Programming Languages

How to Reason about it?

Praseed Pai KT

Sr. Solutions Architect
Gadgeon Smart Systems Pvt Limited
Kochi, India

About the Presenter

- A Seasoned Software Engineering Professional with more than twenty five years of Exposure
- Author of Two books on Computer Programming
- Explorer in “Philosophical Tools for Software Engineering” (Has Presented on it, Written one university accredited
- paper, Designed a Pattern based on Advaita Vedanta to transition from OOP to FRP)
- An Expert level professional in Cross Cultural Encounters
- A Critique of Digital Technology Fads (Programmers will be better off , if they stick to Programming. Do not run
- after so called AI/ML, BlockChain etc) - “Plumbing is preferred over Painting!”
- Has Presented in more than three hundred sessions in the past twenty five years
- I also help Programmers eliminate their “Math-Phobia”
- Currently designated as “Sr. Solutions Architect @ Gadgeon”

A Remarkable Book By Ravi Sethi

Programming Languages

2nd edition

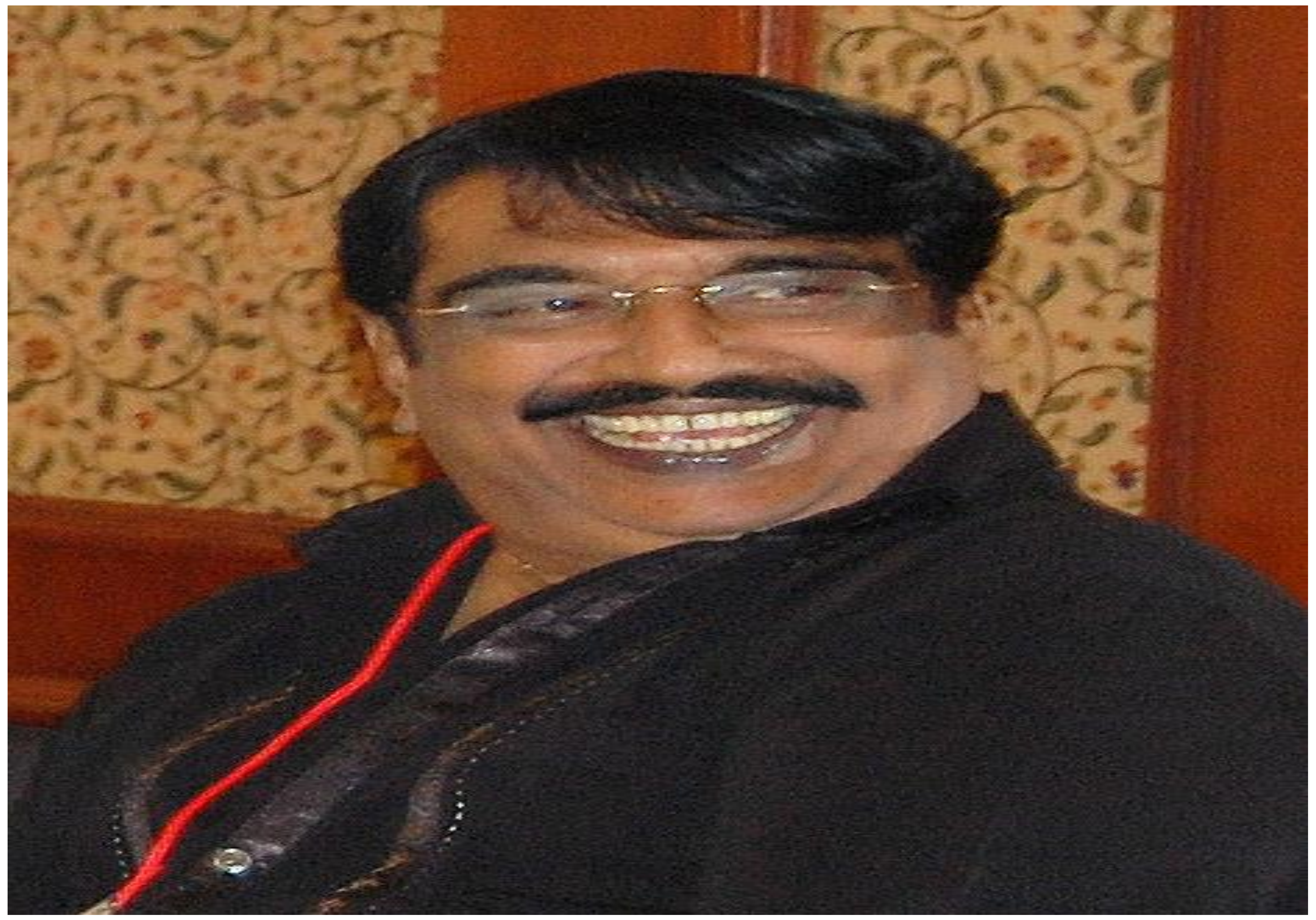
concepts & constructs

Ravi Sethi

PEARSON

Whet your appetite!

Comparative Study of Programming Languages



In India , it is like this. I do not how it is in Punjab? – Punjabi House

An “argument” with a world renowned personality (Neil Gershenfeld)

- Only few people have got ability to assemble circuits which requires precision. Only those guys need to learn from Fablab.
- What If someone wants to learn stuff for understanding it or to take executive decision on a stuff?

……

…….

…….

“ Why should you learn something which is not going to be used”?

- “I would like to learn a stuff to reason about it!

Learn for Reasoning!

Let us Discuss these Three Things

- What is the primary difference between Stored Procedure and a Stored Function in the case of RDBMS Software?
- Have u Come across a Declarative Language in an Imperative Language and Vice Versa ?
- Please name a Reference Type which obeys value semantics in Java and C#?

Programming Languages – A Model driven reality

- By Formal Models – Turing Machine/Lambda Calculus/Predicate Logic based Languages
- By Typing – Static/Dynamic and Type annotation based languages
- By Expressive Power – Turing Complete and Otherwise languages
- By Compilation Strategy – Compiled/Semi-Compiled and Quasi Compiled (Interpreted languages)
- By Programming Paradigms – Procedure/Functional/Logic/Object Procedural/Object Functional
- By Intent specification – Declarative vs Procedural

A Tactical Approach towards Learning Multiple Languages

Hello Script in Python/Java/Ruby
Maintain a Text File of Executable Code snippets

Pros

A Bootstrapping Strategy
A Handy Tool for “Assembling” Code very fast
Please your boss!

Cons

Too much Axiology driven
Cannot exploit deeper synergies between languages

Let us Jump into the water!

A Straight Line Program which accepts a list of numbers as command line arguments and sort them, to print to the Console (Data Parametrization)

- Implementation in Java/C#/Python/Typescript

Improving the above program by Variable Parametrization

- Implementation in Java/C#/Python/TypeScript

Improving the above program by adding Type Parametrization

- Implementation in Java/C#/Python/TypeScript

Improving the above program by adding Behaviour Parametrization

- Implementation in Java/C#/Python/TypeScript

An Unusual story of JavaScript

A Story of JavaScript

- A Programmer was asked by his “boss” (in 1995) to design a language with following attributes
  - Simple
  - Modern
  - To be used by Casual Programmers
  - Embeddable inside a Browser
- A Choice made by that programmer had deep consequences and resulted in “grotesque” language for those times.
- Years later, when processing power improved, that language morphed into a “ubiquitous language”

Simple means “Dynamic Typing”

Bane of newbies is “Static Typing”
Brendon Eich chose Dynamic Typing

```
var a = 2;
b = 2.3
console.log(a/b);
a = "Hello World..";
console.log(a/b);
a = new Date();
b = "ddd";
console.log(a/b);
a = 3.0;
b = 0;
console.log(a/b);
```

Modern means “OOP (in 1995)”

- Three Kinds of OOP
  - Class based OOP
  - Actor based OOP (Event based)
  - Prototype OOP
- Class based OOP is apt for Static Typing
- Actor based OOP might result in “Event Cacophony”
- The only choice available is “Prototype OOP”
- Prototype treats Objects as dictionary
- How to reduce a class to Key/Value dictionary?
- What constitutes a class?
  - Static and Instance variables & Static and Instance methods

```
var a = {};

a.test = 10;


console.log(a.test);


a["test"]=20;

console.log(a.test);
```

How to reduce Object into Value?

- Variables can be mapped into key/value
- Static variables can be mapped into key/value in prototype
- Reference to Static Map is placed inside each Object
- How to reduce Method to Key/Value?
  - We need a paradigm which treats Method/Function as value
  - Functional Programming fits the Bill
  - Support for functional programming was added to support prototype OOP

```
//////////////////////////////////
//
// Clone Temp1 object to create temp2
//
var temp1 = new Person("T",0,1000);
var newobj = {};
for( var n in temp1 )
{
   newobj [n] = temp1[n];
}

temp1.SayHello = function() {
  console.log("I did a dirty trick");

}

newobj ["SayHello"]();
temp1  ["SayHello"]();
```

Questions? If any!