Programming Languages

Project 4: Paper Critique

Dmitriy Bespalov

The author of the paper presents an approach to compare seven programming languages using statistical methods. Prechelt uses several implementations (80 total) of the same problem written by different people in C, C++, Java, Perl, Python, Rexx and Tcl programming languages. The programs read a dictionary of words and telephone numbers from two files, and then "translate" the phone numbers into words. In the experiments, these programs are executed and execution times (total, dictionary creation and search) and memory usage are recorded. The author also provides data for length of the programs (LOCs) as well as working time (time took to write each program) and productivity information.

The choice of programming languages is good, it includes the most popular languages that are used nowadays. Certainly, in order for the experiments to be feasible, constraints had to be introduced. The author chose to only use 80 different implementations of only one particular problem. The assumption that the author makes is that most of the programmers that wrote the code have sufficient experience using particular language, which would allow better utilization of the language' capabilities. Also, the code writers are simply asked to write the implementation of the program, and it is unclear how they should go about it (i.e. stress on efficiency, readability, reliability, compactness, etc). Overall, I believe the limitations are reasonable, although the experiments could definitely use higher number of programs (especially for Rexx, Tcl, C and C++). In addition, using more than one problem for experiments could produce more general results.

As already been mentioned, the choice of the phonecode problem limits information that could be obtained from the experimental results. The problem utilizes several (but by far not all) aspects of the language (file I/O, array manipulation, use of advanced data-structures, etc) in its implementations. While the use of this problem provides us with important information, the use of more problems that employ broader range of aspects of the language (i.e. math computations, GUI, etc) would produce more general picture of each language.

The results of the experiments are presented very clearly and produce clear picture of comparisons. Author provides thorough statistical measures for the experimental data, the charts are clear and easy to understand. In addition, textual explanations are clearly understandable as well. Finally, to improve the presentation of the results, additional data tables can be provided. For instance, a table of ratio values for the run-times for 80 percentile. This could provide additional information of how the different languages compare to each other.

The conclusions that the author provides are somewhat contradicting the data presented throughout the paper. For example, C appears to use as much memory as Perl, Python or Rexx, while author claims that scripting languages consume twice as much memory. Also, for search phase, author says that Java is slower than C or C++, while from the charts, it appears that C++ is slower than Java. Additionally, runtime experiment should be reconsidered for C++ programming language. I believe, that such slow running time for this language is due to poor implementation practices. The memory consumption data, should also contain how much memory interpreted languages require for the interpreter. Since, mostly this is going to be fixed amount, it would be possible to determine how efficiently each interpreter can store the data.

There is a number of additional experiments that could be performed. First, more problems should be considered. New problems should address mathematical calculations (i.e. Singular Value Decomposition), GUI implementation, networking applications. Each problem should be implemented at least 15 times, and same person should be asked to implement same problem in several languages. I believe that such approach will produce more robust and general comparison information.

Also, it should be made clear for coders what aspect (i.e. efficiency, readability, compactness, etc) is most important. By introducing such discrimination, it would be possible to determine which languages are capable of producing the most efficient code for example. It would also be possible to see how readable the code is. For this experiment, people could be asked to determine what the code is doing (provided that all of the comments are removed), and the time it takes each person to answer this question would be a good numerical estimate for code readability.

Another interesting experiment would be portability experiment for Java. This would require more sophisticated programs that utilize networking code or GUIs. Each program could be executed on different platforms and the time it takes to fix all of the problems could be used as numerical measure.

Finally, data can be collected using datasets of various sizes. This experiment would give us an idea of how efficient each language is when the size of data increases.

The conclusions provided in significant findings appear to correspond to preconceived beliefs. Most of

the data that is provided in the paper also confirm that. Although, Some of the data presented contradicts my previous beliefs. For instance, I was very surprise to see programs written in C++ be slower than some of the programs written in Python or Perl. Also, C memory consumption was not expected to be as large as some of the scripting languages.