1 Literature Survey of Previous Work

There appear to be two kinds of research done to compare programming languages in the literature: feature by feature comparisons and comparisons of small programs.

1.1 Feature Comparisons

Feuer and Gehani take a more academic and conceptual approach comparing C and Pascal.[2] The only code they directly compared was a single function implementing a binary search. They begin with a history of the languages and discuss design decisions, followed by a step by step walk through each language's major features. They also evaluate C and Pascal for different problem domains. This paper is more of an informative overview than a hard empirical evaluation.

In a comparison of Ada 95 and Java, Brosgol takes a similar approach,[1] going feature by feature through the two languages. He provides sample code snippets throughout. He arrives at a table of features, highlighting differences in syntax and programming organization features, memory management, and OO features like inheritance, polymorphism, and encapsulation.

Nami presents a factual comparison of Eiffel, C++, Java, and Smalltalk[4]. He gives a brief introduction to each language, describing design decisions and history. He then classifies each language based on static vs dynamic typing, compiled vs interpreted build methods, built-in quality assurance facilities, automatic documentation facilities, multiple vs single inheritance, and concludes with a brief discussion on each language's efficacy for building infrastructures.

Tang does a similar comparison of Ada and C++ using much of the same methods as the other studies.[6]

1.2 Small Program Comparisons

Perhaps more useful and interesting are those comparisons done by inspecting the same program written in different languages. These give concrete examples of the differences.

The method I follow for this project is closely related to the method used by [5], in which Prechelt compared C, C++, Java, Perl, Python, Rexx, and Tcl by having various computer science masters students and volunteers from newsgroups online write the same small program in one of the languages. He then compared the resulting programs based on program runtime, memory consumption, lines of code, program reliability (based on whether the program crashes or not), the amount of time it took each programmer to write the program, and program structure. The program he had the participants write was a simple string processing program that consisted of converting telephone numbers into sentences based on a large dictionary and mapping scheme. Most of the programs he received were fairly small programs, taking a median of 3.1 hours to write, and averaging around 200-300 line of code.

Some of the more interesting results from Prechelt's study included the observation that in lower level languages, a lot of code was dedicated to writing the data structures, while in the higher level languages, the programmer usually took advantage of the language's existing built-in data structures. He also found that scripts tended to require about twice as much memory as C and C++, with Java taking 3-4 times as much, that C and C++ was about twice as fast as Java and several more times faster than scripting languages.

Prechelt includes a discussion of the validity of his evaluations. He acknowledges the potential problems of asking for self-reported data from the Internet, as well as potential differences in programmer ability and working conditions. He suggests that the large number of people (80) who contributed programs balances out many of these problems, and that the results should not be trusted for small differences. He does assert that large differences are more likely to be accurate.

Henderson and Zorn perform a similar study in [3]. They compare C++, a well known language, with four lesser-known languages: Oberon-2, Modula-3, Sather, and Self. They also write a short program in each of the languages, this time a simple database for university personnel information. These are all Object Oriented languages, and as such, the comparison is weighted specifically towards OO features. They compare each language's methods of implementing and capabilities fr inheritance, dynamic dispatch, code reuse, and information hiding. In addition to OO features, they also compare execution time, lines of code, and compile time. Henderson and Zorn explicitly state that one of the goals of their survey is to increase programmer awareness of lesser-known languages.

In a more informal study, Floyd compares C++, Smalltalk, Eiffel, Sather, Objective-C, Parasol, Beta, Turbo Pascal, C+@, Liana, Ada, and, Drool. He collects an implementation of a linked-list structure from various people and then summarizes the results in a table that compares garbage collection schemes, inheritance (single or multiple), binding time, compilation (compiled vs interpreted), exception handling features, and lines of code. He simply enumerates the implementations and does not do further analysis.

References

[1] Benjamin M. Brosgol. A comparison of the object-oriented features of ada 95 and java. In *TRI-Ada '97: Proceedings of the conference on TRI-Ada '97*, pages 213–229, New York, NY, USA, 1997. ACM.

[2] Alan R. Feuer and Narain H. Gehani. Comparison of the programming languages c and pascal. *ACM Comput. Surv.*, 14(1):73–92, 1982.

[3] Robert Henderson and Benjamin Zorn. A comparison of object-oriented programming in four modern languages. *Softw. Pract. Exper.*, 24(11):1077–1095, 1994.

[4] Mohammad Reza Nami. A comparison of object-oriented languages in software engineering. *SIGSOFT Softw. Eng. Notes*, 33(4):1–5, 2008.

[5] Lutz Prechelt. An empirical comparison of seven programming languages. *Computer*, 33(10):23–29, 2000.

[6] L. S. Tang. A comparison of ada and c++. In *TRI-Ada '92: Proceedings of the conference on TRI-Ada '92*, pages 338–349, New York, NY, USA, 1992. ACM.