it is the most stable of all the choices that could be made and stability is of crucial importance in an endeavor such as nuclear reactors. The reasons for this recommendation are given below.

**Dracula is a Functional Programming Language.**

The boast of a functional program is the fact that it treats computation as the evaluation of a mathematical function. This means that it will be easier to guarantee the results of a function. A functional language avoids using state in its calculations, in an imperative or object oriented paradigm, the program relies heavily on the use of state changes. An example of this capability is the use of variables in the language. In an environment such as Java, a program might be allowed to have such a variable. Let us assume that a programmer declares a variable x that is used to save the amount of fuel in a tank, Let us also assume that the amount of fuel variable is allowed to be manipulated by two controls, if the two controls try to modify the variable at the exact same time, then there is no guarantee what the result of such an event would be. In an environment such as Dracula, this would not happen because there is no variable state being saved. This is a highly generalized explanation but it serves to demonstrate the value of choosing a functional programming environment. This side effects behavior is called referential transparency which says that given the same arguments at different times, a functional program will return the same results.

**Dracula contains a random data testing environment.**

There is a capability to test functions with random data sizes and other requirements in Dracula. An issue such as guaranteeing that two different functions will return the same result will be sure to arise while developing functions for the control. If we want to guarantee a property such as this in an environment such as C++, there would not be a functionality to do this. Let us assume that we have written a function to manipulate a random size database, we can test that the function will be successful for a range of sizes, we can test the success of the function for a database of size between zero and one hundred thousand and Dracula will be able to test this requirement.

**Most up to date Functional programming language.**

Although there are other functional programming languages in use today, the choice of Dracula is recommended because it is a highly well-known language according to prebuilt functions database. Although it will be highly critical to try and build all the functions from no previously written functions, this undertaking might prove to be unreasonable as the project progresses. Choosing an environment with a high quantity of prebuilt functions will allow the programmers to start using prebuilt functions if required.

In Conclusion, the recommendation of the research is to use Dracula as the programming environment for the project, this recommendation is strengthened by the fact that almost all other critical systems utilize functional programming.