North South University

(Department of Electrical and Computer Engineering)

Semester: Fall 2019

Course Title: Concepts of Programming Language

Section: 01

Lecture Topic: Efficient Language Design Criteria

Lecture no.: 06

Prepared By-

Roksana Parvin

1620282042

Md. Jafar Sadik

1620660042

Under the guidance of-

Dr. Md. Shahriar Karim

Assistant Professor, Department of ECE, North South University.

Efficient Language Design Criteria

Reasons Behind the Success:

- **Availability:** For instance, Python is freely available
- **Price:** For instance, Python is free and used extensively where Matlab requires subscription and used by limited people.
- **Politics:** Politics also have some weightage in success of anything.
- **Timing:** For instance, C succeed because of its timing. It was developed right after the UNIX OS to support the utilities of it.
- **Markets:** For instance, Ada succeed because of its use in US military.

Some Programming Languages and Their Success Reasons:

- **Ada Language:** This language is required in several US defense department projects and hence, remained popular.
- **COBOL Language:** Scientific community didn't use it extensively. But it had a great use in business industries. Although, now-a-days we have so many fancy languages, it is still in the market because of the large number of old applications that are maintained.
- **C Language:** The one of the important reasons for the success of C is the timing, it has a very close relation with the success of UNIX Operating System, released in 1971. The C language was developed between 1972 and 1973 to make utilities running on UNIX.
- **Java and Python:** Unlike MATLAB and other commercial platforms such as, these are distributed free with many interesting built-in libraries and features. The growth of internet made these functionalities available for all.

Reasons for Success of Programming Languages:

- Some argue that, language designed by an individual, or small group of individuals, has a better chance of becoming successful language. Examples: Pascal, C, C++, LISP etc.

However, there are other languages that are designed by committees and still are successful. For instance, COBOL, Algol, Ada.

Common Goal for a Language designer:

- Language design should have an overall goal. That is, designers must keep the goal in mind during the overall design process.
- Designers have to think about, whether the language will be used for a particular purpose or will be used for a general purpose. Some Particular-Purpose languages are

- Database Languages
- Graphics Languages
- Real-Time Languages

Some General-Purpose Languages are

- FORTRAN is a general-purpose, compiled imperative programming language that is especially suited to numeric computation and scientific computing, where designers focused on efficient execution. It was developed by John Backus from IBM.
- COBOL, where designers focused on the readability. It is a compiled English-like computer programming language designed for business use.
- ALGOL 60, here designers provided a block structured language for describing algorithms. It was the first programming language to introduce block structure in programming.

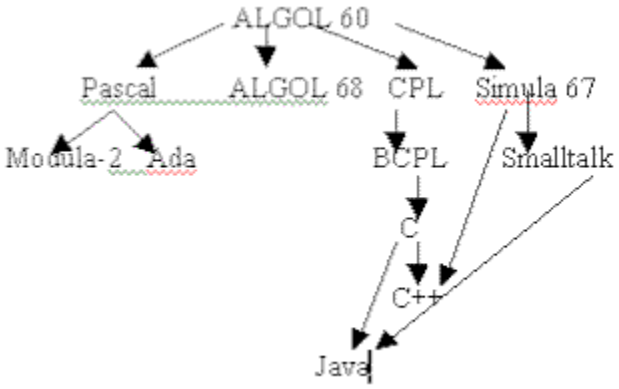

Figure: Block structure history in programming language

An example of block structure in ALGOL 60-

```
if x=3 then
    begin
    y:=9;  k:=10
    end;
```

- C++, designers focused on users' need for greater abstraction while preserving efficiency and capability with C. With C++ they brought up the Object-Oriented features for the programmers.

A Programming Language Efficiency:

A Programming Language has two types of efficiency

- Execution Time Efficiency
- Programmer Efficiency

Execution Time Efficiency:

A measure of amount of time for an algorithm to execute. This is the earliest of design criteria, because of small memories and slow execution. FORTRAN had (has) statements that resembled machine instructions.

Let's consider the two examples given below. The first pseudocode is written in Java and the second one is in Python.

1)

```
int lecture= 6;
String topic= "Language Design Criteria";
```

2)

```
lecture= 6;
topic= "Language Design Criteria";
```

In example one the Data-Types of "lecture" and "topic" are defined. So as soon as the user hit run the variable types are known at the compile time. The compiler can guarantee that only integer and string operations will be performed on variables "lecture" and "topic".

In the second example the Data-Types are not defined. So, as the type specification is not done, it forces runtime system to check the type of a variable's value before executing any operations on it.

Programmer Efficiency:

How quickly and easily can programs be written in the language. That is, how easy it is for a programmer to map his idea into his program. The less struggle a programmer needs to do to implement an idea in a language the more programmer efficient a language is.

The criteria for a programming language to be programming efficient is-

- Debugging is easier.

- How easy it is to map your ideas to exact codes
- How easy it is to express complex process and structures.
- Easy syntax that's mean easy learning.
- Conciseness of the syntax is also a part of programming efficiency. Language that require a complex syntax are often less efficient in programmer point of view. Python is an ideal language for programmer efficiency because its syntax is extremely concise.

A comparison of conciseness of syntax for a same Program in two different languages:

In C:	In Python:

int x=10;
if(x>0)
{
printf("the number is positive");
}
else if(x==0)
{
printf("The number is zero");
}
else
{
printf("The number is negative");
}
	if x>0:
	print("The number is positive");
	elif x = 0:
	print("The number is zero");
	else:
	print("The number is negative");

Here in C, while programming a programmer must have to put semicolon at the end of every line and in each block, he or she has to maintain proper curly braces structure.
In Python on the other hand, the programmer doesn't need to put semicolon at the end of every line and while declaring the conditional statement he or she doesn't have to maintain the curly braces structure, a colon after that would be enough to do the task.

Criteria in a good language design:

Readability:

Readability, means that the code is easy to follow, logically. Standards of indentation and formatting are followed, so that the code and its structure are clearly visible. Variables are named meaningfully, so that they communicate intent. Because ease of maintenance is determined in large part by the readability of programs, readability became an important measure of the quality of programs and programming languages. The result is a crossover from focus on machine orientation to focus on human orientation.

- The quality of a language that enables the reader (even non-programmers) to understand the nature of the computation or algorithm.
- Because of the ease of maintenance is determined in large part by the readability of programs, readability became an important measure of the quality of programs and programming languages. The result is a crossover from focus on machine orientation to focus on human orientation.
- The most important criterion “ease of use”.
- Overall simplicity “Strongly affects readability”.

Write-ability:

Writability is a measure of coding efficiency. It is the ease at which a coder can learn the language in order to write code and how long it takes to write code once you know the language. Ease at which a language can be used to create programs for a specific problem domain.

- This is the quality of expressivity in a language. Writability should be clear, concise, quick and correct.
- It is a measure of how easily a language can be used to create programs for a chosen problem domain.
- Most of the language characteristics that affect readability also affect writability.
- Simplicity and orthogonality
    - A smaller number of primitive constructs and a consistent set of rules for combining them is much better than simply having a large number of primitives.
    - Expressivity
    - It means that a language has relatively convenient, rather than cumbersome, ways of specifying computations.
      Example-1: ++count ⇔ count = count + 1 // more convenient and shorter.

Reliability:

A reliable program performs to its specifications under all conditions.

- A program is said to be reliable if it performs to its specifications under all conditions.
- Assures a program will not behave in unexpected or disastrous ways.
- Both readability and writability influence reliability.
- Type checking– Testing for type errors in a given program. For example, can have errors if a function that is expecting an integer receives a float instead
- Exception handling– Used in Ada C++ and Java, but not in C and Fortran. For example, the try & catch blocks of C++ catch run-time errors, fix the problem, and then continue the program without an “abnormal end’’.

Maintainability:

Maintainability is about writing code in a way which is easy to add new features, modify existing features or fix issues with a minimum effort without the risk of affecting other related modules and functionalities. Obviously software needs new features and bug fixes.

- The ease of which errors can be found and corrected and new features added. Or, in other words, how easily one can change or modify a code?
- **The higher the maintainability is, the less time one need to make the change.**

Regularity:

A set of objects is said to be regular with respect to somecondition if, and only if, the condition is applicable to eachelement of the set. The basic concepts of the language shouldbe applied consistently and universally.

- It increases the maintainability.
- Is a measure of how well a language integrates its features, so that there are no unusual restrictions, interactions, or behavior.
  Regularity is divided into three more definite concepts-
  - Uniformity
  - Generality
  - Orthogonality

Uniformity:

Similar things look similar, different things look different.

- The consistency of appearance and behavior of language constructs.
  Example:
  In C++:

```
class  xyz
      {
                  };
    Int fn ()
    {
                  } //no semicolon
```

  Here we see there is a semicolon at the end of the block when we are working with the class, but there is no semicolon at the end of the block when working with the function in C++

This occurrence shows the lack of uniformity in C++.

Generality: Applicability to a wide range of applications. The avoidance of special cases and generalizing related constructs into one construct.

- Combining closely related constructs into a single more general one.
- Too much generality is bad.
- Ada has 1 loop structure with variations -- a good example of generality

  loop ... end loop

  for i=1..n loop ... end loop

  while cond loop ... end loop

Example: 1) Nested function is not possible in "C".

2) "==" is generally use compare equality.

3) We cannot compare two Array's with

"==" this equality operator.

Orthogonality:

Used to mean that any language element can be used in all reasonable contents. For instance, In C, values of all datatypes can be returned from a function, except array types. Criteria for a language to be orthogonal-

- Independent functions should be controlled by independent mechanisms.
- Language constructs should NOT behave differently in different contexts.
- The language constructs can be combined in a meaningful way.
- Non-uniformity and non-orthogonality may be very closely related in some instances.

Simplicity:

A programming language should be kept simple. The simpler the programming language is, the easier for a programmer to understand. But too much simplicity is also bad when it comes to features' limitations.

- Simplicity was a reason behind the Pascal's success
- BASIC was simple but lacked declarations and blocks
- Java tried to simplify C++

References:

http://jcsites.juniata.edu/faculty/rhodes/lt/plcriteria.htm

https://courses.cs.washington.edu/courses/cse505/99au/imperative/algol.html

http://jcsites.juniata.edu/faculty/rhodes/lt/blockstr.htm

TextBook:

CONCEPTS OF PROGRAMMING LANGUAGES, 10th Edition
(ROBERT W. SEBESTA)