A Research Study of Data Collection and Analysis of Semantics of programming Languages

Sajjade Daraqshan Arshiya Zill-E-Ilahi Shaheen[1] , Sajjade Zeba Shahezeab[2], Dr.Abhijeet Kaiwade[3], Sajjada Gulfishan Tamkanat[4]

[1]Research Scholar, JJT University, Rajasthan, Associate Software Developer, ISRO Satellite Center [ISCA], Bangalore.
[2] Professor, Department Of Engineering, JSPM College, Pune.
[3] Professor, Department of MCA, Dr. Dypatil College, Pune.

Abstract:*-In this paper we review the literature relating to the data collection and analysis study of programming language. We identify general developments evaluating trainee and specialist programmers, programming awareness and approaches, programmer creations and conceptions, and object oriented versus procedural programming. (We do not cover research relating specially to other programming style). The main focus of the review is on programming trainee and topics relating to data collection and analysis. Various problems practiced trainee are acknowledged, as well as problem raise relating to various programming language which is most useful programming language of entire programming languages, to algorithmic complexity in certain language features, to the "Weakness" of trainees knowledge and so on. We reviews this materializes and give an opinion few practical suggestions for future work. We implicate that the key issue that materialize is difference of powerful and in- powerful trainees.*

1. INTRODUCTION

Programming is very useful skills and can be a rewarding career. In recent years the demands of programming languages for implementing in different software project areas. Introductory programming languages have become increasing popular. Programming course is generally recorded as difficult and often have the heights failures rates. It's generally acetated that is takes about experienced to turn trainees into expert programming languages. The programming language is data designed to attach commands to device or a computer. Programming languages are mostly used to manage presentation of a device or to state algorithms. At an attendance, 1000's of programming languages have been developed. During the computer field, a lot of languages require to be declared in a very important structure as, further programming languages make use of declarative structure. The program shall be categorized into two types like as syntax and semantics. A few languages are explained by SO normal like C language.

Figure 2.1:-Symbolic representation of PL

2. DATA OF DIFFERENT PROGRAMMING LANGUAGES

2.1 C++ Language

The C++/CPP language has been an object oriented formation which is used in big projects.

Programmers be able to work together one program into dissimilar segments or even single case by case work on every element of the program. The formation of entity oriented also allows code to used again a lot of times. This language is an practical language. Other than a lot of programmers will differ C++ language.

2.2 C Language

The C language is fundamental programming language and it is extremely famous language, mainly used in

game programming, since C language contains the extra stuffing of the C++/CPP, each and every programmer uses this language since it programs quicker. The worth of this language awards the reusability of C++ to obtain the light boots in presentation through C language.

2.3 Pascal Language

Pascal language is mainly a teaching language and not many industries use this language to write programs. This language be likely to use keywords in its Pascal symbols and bares in C language. Therefore this language is extremely simple for debs to know then programming language like C and C++/CPP. Borland is a compiler software company, which is using Delphi programming language for industrial power. Delphi is an object oriented language of Pascal, and currently Borland compilers simply make use of it.

2.4 Fortran Language

FORTRAN language is a number chomping language and immobile it is used by scientists. This language permits various sizes of variables awake to the memory maximum value in the device. This language is appropriate for engineers, who have to work out standards through high up correctness. Program in FORTRAN is nonflexible and from time to time its crates not easy to read.

2.5 Java Language

The Java language is multiplatform language that's mainly obliging in networking. Sure frequently this language is used on web by Java applets. Though, this language is used to plan bridge platform programs, because it same to C++ in formation and grammar. For C++/CPP programmers, Java language is extremely simple to study and it presents a number of benefits makes it's available through object oriented programming. Similar to reusability and it shall be not easy to write well organized code in java. Other than, these days the rate of Java language has greater than before and 1.5 version presents a number high quality features for trouble-free program creating.

2.6 Perl Language:-

Perl language is file management language for Unix. Other than it is more famous for its common gateway interface programming (CGI). It is a idiom for programs so as web servers shall represent to permit additional abilities of web pages. Pearl language is a technique for presenting text and it is used for useful purposes and additional databases, and it is extremely simple to choose the basic if we have several language over C++ language. Since, the web host shall analysis Perl script files because they are text files, when C++ is compiled.

2.7 PHP Language

In this PHP Booklet The Summit Language, you will find details about some words before you start writing. Sure, you can download PHP Myself with Apache. The PHP program language is a HTML-sponsored list of servers. Let's specify this sentence.

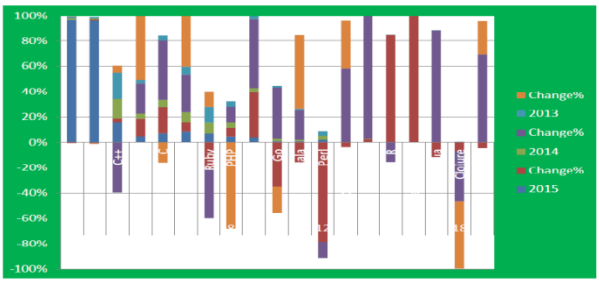
2.8 Scheme Language

The scheme language is a exchange of LISP language. Whichever project beneath the scheme language be

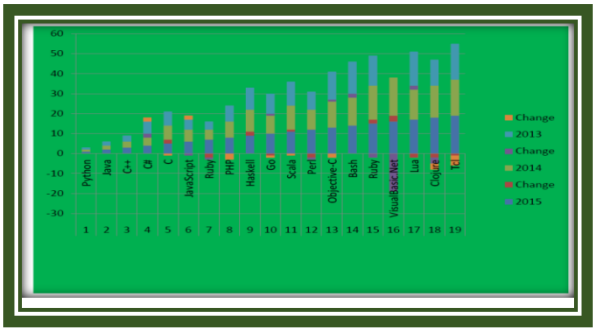
going to result in the reimplementation of the majority of the LISP language. This is all concerning the dissimilarities flanked by programming languages and some main programming languages are talk about. Along with, the left over languages such as Tcl, Python, Smalltalk, COBOL, C# and prolog are same as the above languages which are explained. Other than choosing the appropriate language for increasing a program or application is extremely important Table 2.1 Programming Language Year-wise Usage Percentage Changes.

2015 Rank	Languages	2015	Change%	2014	Change%	2013	Change%
1	Python	26.67	-14.64%	31.24%	3.10%	30.30%	5.21%
2	Java	22.58	15.37%	19.57%	-11.85%	22.20%	-13.95%
3	C++	9.96%	1.76%	9.79%	-24.70%	13.00%	3.17%
4	C#	9.39%	27.37%	7.37%	47.37%	5.00%	100.00%
5	C	7.37%	21.37%	6.07%	48.14%	4.10%	-16.33%
6	JavaScript	6.88%	6.09%	6.48%	24.66%	5.20%	33.33%
7	Ruby	5.88%	-17.27%	7.11%	-32.90%	10.60%	10.42%
8	PHP	3.82%	5.45%	3.62%	9.84%	3.30%	-54.79%
9	Haskell	1.77%	17.24%	1.51%	25.83%	1.20%	
10	Go	1.27%	-44.00%	2.26%	50.67%	1.50%	-25.00%
11	Scala	1.04%	-17.80%	1.27%	27.00%	1.00%	66.67%
12	Perl	0.95%	-37.33%	1.52%	-6.17%	1.62%	
13	Objective-C	0.82%	-17.62%	1.00%	265.76%	0.27%	173.40%
14	Bash	0.46%	7.21%	0.43%	290.91%	0.11%	
15	R	0.37%	165.71%	0.14%	-30.00%	0.20%	
16	VisualBasic.Net	0.37%	825.50%	0.04%			
17	Lua	0.19%	-44.51%	0.35%	337.50%	0.08%	
18	Clojure	0.14%	-8.53%	0.15%	-48.28%	0.29%	-63.75%
19	Tcl	0.06%	-8.57%	0.07%	133.33%	0.03%	50.00%

Table 5.1: Representation of Programming language



Hence, the above programming language are known to be the best ones of 2015. So the developer and programmers should ensure that they're updated regarding them. Knowing such programming languages will certainly take them to a greater level altogether in their career !



Graph5.1: Programming Language Year-Wise Rank changes,

2015 Rank	Languages	2015	Change	2014	Change	2013	Change
1	Python	1	0	1	0	1	0
2	Java	2	0	2	0	2	0
3	C++	3	0	3	0	3	0
4	C#	4	0	4	2	6	2
5	C	5	2	7	0	7	-1
6	JavaScript	6	0	6	-1	5	2
7	Ruby	7	-2	5	-1	4	0
8	PHP	8	0	8	0	8	-3
9	Haskell	9	2	11	0	11	
10	Go	10	-1	9	1	10	-1
11	Scala	11	1	12	0	12	-1
12	Perl	12	-2	10	-1	9	
13	Objective-C	13	0	13	1	14	-2
14	Bash	14	0	14	2	16	
15	Ruby	15	2	17	-2	15	
16	VisualBasic.Net	16	3	19	-19		
17	Lua	17	-2	15	2	17	
18	Clojure	18	-2	16	-3	13	-3
19	Tcl	19	-1	18	0	18	-5

Table 5.2: Representation of PL

That's one of the most popular questions for anyone who is getting started in data science. You have several programming languages to begin with. When I got too started in the field of data science, I had the same confusion as you, and I wasted several hours browsing to come up with a good choice.

2.9 Aspects of Python

By surfing the internet, I was able to find that python was easier to learn than R. R on the other hand had a high learning curve when compared to the python. This reminded me the quote by Mark Zuckerberg:

"If you do the things that are easier first, then you can actually make a lot of progress". So you can understand the code written by other developers without pain.

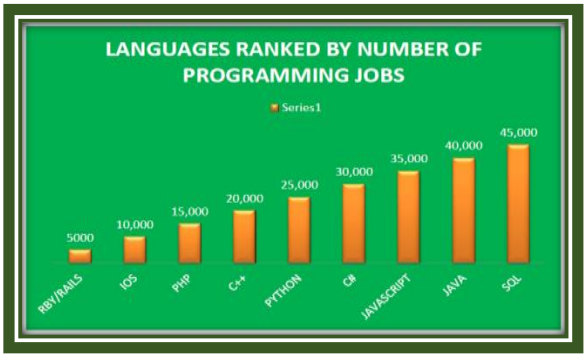
The first code example below is written in C++:

```
#include stdout
Int main()
{
        Std::cout<<"Hello,World!n";
}
```

Here is the code with the same output in python:
Print("Hello, World")

3. FINAL THOUHHTS

R is the standard language for performing statistical analysis, it has quite a high learning curve and there are certain areas of data science for which it is not well suited. Python is an extremely coherent, compact, object oriented language while R is frankly a jumble of features, which makes it intimidating for beginner.



Graph5.2:Most In Demand Programming Language

4. THE ROLE OF LANGUAGE SEMANTICS

Programming language semantics has lost touch with large groups of potential users [39]. Among the reasons for this unfortunate state of affairs, one stands out. Semantic results are rarely incorporated in practical systems that would help language designers to implement and test a language under development, or assist programmers in answering their questions about the meaning of some language feature not properly documented in the language's reference manual. The current situation in which semantics, languages, and tools are drifting steadily further apart is shown in Figure 1. The tool-oriented approach to semantics aims at making semantics definitions more useful and productive by generating as many language-based tools from them as possible. This will, we expect, reverse the current trend as shown in Figure 2. The goal is to produce semantically well-founded languages and tools. Ultimately, we envision the emergence of "Language Design Assistants" incorporating substantial amounts of semantic knowledge. Table 1 lists the semantics definition methods we are aware of. Examples of their use can be found in [40]. Petri nets, process algebras, and other methods that do not specifically address the semantics of programming languages, are not included. Dating back to the sixties, attribute grammars and denotational semantics are among the oldest methods, while abstract state machines (formerly called evolving algebras), coalgebra semantics, and program algebra are the latest additions to the field. Ironically, while attribute grammars are popular with tool builders, semanticists do not consider them a particularly interesting definition method. Table 2 lists a representative language development system (if any) for the semantics definition methods of Table 1. The last entry, Software Refinery, which has its origins in knowledge-based software

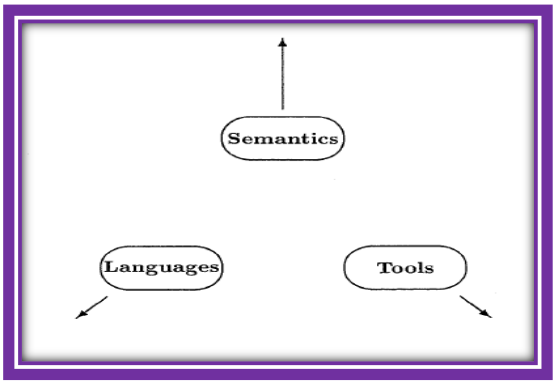


Figure 2.3:Semantics ,Languages

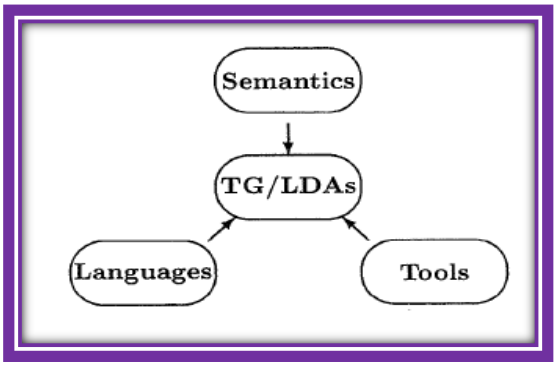


Figure 2.4:Tool Oriented approach

Semantics	Definition in terms of
Axiomatic [4]	Pre- and postconditions
Attribute grammars [12]	Attribute propagation rules
Denotational [38]	Lambda-expressions
Algebraic [7]	Equations/rewrite rules
Structural operational [35]/ Natural [23]	Inference rules
Action [31]	Action expressions
Abstract state machines [19]	Transition rules
Coalgebraic [21]	Behavioral specification rules
Program algebra [8]	Equations

Table 5.3: Current approaches to PL semantics

Environments research at Kestrel Institute does not fit any of the current semantics paradigms. The pioneering Seminole system [2] is, to the best of our knowledge, no longer in use and is not included. The systems listed have widely different capabilities and are in widely different stages of development. Before discussing their characteristics and applications in Section 3, we first explain the general ideas underlying the tool-oriented approach to programming language semantics. These were shaped by our experiences with the ASF+SDF Meta-

Environment (Table 2) over the past ten years. Finally, we discuss Language Design Assistants in Section 4.

5. A TOOL -ORIENTED APPRROACH TO SEMANTICS

The tool-oriented approach to semantics aims at making semantics definitions more useful and productive by generating as many language-based tools from them as possible. This affects many aspects of the way programming languages

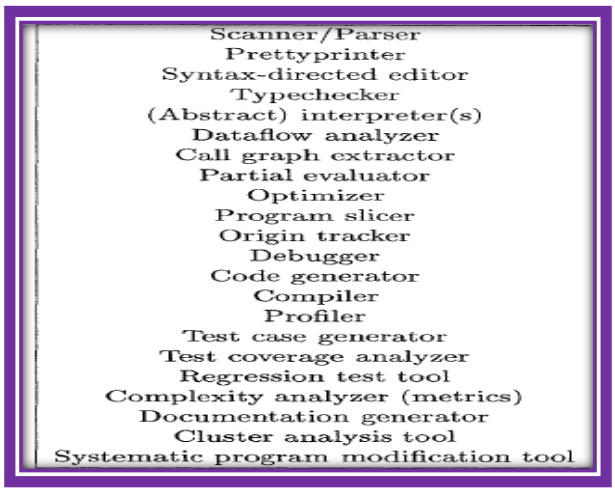


Table 5.5: language definition

Semantics is practiced and upsets some of its dogmas. Table 3 lists some of the tools that might be generated. In principle, the language definition has to be augmented with suitable tool-specific information for each tool to be generated, and this may require tool-specific language extensions to the core semantics definition formalism. In practice, this is not always necessary since semantics definitions tend to contain a good deal of implicit information that may be extracted and used for tool generation.

The first entry of Table 3, scanner and parser generation is standard technology. The key features of the tool-oriented approach are:

System	Generated tools	Semantic engine
Synthesizer Generator	scanner/parser (LALR), prettyprinter, syntax-directed editor, incremental typechecker, incremental translator, …	incremental attribute evaluator
PSG	scanner/parser, syntax-directed editor, incremental typechecker (even for incomplete program fragments), interpreter	functional language interpreter
ASF+SDF Meta-Environment	scanner/parser (generalized LR), prettyprinter, syntax-directed editor, typechecker, interpreter, origin tracker, translator, renovation tools, …	conditional rewrite rule engine
Centaur	scanner/parser (LALR), prettyprinter, syntax-directed editor, typechecker, interpreter, origin tracker, translator, …	inference rule engine
ASD	scanner/parser, syntax-directed editor, checker, interpreter	conditional rewrite rule engine
Gem-Mex	scanner/parser, typechecker, interpreter, debugger	transition rule engine
Software Refinery	scanner/parser (LALR), prettyprinter, syntax-directed editor, object-oriented parse tree repository (including dataflow relations), Y2K/Euro tools, program slicer, …	tree manipulation engine

Table 5.6 : Result analysis

6. EXSTING LANGUAGE DEVELOPMENT SYSTEMS

Table 4 summarizes the tool generation capabilities of the representative language development systems listed in Table 2. All of them can generate lexical scanners, parsers, and pretty printers, many of them can produce syntax-directed editors, type checkers, and interpreters, and a few can produce various kinds of software renovation tools. To this end, they support one or more specification formalisms, but these differ in generality and application domain.

7 CONCLUSIONS

Repeat user? , And + they are also explicitly described, such as harsh drivers who can use a taxi. Non-proud support has been restored by two syntax [18] drivers, and with the off-stage! E, try to estimate that there is no commitment, failure, and e-mail alerts, unsuccessful with e-mail, tools used by majeure pike And like Vatic E, but if E is happening, they are the same, and not. It does not contain text content.

REFERENCES

- Juwana, B. J. C. Perera, N. Muttil(2010),"Water Science and Technology", IEEE, Vol. 62 ,PP 7-10.
- David M. Hansen (2005),"Assessing Programming Errors Attributable to Type less Programming Languages ",A Research Grant Proposal,Vol.12,PP.3-10.
- John K. Ousterhou (1998),"Scripting: Higher-Level Programming for the 21st Century", IEEE, Vol.45, PP.34-40.

- Stewart C. Baker and Taylor Francis (2014),"Making It Work for Everyone: HTML5 and CSS Level 3 for Responsive", Accessible in Journal of Library and Information Services in Distance Learning, Vol.66, PP.23-45.
- ANDREAS, J., VLACHOS, A., AND CLARK, S. Semantic parsing as machine translation. The Association for Computer Linguistics, PP. 47–52.
- BANEA, C., MIHALCEA, R., WIEBE, J., AND HASSAN, S. Multilingual subjectivity analysis using machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, PP. 127–135.
- CER, D., GALLEY, M., JURAFSKY, D., AND MANNING, C. D. Phrasal: A statistical machine translation toolkit for exploring new model features. In Proceedings of the NAACL HLT 2010 Demonstration Session, Association for Computational Linguistics, PP. 9–12.
- HINDLE, A., BARR, E. T., SU, Z., GABEL, M., AND DEVANBU, P. On the naturalness of software. In ICSE 2012 (2012).
- HOPCROFT, J. E., AND ULLMAN, J. D. Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 1979.
- KOEHN, P. Statistical Machine Translation, 1st ed. Cambridge University Press, New York, NY, USA, 2010.