Programming Paradigms

Jignesh Mangukiya

Jmm.comp@coep.ac.in

(O) 020-25507100

(M) 07875044731

Programming Paradigm

- It is a style or “a way” of programming.
- *A paradigm is a way of doing something, and not a concrete thing (like a language).*
- If a programming language L happens to may a particular programming paradigm P, then we can say “L is a P language”

List of Common Paradigms

- **Imperative** — Control flow is an explicit sequence of commands.
- **Declarative** — Programs state the result you want, not how to get it.
- **Structured** — Programs have clean, goto-free, nested control structures.
- **Procedural** — Imperative programming with procedure calls.
- **Functional** — Computation proceeds by (nested) function calls that avoid any global state.
- **Object-Oriented** — Computation is effected by sending messages to objects; objects have state and behavior.

List of Common Paradigms

- **Event-Driven** — Control flow is determined by asynchronous actions (from humans or sensors).
- **Logic (Rule-based)** — Programmer specifies a set of facts and rules, and an engine infers the answers to questions.
- **Concurrent**

Imperative Programming

- Control flow is an explicit sequence of commands.

```
    result = []
    i = 0
start:
    numPeople = length(people)
    if i >= numPeople goto end
    p = people[i]
    nameLength = length(p.name)
    if nameLength <= 5 goto next
    upperName = toUpper(p.name)
    addToList(result, upperName)
next:
    i = i + 1
    goto start
end:
    return sort(result)
```

Declarative Programming

- Programs state the result you want, not how to get it.
- Control flow in declarative programming is *implicit*: the programmer states only *what* the result should look like, **r**

```
select upper(name)
from people
where length(name) > 5
order by name
```

Declarative Programming

- No loops, no assignments, etc.

- Whatever engine interprets this code is just supposed to go and get the desired information, and can use whatever approach it wants.

Object Oriented Programming

- OOP is based on the sending of messages to objects. Objects respond to messages by performing operations.

```
result = []
for p in people {
    if p.name.length > 5 {
        result.add(p.name.toUpper);
    }
}
return result.sort;
```

Functional Programming

- Control flow is expressed by combining function calls, rather than by assigning values to variables.

```
let(
  f, fun(
    people,
    if(equals(people, emptylist),
        emptylist,
        if(greater(length(name(head(people))), 5),
          append(to_upper(name(head(people))), f(tail(people))),
          f(tail(people))))),
    sort(f(people)))
```

Functional Programming

- There are no commands.
- Code is much shorter, less error-prone, and much easier to prove correct

Logic Programming

- We express computation in exclusively in terms of mathematical logic.
- While the functional paradigm emphasizes the idea of a mathematical function, the logic paradigm focuses on predicate logic, in which the basic concept is a relation.
- Logic languages are useful for expressing problems where it is not obvious what the functions should be.

Logic Programming

- Let us consider now how we can define the brother relation in terms of simpler relations and properties father, mother, and male.

Using the Prolog logic language one can say:

```
brother(X,Y)      /* X is the brother of Y                    */
                  /* if there are two people F and M for which*/
    father(F,X),        /*      F is the father of X          */
    father(F,Y),        /*  and F is the father of Y          */
    mother(M,X),        /*  and M is the mother of X          */
    mother(M,Y),        /*  and M is the mother of Y          */
    male(X).            /*  and X is male                     */
```

Concurrent programming

- Improve performance

- Multiprogramming systems attempt to utilize resources that would otherwise be wasted, by running two or more jobs concurrently.

- Multi-access systems extend this principle, allowing many jobs to be run, each on behalf of a user at an interactive terminal.

Concurrent programming

- Classified as:
- single processor (interleaved execution of concurrent tasks)
- multiprocessor environment

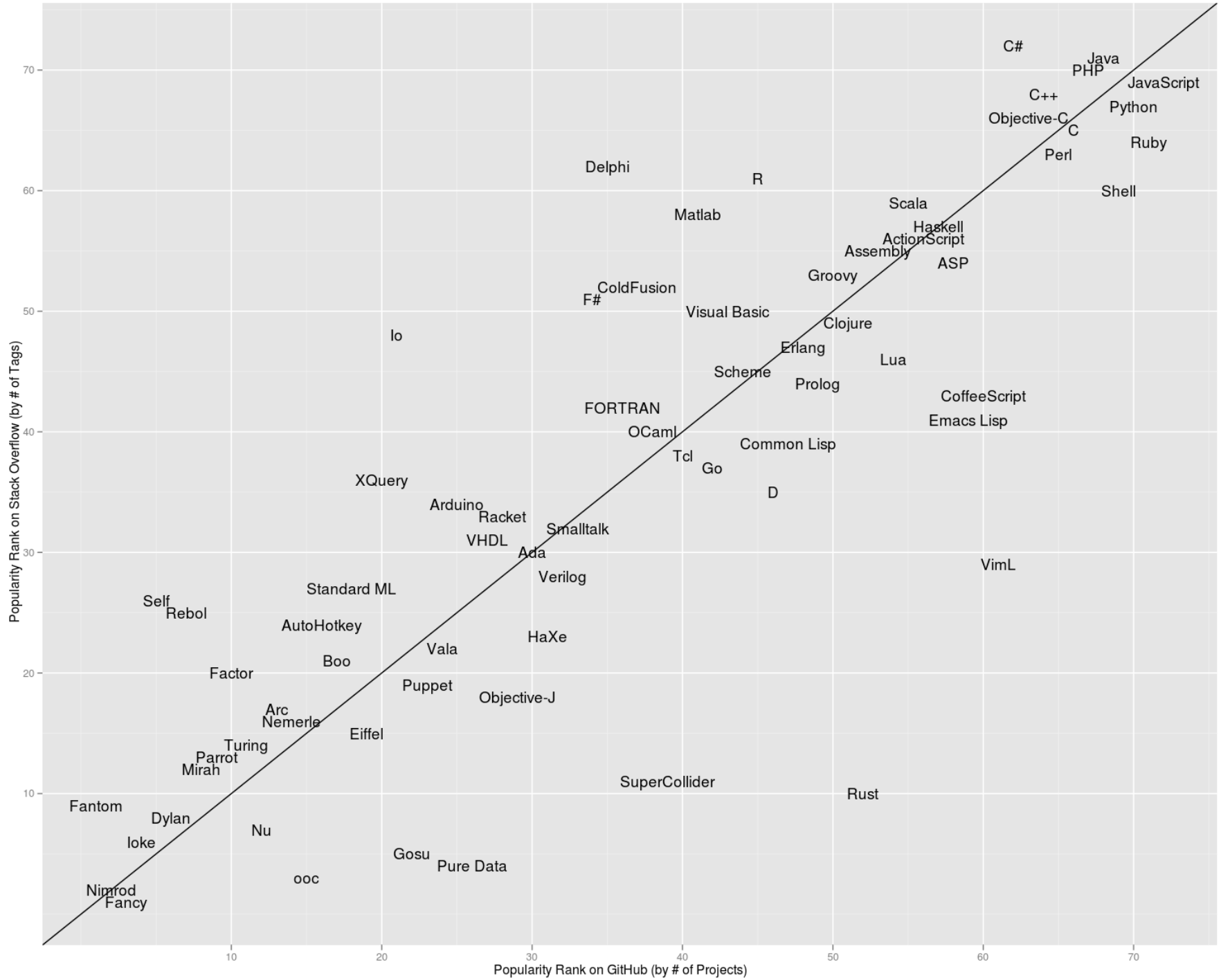
Paradigm	Aspect	
	Corresponding languages	Key features
Structured programming	Imperative programming languages Examples: FORTRAN, COBOL, Pascal, C	• Sequential execution of instructions • "Goto" less programs • Use of variables representing memory locations • Use of assignment to change values of variables • Conditional branch and iterative statements • Recursion is an alternative to iteration
Object-oriented programming	Object oriented programming languages Examples: Smalltalk, SNOBOL, C++, Java	• Object is the basic building block. An object is characterized by state and behavior. The state is specified by the attributes and the behavior is specified by the methods • Encapsulation, polymorphism, and inheritance as the foundational concepts that give an identity to this paradigm
Functional programming	Applicative programming languages Example: LISP	• The basic building block is a function • There is no notion of variable and assignment • Iteration is not supported • Recursion is the key facility
Logic programming	Declarative programming languages Example: PROLOG	• Logic programming is based on symbolic logic • A logic program is a collection of declarations which are true about the desired result. These are called facts • No notion of flow-of-control • A set of rules that operate on the facts are defined • A query reports the results drawn from the facts and governed by the rule base • The inference engine ensures the validity of the results

Event-driven programming	Visual programming languages Examples: Visual Basic, Visual C++	• Programming is based on the set of anticipated events • The base system recognizes the events as they occur and coordinates the necessary responses • This paradigm is very useful in developing a good user interface
Concurrent programming	Parallel programming languages Examples: Concurrent Pascal, ParC, PARLOG, Occam	This paradigm supports multi-threading (segments of the same program can be executed concurrently) and synchronization (facilitates cooperation amongst the several threads)
Distributed programming	Network and internet programming languages Example: Java	Synchronization and Semantics for message passing form the core support for implementing Remote Procedure Call (RPC) or Remote Method Invocation (RMI)
Database programming [4 GLs]	Structured query languages Example: SQL	• This paradigm provides a structured way of framing the query on a RDBMS • It also provides the framework for verifying and validating the query results

Classifications

This is just an example:

Imperative/ Algorithmic	Declarative		Object-Oriented
	Functional Programming	Logic Programming	
Algol Cobol PL/1 Ada C Modula-3	Lisp Haskell ML Miranda APL	Prolog	Smalltalk Simula C++ Java


C#
Java
PHP
JavaScript
C++
Python
Objective-C
C
Ruby
Perl
Shell
Delphi
R
Scala
Matlab
Haskell
ActionScript
Assembly
ASP
Groovy
ColdFusion
F#
Visual Basic
Clojure
Io
Erlang
Lua
Scheme
Prolog
CoffeeScript
FORTRAN
Emacs Lisp
OCaml
Common Lisp
Tcl
Go
XQuery
D
Arduino
Racket
Smalltalk
VHDL
Ada
VimL
Verilog
Standard ML
Self
Rebol
AutoHotkey
HaXe
Vala
Boo
Factor
Puppet
Objective-J
Arc
Nemerle
Eiffel
Turing
Parrot
Mirah
SuperCollider
Rust
Fantom
Dylan
Nu
Ioke
Gosu
Pure Data
ooc
Nimrod
Fancy
10
20
30
40
50
60
70
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)