An Approach To Bridge Between Different Software Development Models, Application Areas And Different computer languages

Bipasha Mallick
Assistant Professor,
Haldia Institute Of Technology
bipasm@gmail.com

Dipankar Pramanik
CSE,AIET
prdipu@gmail.com

Abstract. This article related to the software development [1] in term of software design models, application areas and computer languages .It is concerned with the software management processes that examine the area of software development through the development models, which are known as software development life cycle. Programmers are chose the programming language [2] and software development model according to the client requirement and the area where the software will be used. Here this article also shows the relation among the software development models, application areas and the computer programming languages.

Keywords: *Software Management Processes, Software Development, Development Models, Application Area, Programming Languages.*

1. Introduction

During the present time No one can deny the importance of computer in our life. In fact, computer has become indispensible in today's life as it is used in many fields of life such as industry, medicine, commerce, education and even agriculture. Computer is considered a time-saving device and its progress helps in executing complex, long, repeated processes in a very short time with a high speed. In addition to using computer for work, people use it for fun and entertainment. Noticeably, the number of companies that produce software programs for the purpose of facilitating works of offices, administrations, banks, etc, has increased recently which results in the difficulty of enumerating such companies. For that they are developing the software using modern computer languages. Presently there are near about thousand of computer languages, so on which they should choose one? The answer of this question related to three things. Those are the application areas and purpose of the software, software development models and tools, and the computer language.

2. Software development Models

A software process model [4] is an abstract representation of a process. It presents a description of a process from some particular perspective as:
1. Specification.
2. Design.
3. Validation.
4. Evolution.

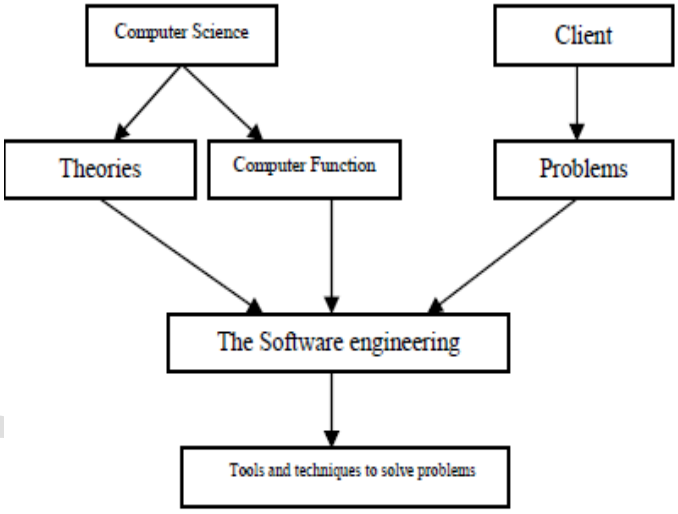

Fig. 1 Explanation of software engineering conception.

General Software Process Models are
1. Waterfall model: Separate and distinct phases of specification and development.
2. Prototype model.
3. Rapid application development model (RAD).
4. Evolutionary development: Specification, development and validation are interleaved.
5. Incremental model.
6. Iterative model.
7. Spiral model.
8. Component-based software engineering: The system is assembled from existing components.

There are many variants of these models e.g. formal development where a waterfall-like process is used, but the specification is formal that is refined through several stages to an implementable design.

Before develop software, have to details requirement and the application area from the client. According to the requirement we have to choose the software development model and tools and design the software architecture. So for choosing the programming language development models take a vital role.

3. Software Application Areas

Software application Area means in which area or for which purpose the software will be used. Now a day's Software are

developed for commercial, business, entertainment, manufacturing, agriculture, scientific experiments and also so many other areas. There have to also consider so many metrics, like user authentication, user friendly, security, easy to access etc. So first we have to decide the application area of software, then have to decide which modern computer language will suits for that area. Suppose we need a software for a nationalize bank. The purpose of the software is transaction. So there we have to develop the software with the high Security programming language.
Some of the software application areas are:

a. Business software
b. Analytics
c. Airline reservations
d. Banking
e. Automatic teller machines
f. Commerce
g. Compilers
h. Communication
i. Computer graphics
j. Cryptography
k. Databases, support almost every field
l. Embedded systems Both software engineers and traditional engineers write software control systems for embedded products.
m. Engineering
n. Computer Aided Design (CAD)
o. File
p. Games
q. Information systems, support almost every field
r. Logistics
s. Manufacturing
t. Music
u. Network Management
v. Networks and Internet
w. Operating systems
x. Robotics
y. Signal processing, encoding and interpreting signals
z. Simulation, supports almost every field.
aa. Sciences
bb. Traffic Control
cc. Training
dd. Visualization, supports almost every field
ee. Voting
ff. World wide web

4. Different modern computer languages

A programming language is a notation for writing programs, which are specifications of a computation or algorithm. Some, but not all, authors restrict the term "programming language" to those languages that can express all possible algorithms.
To determine and document the influence of software engineering research on programming language design requires exploring how modern day language features have assimilated software engineering ideas. However, the symbiotic relationship between these two fields has made the attribution of some influences difficult. Initially, there was no distinction between the software engineering and programming language research communities; they were a single amorphous field until the late 1970s. As the fields began to evolve into distinct communities, researchers published in journals and conferences from both fields and thus, the classification of research results as either software engineering or programming languages is difficult.
A modern computer language is languages whose are currently in use to developed any software, like C, C++, java, Parl, ruby etc.
In the 1950s, the first three modern programming languages whose descendants are still in widespread use today were designed:
Some important languages that were developed:

- **FORTRAN (1955), the "FORmula TRANslator", invented by John Backus et al.;**
- **LISP (1958), the "LISt Processor", invented by John McCarthy et al.;**
- **COBOL (1959), the COmmon Business Oriented Language, created by the Short Range Committee, heavily influenced by Grace Hopper.**
- **1951 - Regional Assembly Language**
- **1952 - Autocode**
- **1954 - IPL (forerunner to LISP)**
- **1955 - FLOW-MATIC (forerunner to COBOL)**
- **1957 - FORTRAN (First compiler)**
- **1957 - COMTRAN (forerunner to COBOL)**
- **1958 - ALGOL 58**
- **1959 - FACT (forerunner to COBOL)**
- **1959 - COBOL**
- **1959 - RPG**
- **1962 - APL**
- **1962 - Simula**
- **1962 - SNOBOL**
- **1963 - CPL (forerunner to C)**
- **1964 - BASIC**
- **1964 - PL/I**
- **1967 - BCPL (forerunner to C)**
- **1968 - Logo**
- **1969 - B (forerunner to C)**
- **1970 - Pascal**
- **1970 - Forth**
- **1972 - C**
- **1972 - Smalltalk**
- **1972 - Prolog**
- **1973 - ML**
- **1975 - Scheme**
- **1978 - SQL (initially only a query language, later extended with programming constructs)**

- **1978 - SQL (initially only a query language, later extended with programming constructs)**
- **1980 - C++ (as C with classes, name changed in July 1983)**
- **1983 - Ada**
- **1984 - Common Lisp**
- **1984 - MATLAB**
- **1985 - Eiffel**
- **1986 - Objective-C**
- **1986 - Erlang**
- **1987 - Perl**
- **1988 - Tcl**
- **1988 - Mathematica**
- **1989 - FL (Backus);**
- **1990 - Haskell**
- **1991 - Python**
- **1991 - Visual Basic**
- **1993 - Ruby**
- **1993 - Lua**
- **1994 - CLOS (part of ANSI Common Lisp)**
- **1995 - Java**
- **1995 - Delphi (Object Pascal)**
- **1995 - JavaScript**
- **1995 - PHP**
- **1996 - WebDNA**
- **1997 - Rebol**
- **1999 - D**
- **2000 - ActionScript**
- **2001 - C#**
- **2001 - Visual Basic .NET**
- **2002 - F#**
- **2003 - Groovy**
- **2003 - Scala**
- **2003 - Factor**
- **2007 - Clojure**
- **2009 - Go**
- **2011 - Dart**

5. 10 popular modern programming languages

1. PHP :
It is an open-source, interpretive, server-side, cross-platform, HTML scripting language, especially well-suited for Web development as it can be embedded into HTML pages.

2. C#:
It is a general-purpose, compiled, object-oriented programming language developed by Microsoft as part of its .NET initiative; it evolved from C and C++.

3. AJAX (Asynchronous JavaScript and XML):
Though technically not a programming language, AJAX uses XHTML or HTML, JavaScript and XML to create interactive Web applications.

4. JavaScript :
Not to be confused with Java, JavaScript is a an object-oriented, scripting programming language that runs in the Web browser on the client side. Its smaller than Java, with a simplified set of commands, easier to code and doesn't have to be compiled.

5. Perl:
Perl is an open-source, cross-platform, server-side interpretive programming language used extensively to process text through CGI programs.

6. C :
It is A standardized, general-purpose programming language, it's one of the most pervasive languages and the basis for several others (such as C++).

7. Ruby and Ruby on Rails:
Ruby is a dynamic, object-oriented, open-source programming language; Ruby on Rails is an open-source Web application framework written in Ruby that closely follows the MVC (Model-View-Controller) architecture.

8. Java :
This is an object-oriented programming language developed by James Gosling and colleagues at Sun Microsystems in the early 1990s.

9. Python :
This is an interpreter, dynamically object-oriented, open-source programming language that utilizes automatic memory management.

10. VB.Net (Visual Basic .Net) :
An object-oriented language implemented on Microsoft's .Net framework. Using these ten modern languages all the client software is developed.

Every language has some specific area of application and purpose. They are used according to the application area and purpose of the software. A language with appropriate constructs and structure, resting on clean abstractions, is instrumental in building artefacts, and mandatory in education. Homemade, artificial complexity has no place in them.

6. Conclusion

When it comes to choose the perfect programming language [3] for develop the software, it is imperative that you understand that there is no perfect programming language. Once you understand this, it is simply a matter of choosing the language that best serves your needs. Before you decide on what language to use, you should consider the following

- system platform
- software application area
- budget for develop the software
- previous experience in programming
- the database you have chosen for your backend

The Operating system you are running on your system is your platform and your choice of OS may play a major part in the language you choose.

7. Future work

In future, I decide to classify all the modern programming languages according to the application area, operating system, development budget etc.

REFERENCES

[1] Ian Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004.
[2] Niklaus Wirth, "A Brief History of Software Engineering", (25.2.2008).
[3]http://en.wikibooks.org/wiki/Web_Development/Choosing_the_right_programming_language, (17.9.2013).
[4] Rlewallen, "Software Development Life Cycle Models", 2005 ,http://codebeter.com.