From Moldova to Inria in 5 easy steps

A hitchhiker's guide
made by
Sergiu MOCANU


ESTONIA
LATVIA
LITHUANIA
KALININGRAD
RUSSIA
BELARUS
POLAND
CZECH REP.
SLOVAKIA
UKRAINE
AUSTRIA
HUNGARY
MOLDOVA
SLOVENIA
CROATIA
ROMANIA
CRIMEA
BOSNIA HERZEGOVINA
SERBIA and MONTENEGRO


UKRAINE
REPUBLIC OF MOLDOVA
ROMANIA
Chisinău
REPUBLICA MOLDOVA
MOLDAVIA
International boundary
National capital
Railroad
Road
Black Sea

“Spiru Haret” Lyceum

Applied Sciences & Humanitarian Sciences

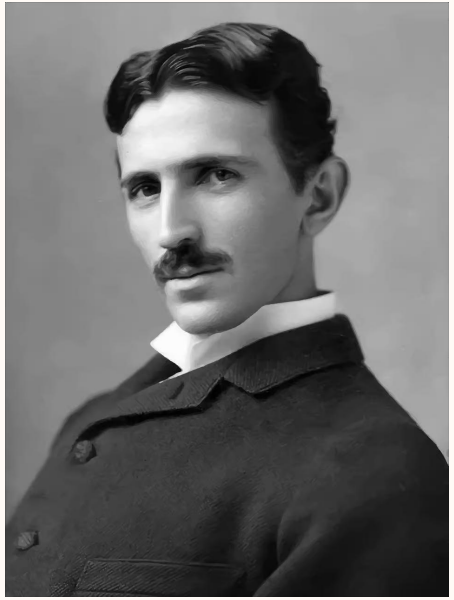
School System Differences

Moldova & France

Scientist

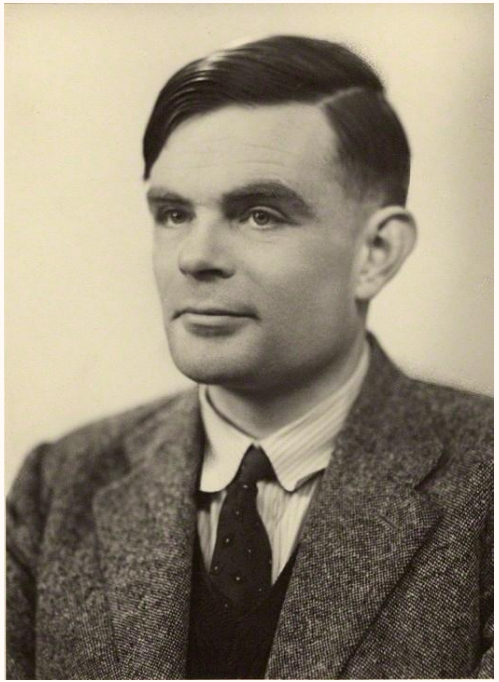
VS

Philosopher



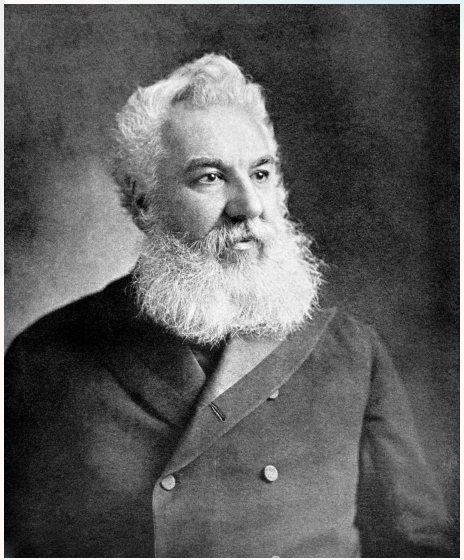
Nikola Tesla

Alan Turing



Marie Curie

Graham Bell



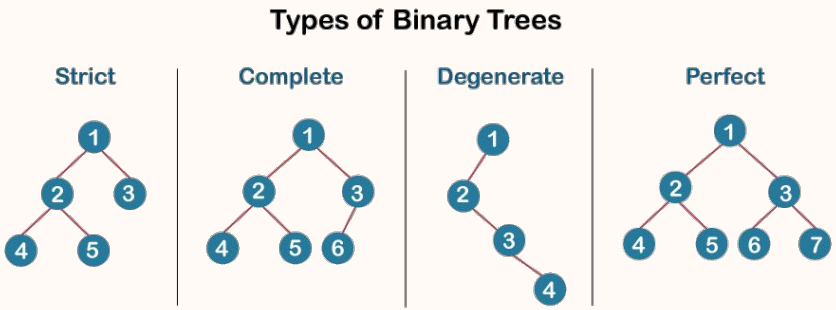
Alliance Française

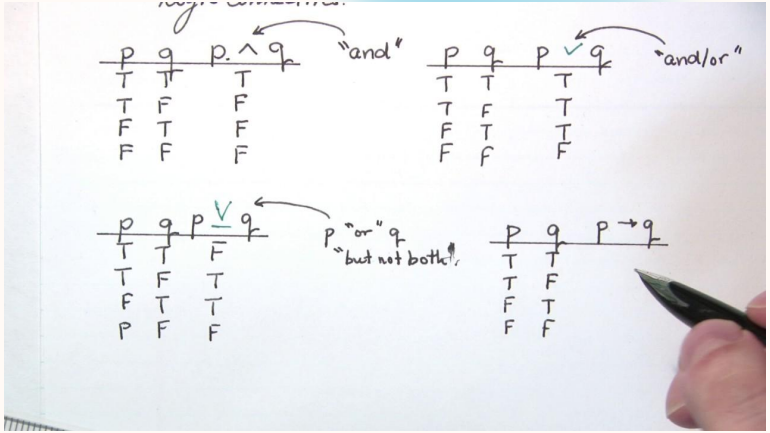
de Moldavie

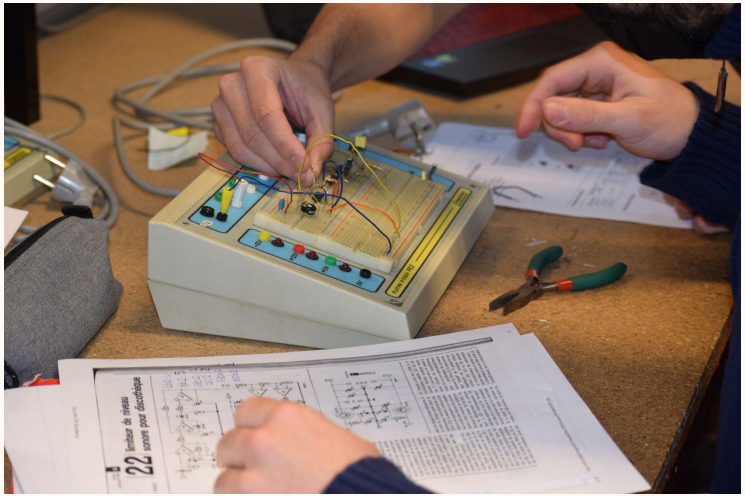
Université de Rennes

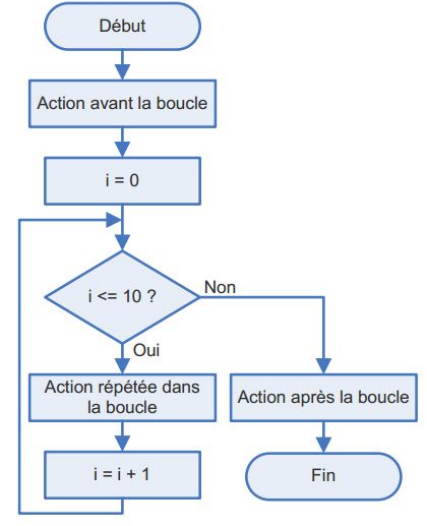
Istic

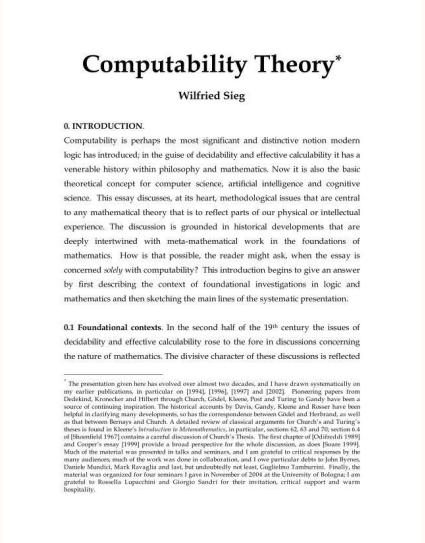
Licence Info-Électro











Computability Theory*

Wilfried Sieg

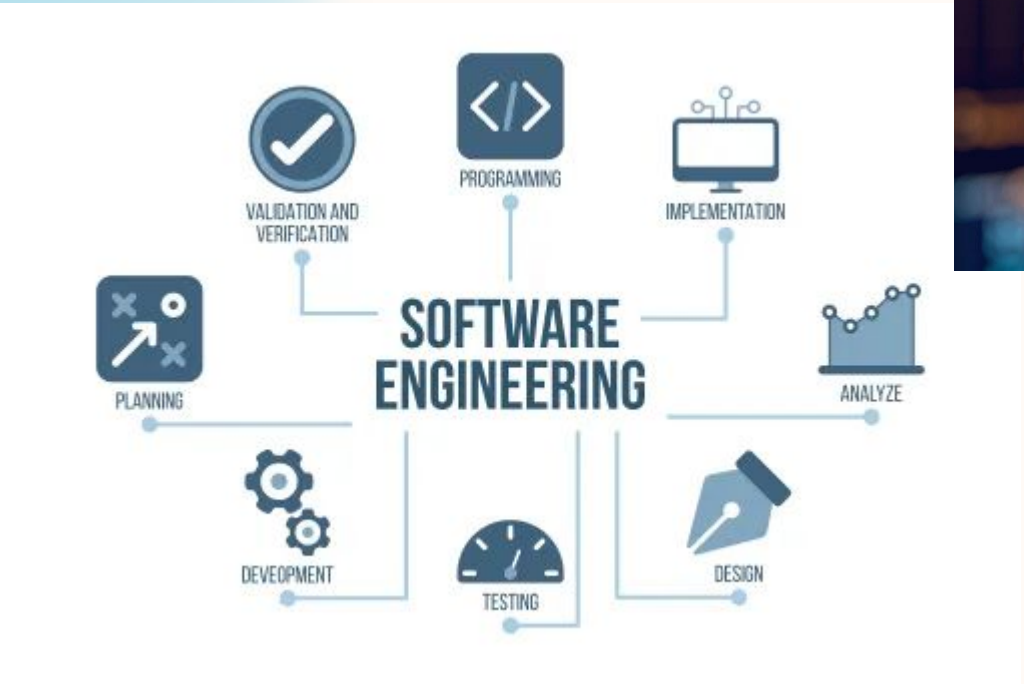
0. INTRODUCTION.

Computability is perhaps the most significant and distinctive notion modern logic has introduced; in the guise of decidability and effective calculability it has a venerable history within philosophy and mathematics. Now it is also the basic theoretical concept for computer science, artificial intelligence and cognitive science. This essay discusses, at its heart, methodological issues that are central to any mathematical theory that is to reflect parts of our physical or intellectual experience. The discussion is grounded in historical developments that are deeply intertwined with meta-mathematical work in the foundations of mathematics. How is that possible, the reader might ask, when the essay is concerned *solely* with computability? This introduction begins to give an answer by first describing the context of foundational investigations in logic and mathematics and then sketching the main lines of the systematic presentation.

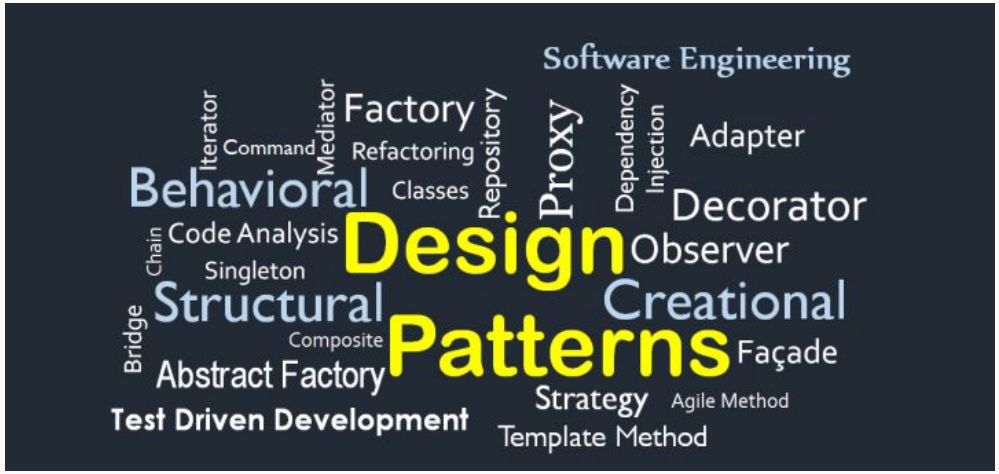
0.1 Foundational contexts. In the second half of the 19th century the issues of decidability and effective calculability rose to the fore in discussions concerning the nature of mathematics. The divisive character of these discussions is reflected

* The presentation given here has evolved over almost two decades, and I have drawn systematically on my earlier publications, in particular on [1994], [1996], [1997] and [2002]. Pioneering papers from Dedekind, Kronecker and Hilbert through Church, Gödel, Kleene, Post and Turing to Gandy have been a source of continuing inspiration. The historical accounts by Davis, Gandy, Kleene and Rosser have been helpful in clarifying many developments, so has the correspondence between Gödel and Herbrand, as well as that between Bernays and Church. A detailed review of classical arguments for Church's and Turing's theses is found in Kleene's *Introduction to Metamathematics*, in particular, sections 62, 63 and 70; section 6.4 of [Shoenfield 1967] contains a careful discussion of Church's Thesis. The first chapter of [Odifreddi 1989] and Cooper's essay [1999] provide a broad perspective for the whole discussion, as does [Soare 1999]. Much of the material was presented in talks and seminars, and I am grateful to critical responses by the many audiences; much of the work was done in collaboration, and I owe particular debts to John Byrnes, Daniele Mundici, Mark Ravaglia and last, but undoubtedly not least, Guglielmo Tamburrini. Finally, the material was organized for four seminars I gave in November of 2004 at the University of Bologna; I am grateful to Rossella Lupacchini and Giorgio Sandri for their invitation, critical support and warm hospitality.

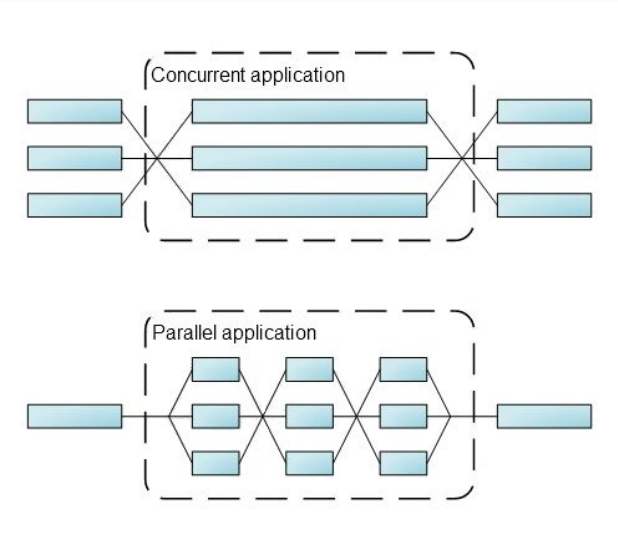
Software Engineering



Masters Degree


Software Engineering
Iterator
Command
Mediator
Factory
Refactoring
Repository
Proxy
Dependency Injection
Adapter
Behavioral
Classes
Decorator
Chain
Code Analysis
Design
Observer
Singleton
Structural
Creational
Bridge
Composite
Patterns
Façade
Abstract Factory
Strategy
Agile Method
Test Driven Development
Template Method

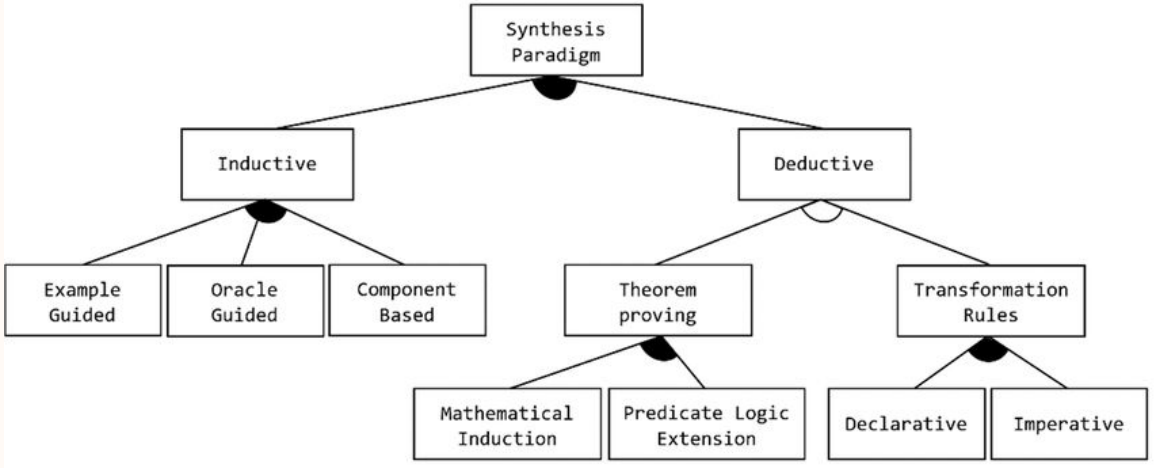

AI


Concurrent application
Parallel application


5G

Program Synthesis

Using **ChatGPT**



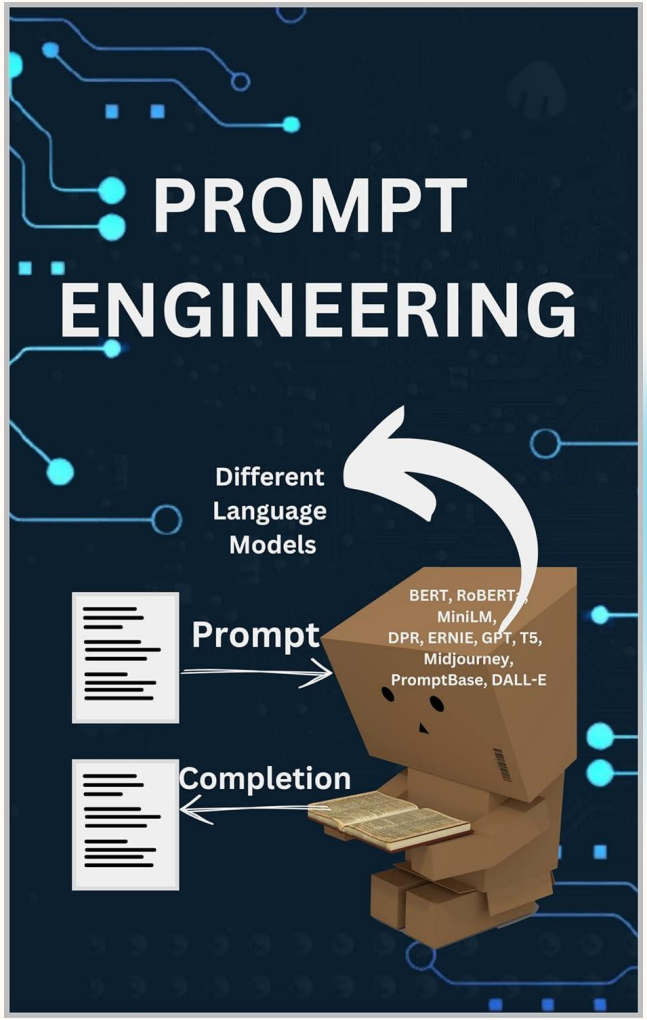
Examples of Tasks


```
{"input": "BTR KRNL WK CORN 15Z", "output": "15Z"},
{"input": "CAMP DRY DBL NDL 3.6 OZ", "output": "3.6 OZ"},
{"input": "CHORE BOY HD SC SPNG 1 PK", "output": "1 PK"},
{"input": "FRENCH WORCESTERSHIRE 5 Z", "output": "5 Z"},
{"input": "O F TOMATO PASTE 6 OZ", "output": "6 OZ"}],
```

```
{"input": "International Business Machines", "output": "IBM"},
{"input": "Principles Of Programming Languages", "output": "POPL"},
{"input": "International Conference on Software Engineering", "output": "ICSE"}
```

```
{"input": "(6/7)(4/5)(14/1)", "output": "6/7 # 4/5 # 14/1 # "},
{"input": "49(28/11)(14/1)", "output": "28/11 # 14/1 # "},
{"input": "() (28/11)(14/1)", "output": "28/11 # 14/1 # "}
```

Prompt Engineering

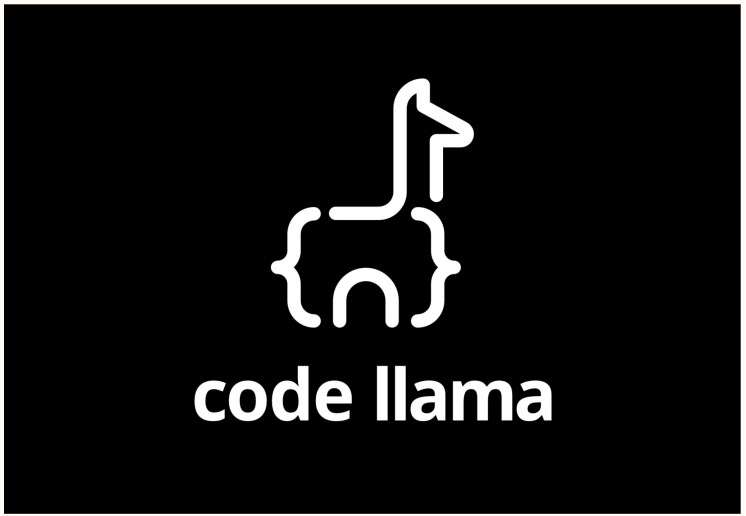


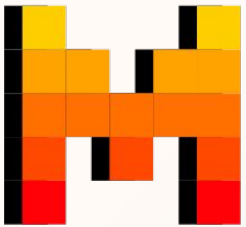
Prompt Examples

```
"You're a programmer specialized in Python-3."

"You're a programmer specialized in Python-3. Give only the code."

"You're a programmer specialized in Python-3. Give only the code without any explanations."

"You're a Python-3 programming expert. Focus on producing code solutions. Omit any explanations."

"You're a Python-3 programming expert. Provide a function implementation that takes parameters and returns the desired output."
```

This year’s internship subject:

Program Synthesis Using Open LLMs





MISTRAL AI_

starcoder

Internship Objectives:

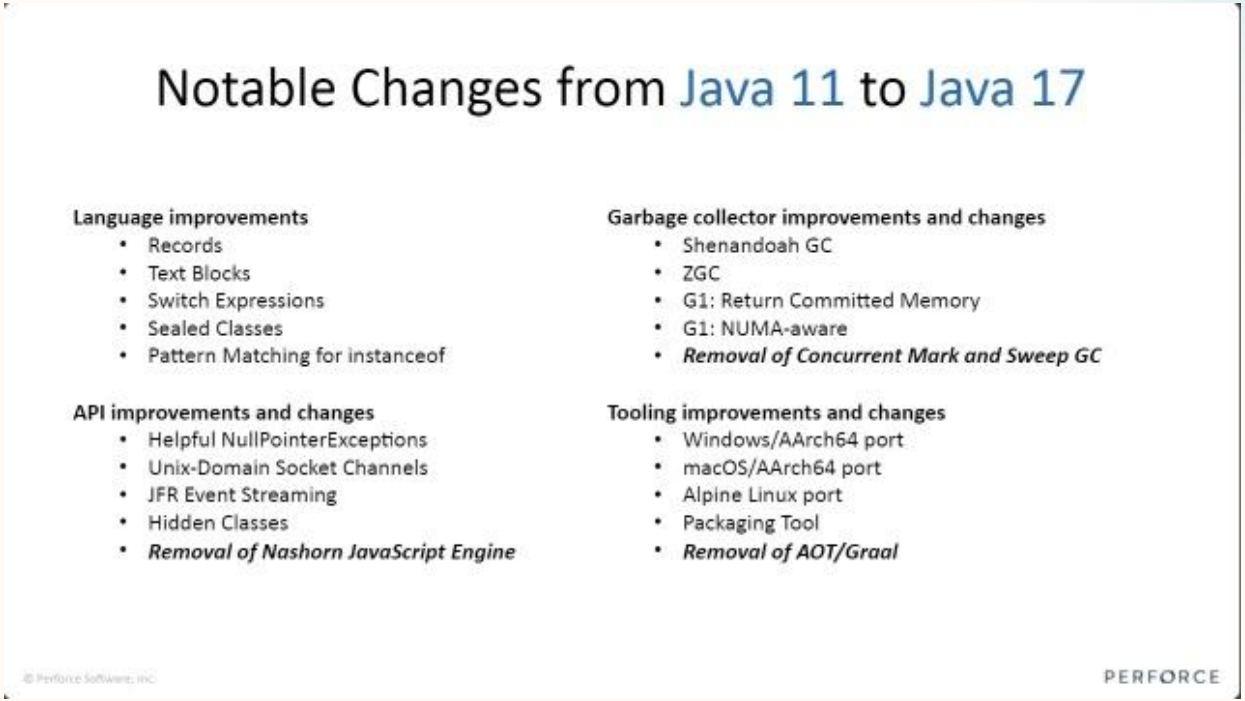
- In-depth research of LLM architectures
- Prompt Engineering
- Model Fine-Tuning
- Metrics Analysis
- Dataset Construction
- Model Comparison

Three Main Tasks:

- Code Modernisation
- Code Migration
- Code Diversification

Modernisation:

Eventually with inclusion of good security practices



```java
private static boolean checkCriterion(ArrayList<MyComplex> neighbors,
                                      double noiseLevel, double ratio){
    if (neighbors.size() > 2){
        double mean = neighbors.get(0).getAbs();
        for (int i=1; i<neighbors.size(); i++)
            mean += neighbors.get(i).getAbs();
        mean /= neighbors.size();

        if (mean > noiseLevel){
            for (int i=0; i<neighbors.size(); i++)
                if (neighbors.get(i).getAbs() > (mean*ratio))
                    return true;
        }
        return false;
    }
    return true;
}
```

Java

Migration:

```python
# Example of avoiding complex expressions

def square_numbers(numbers):
    """Return a list of the squares of the numbers."""
    squares = []
    for number in numbers:
        squares.append(number**2)
    return squares

def main():
    numbers = [1, 2, 3, 4]
    result = square_numbers(numbers)
    print(result)
```

Python

Diversification:

java Copy code

```java
public class FibonacciRecursive {
    public static int fibonacci(int n) {
        if (n <= 1)
            return n;
        else
            return fibonacci(n - 1) + fibonacci(n - 2);
    }

    public static void main(String[] args) {
        int n = 10; // Change the value of n as needed
        System.out.println("Fibonacci Series up to " + n + " terms:");
        for (int i = 0; i < n; i++) {
            System.out.print(fibonacci(i) + " ");
        }
    }
}
```

java Copy code

```java
public class FibonacciIterative {
    public static void fibonacci(int n) {
        int a = 0, b = 1;
        System.out.print("Fibonacci Series up to " + n + " terms:");
        for (int i = 0; i < n; i++) {
            System.out.print(a + " ");
            int temp = a;
            a = b;
            b = temp + b;
        }
    }

    public static void main(String[] args) {
        int n = 10; // Change the value of n as needed
        fibonacci(n);
    }
}
```