Multi-Paradigm Programming - What is a Programming Paradigm?

Dominic Carr

Atlantic Technological University

dominic.carr@atu.ie

What We Will Cover

1. Goals of this Session

2. Programming Paradigms
   - What is a Programming Paradigm?
   - Abstraction
   - State

3. Sources

Goals of this Session

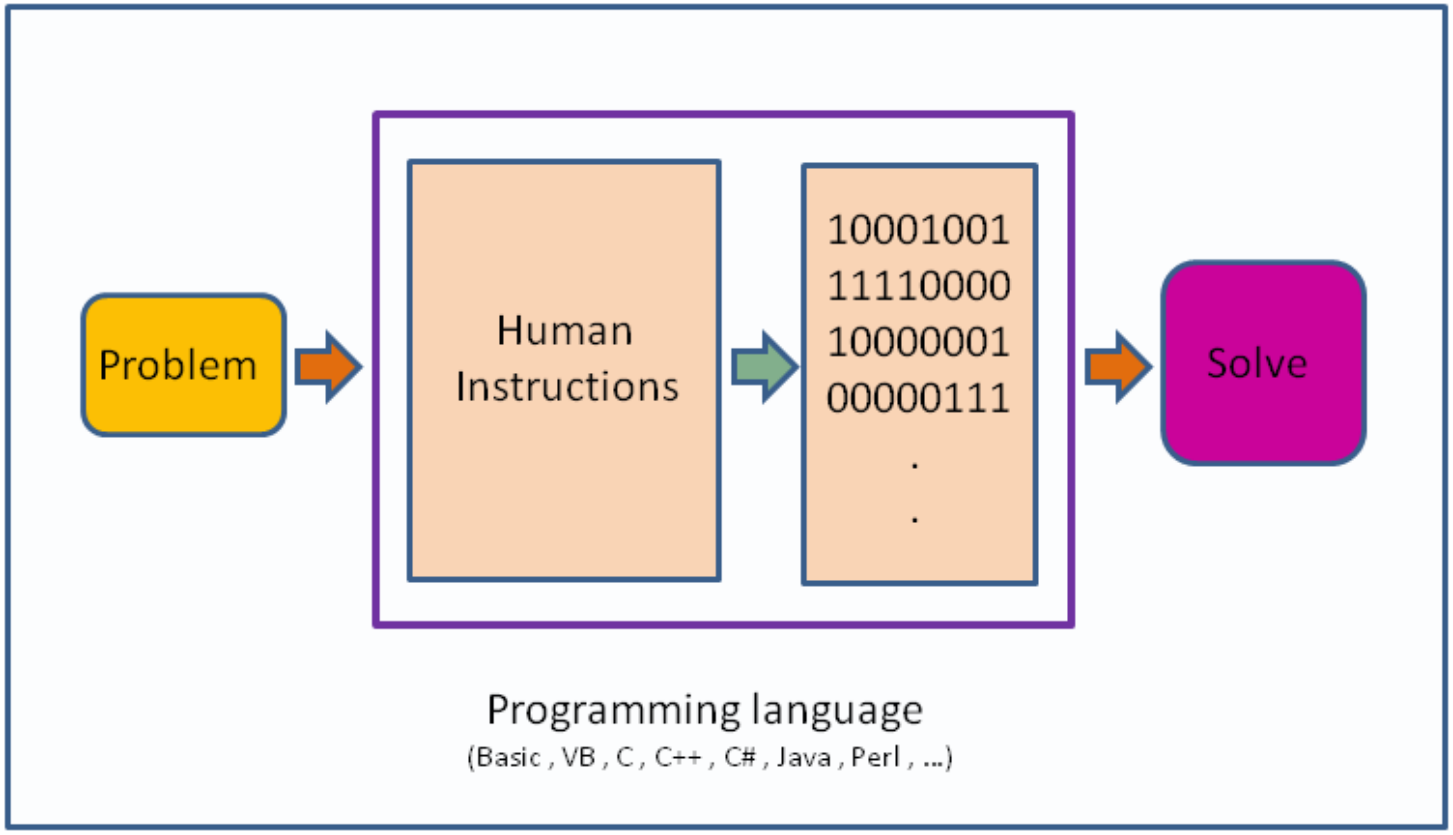
Goals

- To understand....
  - What is a programming paradigm?
  - Why are there different paradigms?
  - How does it relate the notions of State & Abstraction?

Programming Paradigms

Programming

- Programming is a method of communication between an end user and a computer
- *Making the computer do what you want it to do*
- *Does exactly what you* **tell** *it to do*
- Everything in the world of software development is basically a means of making it easier to *convey precise intent to the machine*
  - and to provide structure, to what ultimately becomes a stream of 1s and 0s, for the programmer.
  - and the maintainer, the team, the programmer's future self etc.


Problem
Human
Instructions
10001001
11110000
10000001
00000111
.
.
Solve
Programming language
(Basic , VB , C , C++ , C# , Java , Perl , ...)

Paradigms I

Programming Paradigm

- The term programming paradigm is the style or way of thinking about and approaching problems.
- The chosen paradigm affects how the code is written and structured.
- It can heavily influence how one thinks about the problem being solved
- Some problems map more easily to a particular paradigm
- Each paradigm has it's advocates and detractors, advantages and disadvantages etc.
- A different **language** is not the same thing as a different **paradigm**

Paradigms II

Listing 1: MIPS Assembly[1]

```
LUI  R1, #1
LUI  R2, #2
DADD R3, R1, R2
```

Listing 2: C or Java (In fact this is syntactically valid in a lot of languages)

```
x = 1 + 2;
```

- Both examples show the addition of two numbers
- MIPS is very low level, one step above binary
- C is higher level, though most would consider C to be a low level language
- Both examples follow the same paradigm, such as it can be in a "one liner"

Paradigms III

"if you ever code something that "feels like a hack but it works," just remember that a CPU is literally a rock that we tricked into thinking"

– @daisyowl

Paradigms IV

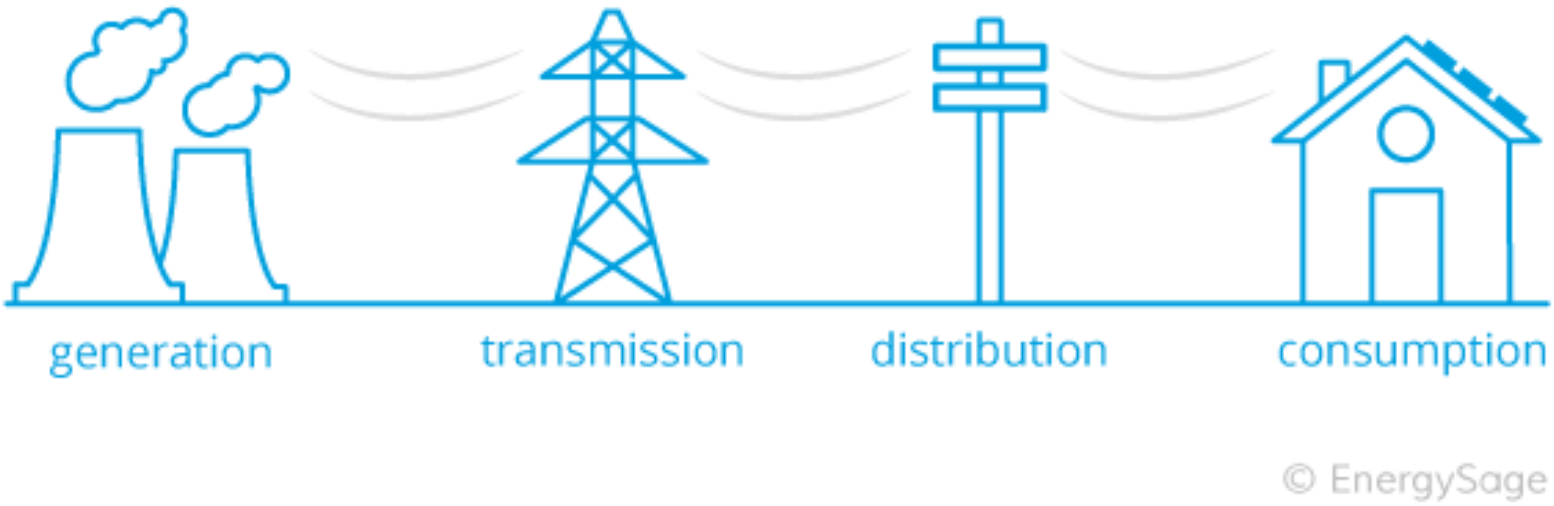
Abstraction

- In software engineering and computer science, abstraction is:
  - the process of removing **physical**, **spatial**, or **temporal details**[2] or attributes in the study of objects or systems in order to focus attention on details of higher importance,[3] it is also very similar in nature to the process of generalization;
  - the creation of abstract concept-objects which are created by mirroring common features or attributes from various non-abstract objects or systems of study[3] — the result of the process of abstraction.
- It is one of the most important concepts in Software Development
  - So much of Computer Science is about abstraction

Paradigms V

Paradigms VI

I don't need to know about....



Paradigms VII

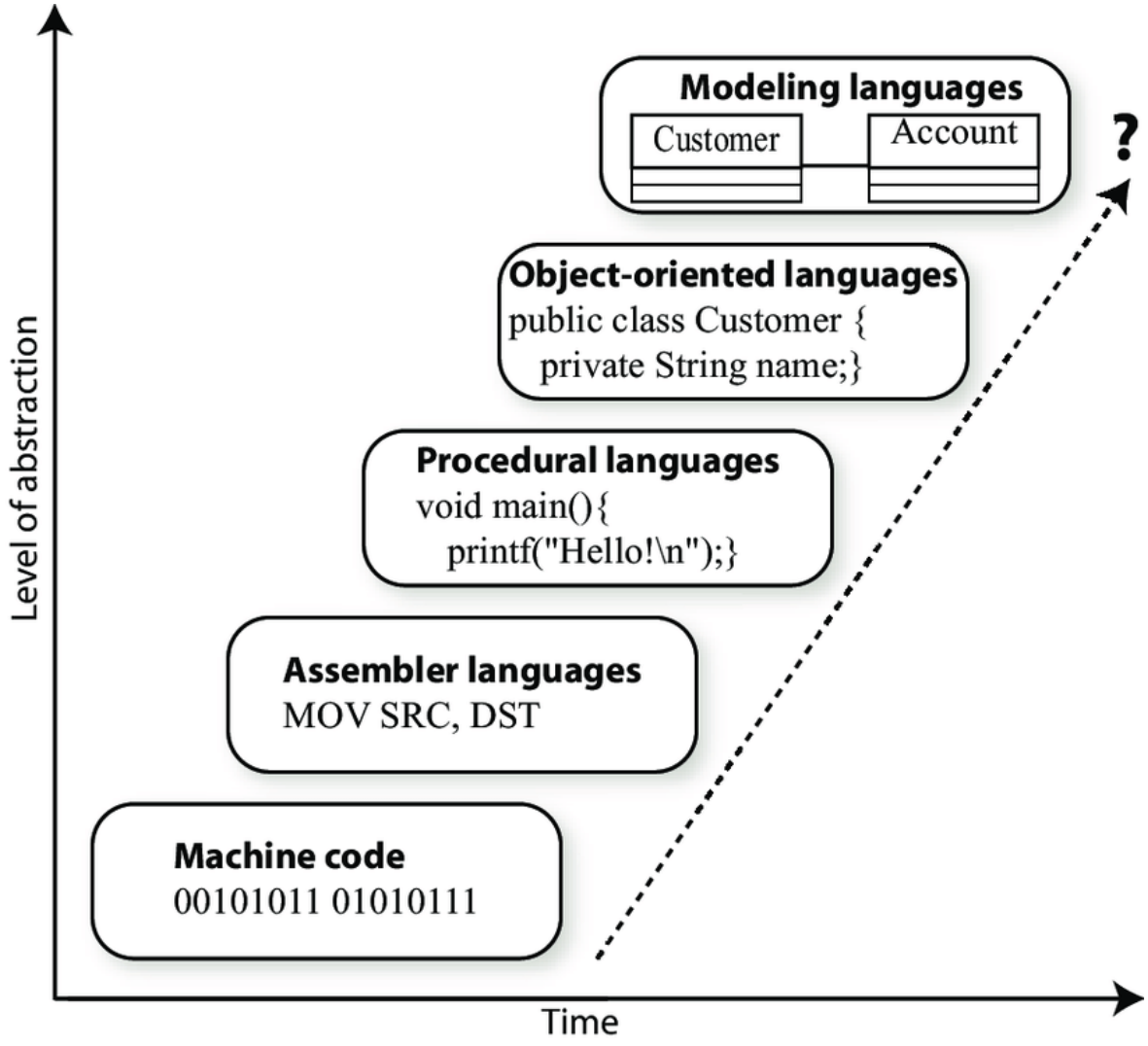


Figure: Levels of Abstraction by Paradigm

Paradigms VIII

- Examples of Abstraction
  - Concurrency Control
  - Memory Management
  - Virtual Machine
  - Operating Systems
  - Drivers
  - APIs
- All Programming languages and paradigms are attempts to **abstract low-level details**
  - allow the programmer to think about and solve problems at a higher level
    - different level!
    - or through a particular prism

Paradigms IX

- Computers understand operations at a very low level
  - i.e. moving bits from one place to another
- Of course we want to do things at a higher level than bitwise operations
- Have a look at the following operation:

```
a := (1 + 2) * 5
```

- so it's "one plus two multiplied by five"
- The low level steps needed to
  - carry out this evaluation
  - return the result (15)
  - and perform assignment (to a)
- are actually quite complex
  - recall it's a rock we *tricked into thinking*

Paradigms X

1. Values converted to binary
2. Calculations broken apart into assembly instructions e.g operations such as shifting a binary register left, or adding the binary complement of the contents of one register to another
3. Assigning the resulting value to the variable “a” (have to look up the variable location in **physical memory**)

Paradigms XI

- If we had no abstraction the programmer would need to specify all the register/binary-level steps *every time*
- That would make it *hard to focus* on solving the problem you are writing the program to solve!
- Abstraction also hides difference for example if the manner of performing low-level operations differs on one machine / instruction set the abstraction hides this.

Paradigms XII

What is state?

- A program can store data in **variables**:
  - which map to storage locations in memory.
  - The contents of these memory locations, at any given point in the program's execution, is called **the program's state.**
- State effects the behaviour of the program.
- The *more state* the *more unpredictable* the program
- "In programming mutable state is evil"
- Some paradigms would seek to do away with it completely.
- In others it is intrinsic, OOP without mutable state is not possible.

Paradigms XIII

```
var total = 0;
var a = 1;
var b = 5;
total = a + b
print total;
```

- In the beginning total is 0
- it's state is modified
- then printed
- No problems here but...
  - this can be complicated by:
    - control flow structures dependant on the value of variables
    - unpredictable values entered by users
    - coming from stored data

Paradigms XIV

```
variable = getFromUser("please enter a number")
if (variable > 10)
    this happens
else
    this happens instead

var2 = someMethod(variable)
if (var2 is an even number)
    something happens
```

Paradigms XV

Various Programming Paradigms

- Imperative / Procedural
- Functional
- Object-oriented
- Declarative
- Data Flow

[1] https://en.m.wikiversity.org/wiki/Introduction_to_Programming/About_Programming

Sources

Sources

- https://www.computerhope.com/jargon/a/al.htm
- https://en.wikipedia.org/wiki/Abstraction_(computer_science)
- https://en.wikipedia.org/wiki/C_(programming_language)

The End