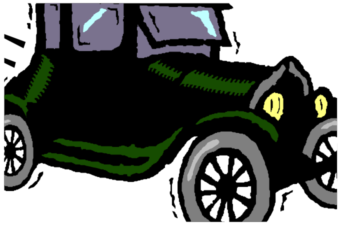


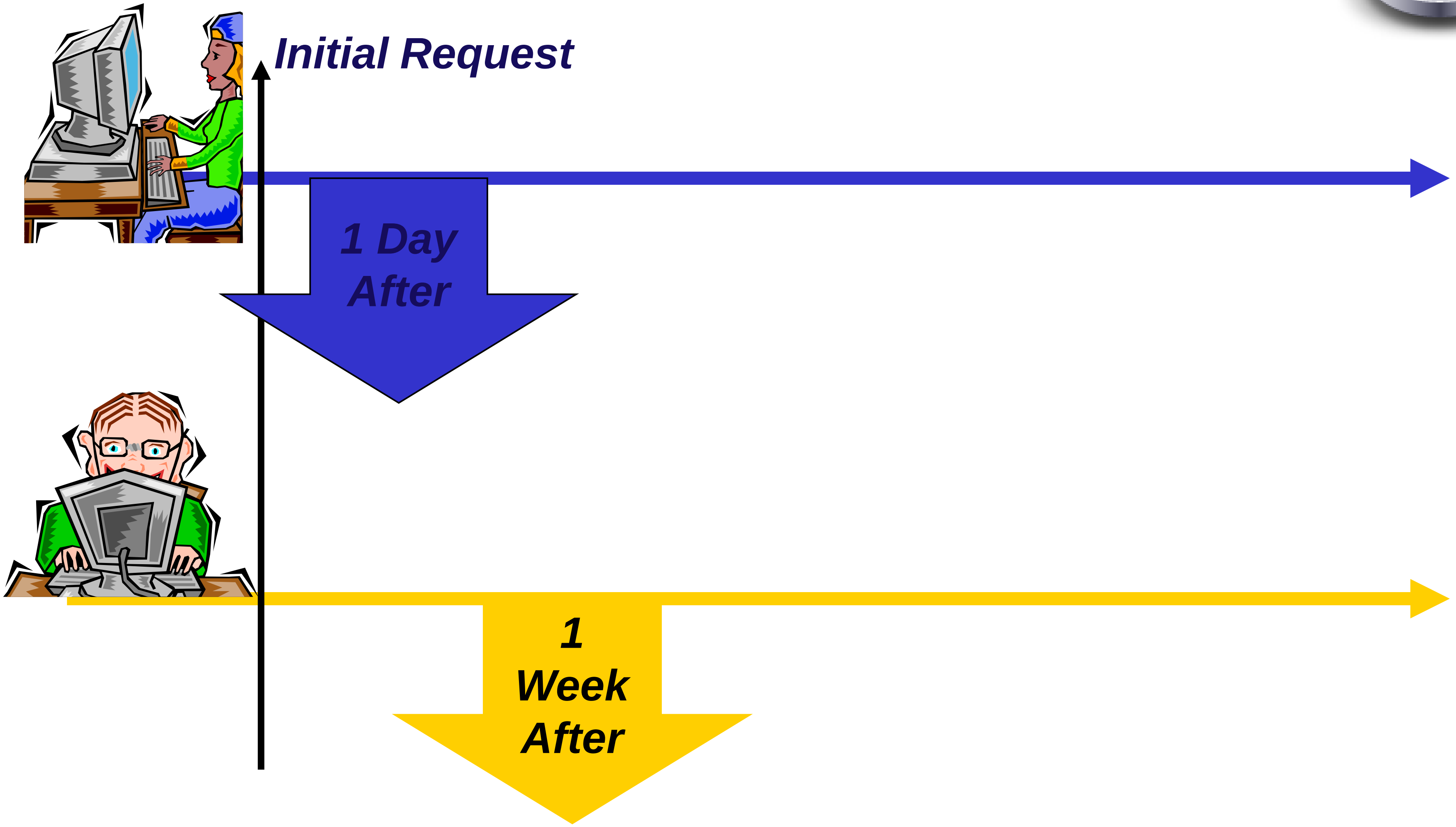
Conception Avancée de Base de Données

Design For Changes

Traduction en cours

Why we want It !!

Changes Request : flexible solution

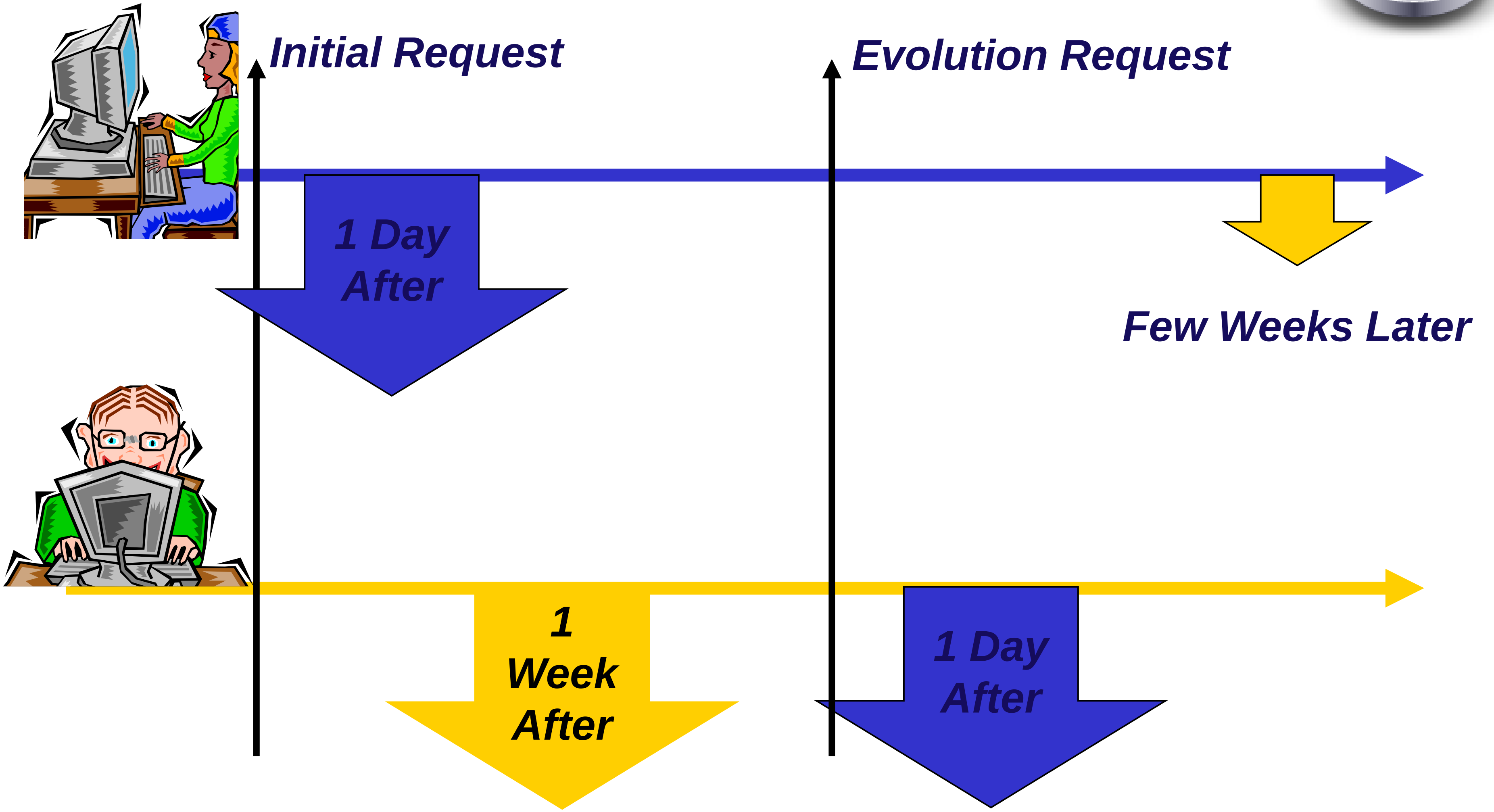


Initial Request

1 Day After

1 Week After

Changes Request : flexible solution



Initial Request

Evolution Request

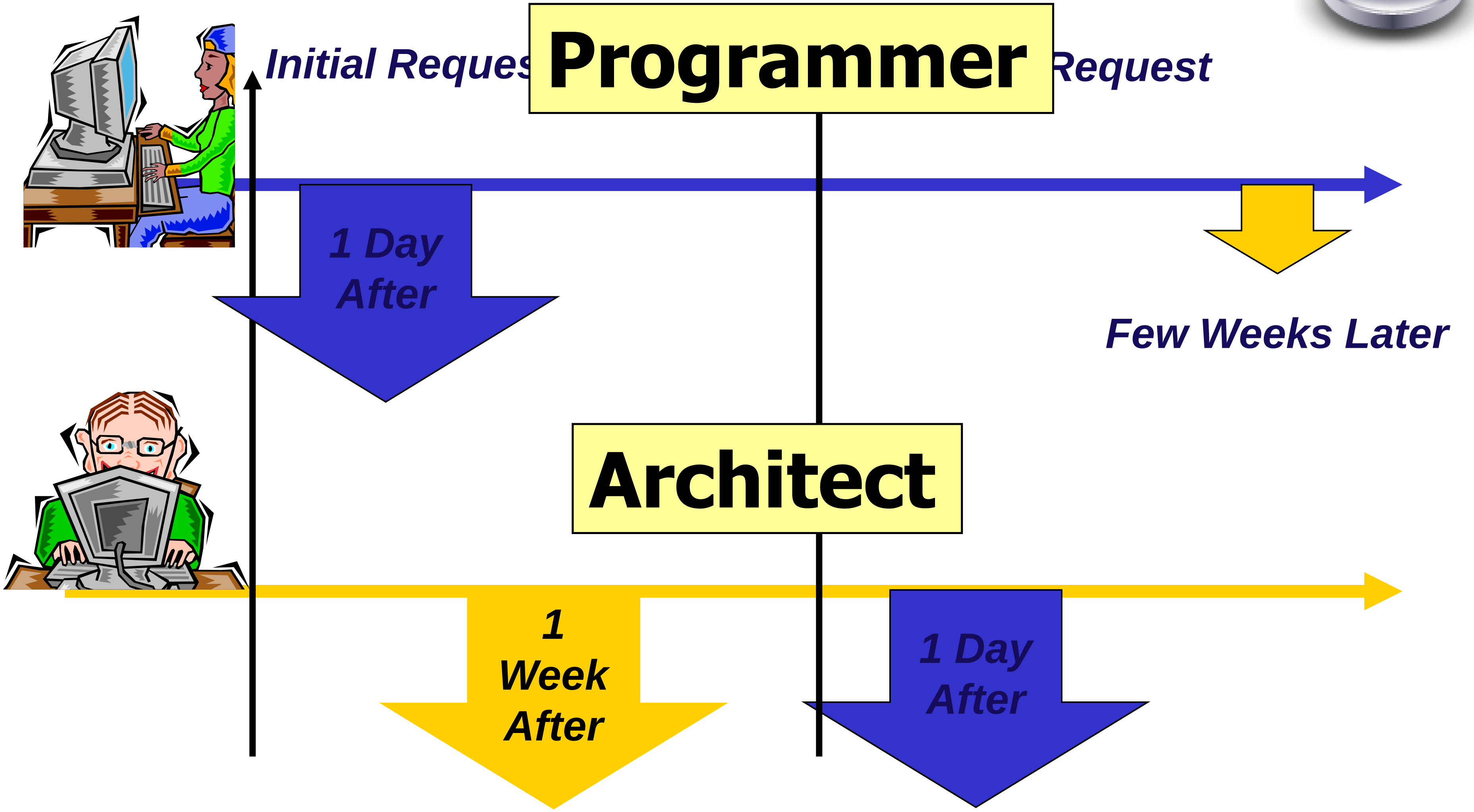
1 Day After

Few Weeks Later

1 Week After

1 Day After

Changes Request : flexible solution



Initial Reques

Programmer

Request

1 Day After

Few Weeks Later

Architect

1 Week After

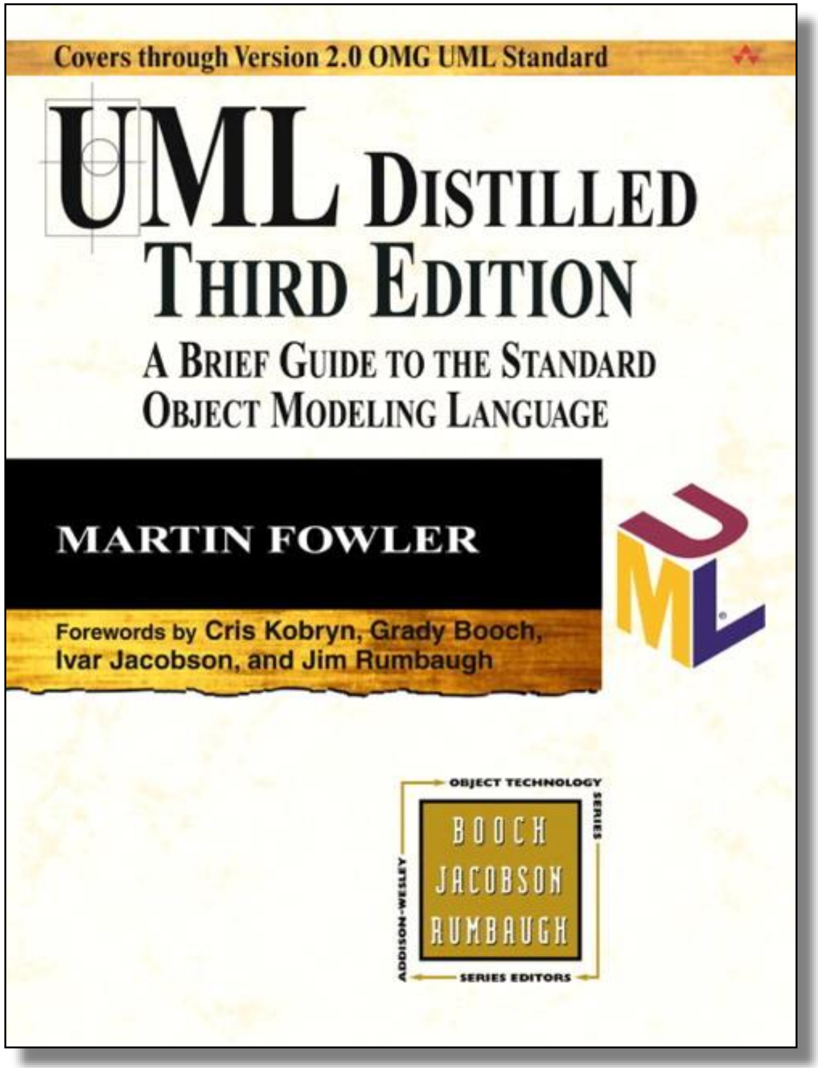
1 Day After

Changes Sources During Development

- Requirements :
  - Customers Discover What they Really Want During or at the End of Developments
- Technology
  - Performances Are Increasing With Time
- Skill
  - We Learn and Understand the Problem and We Discover the Right Solution on the Job
- Short Term Politic
  - No Comments

Martin FOWLER

Martin Fowler



Covers through Version 2.0 OMG UML Standard

UML DISTILLED THIRD EDITION

A BRIEF GUIDE TO THE STANDARD OBJECT MODELING LANGUAGE

MARTIN FOWLER

Forewords by Cris Kobryn, Grady Booch, Ivar Jacobson, and Jim Rumbaugh

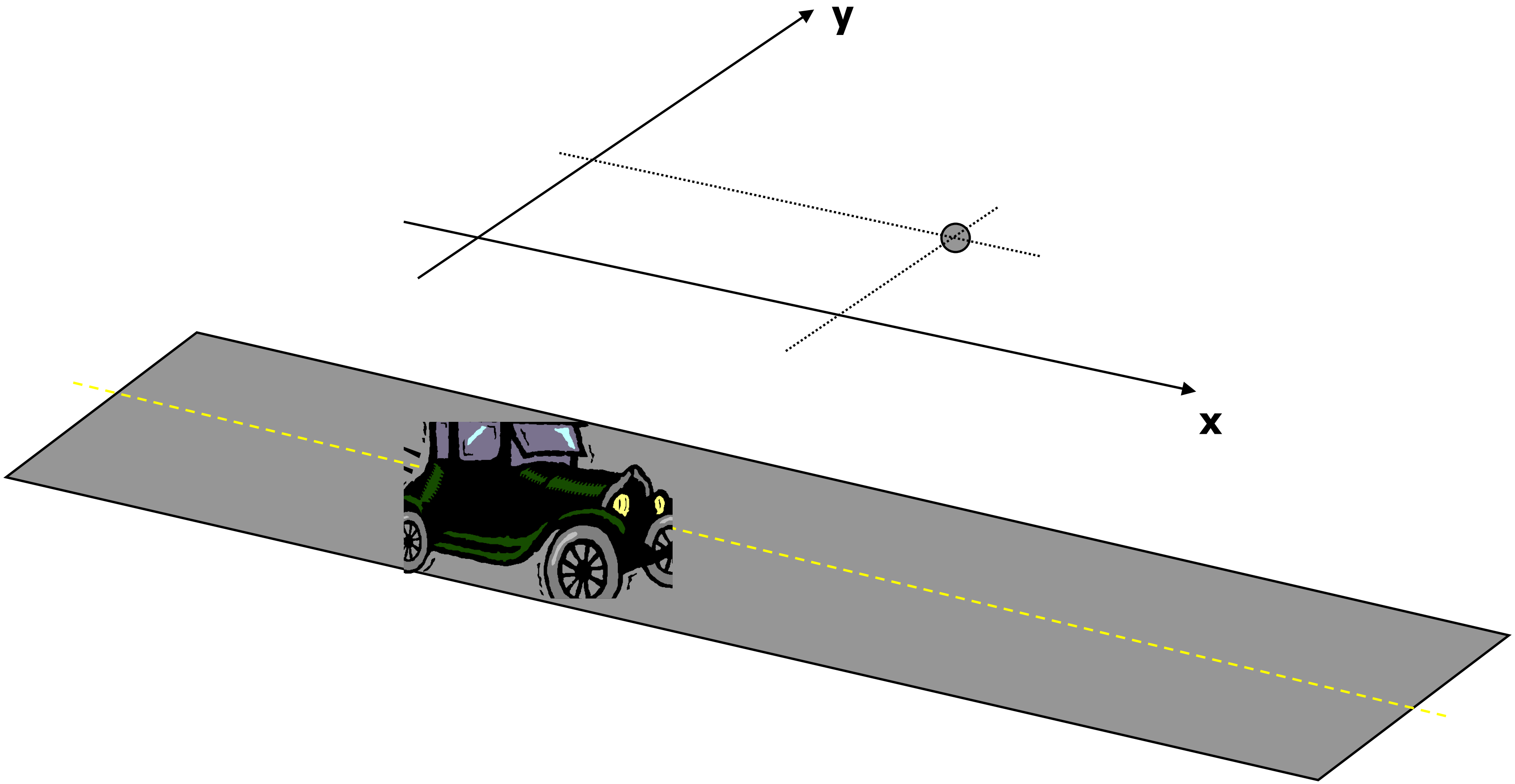
OBJECT TECHNOLOGY SERIES

BOOCH JACOBSON RUMBAUGH

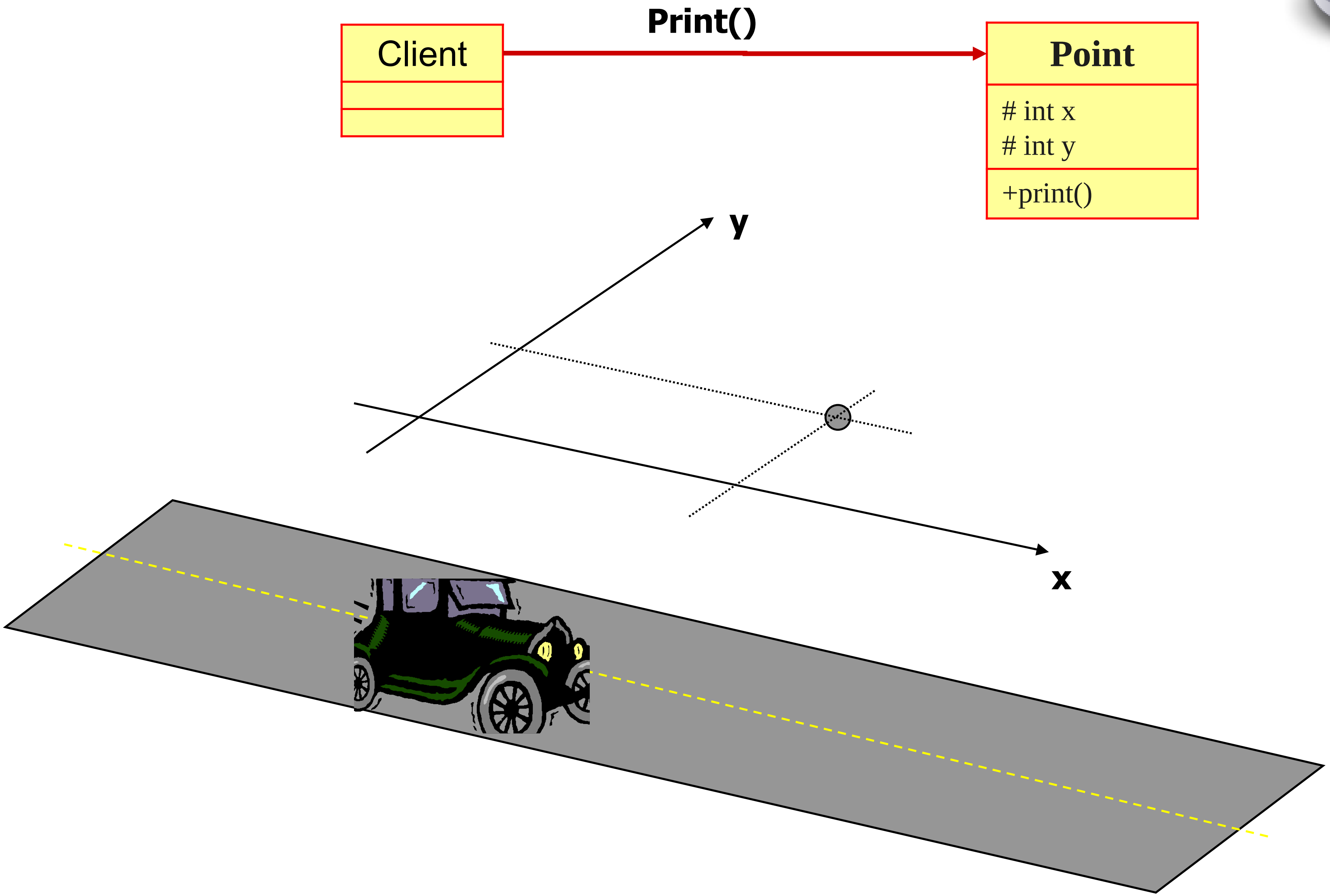
SERIES EDITORS

ADDISON-WESLEY

Programming in “present” tense



Programming in “present” tense



Client

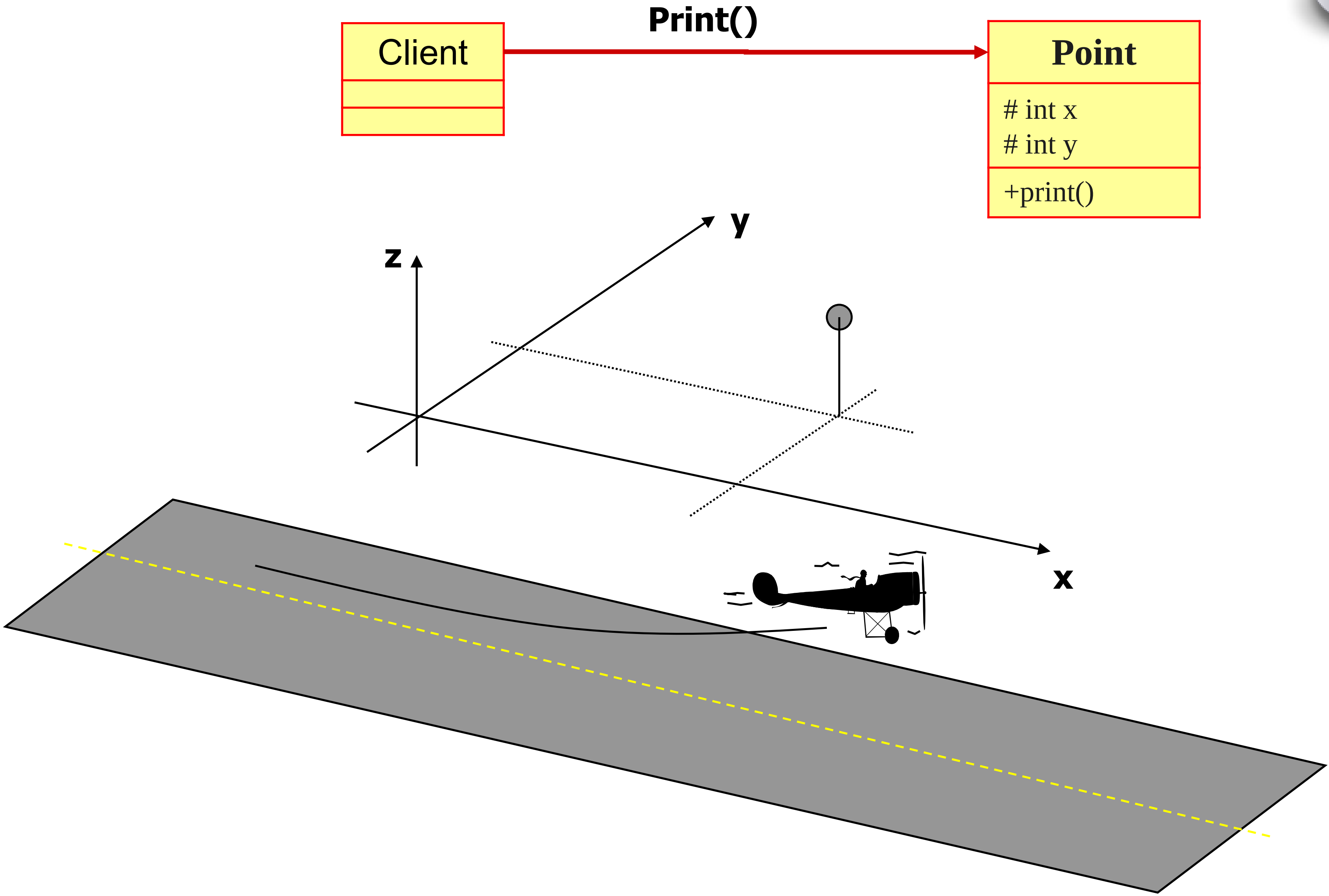
Print()

Point

# int x
# int y

+print()

Future



Client

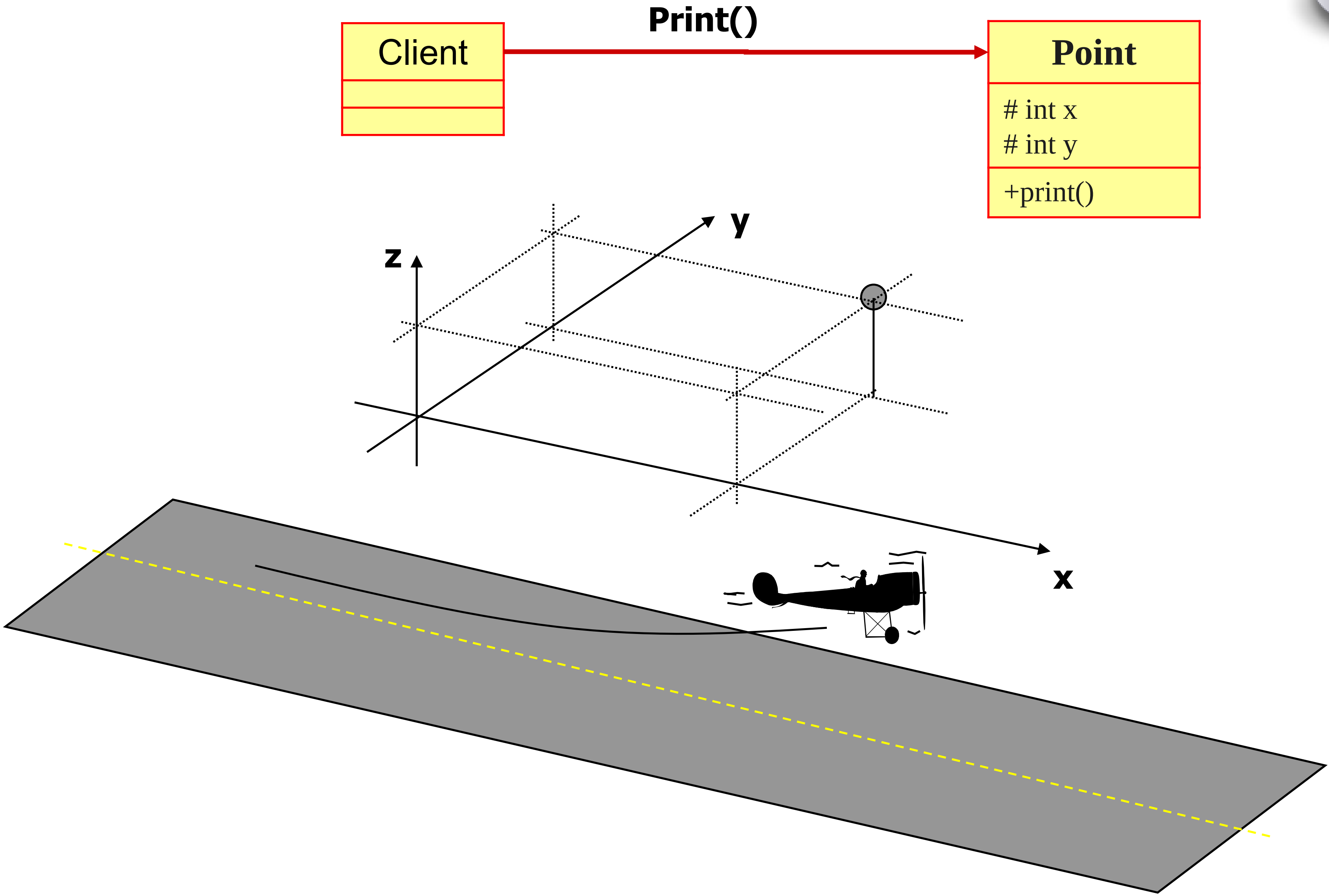
Print()

Point

# int x
# int y

+print()

Future



Print()

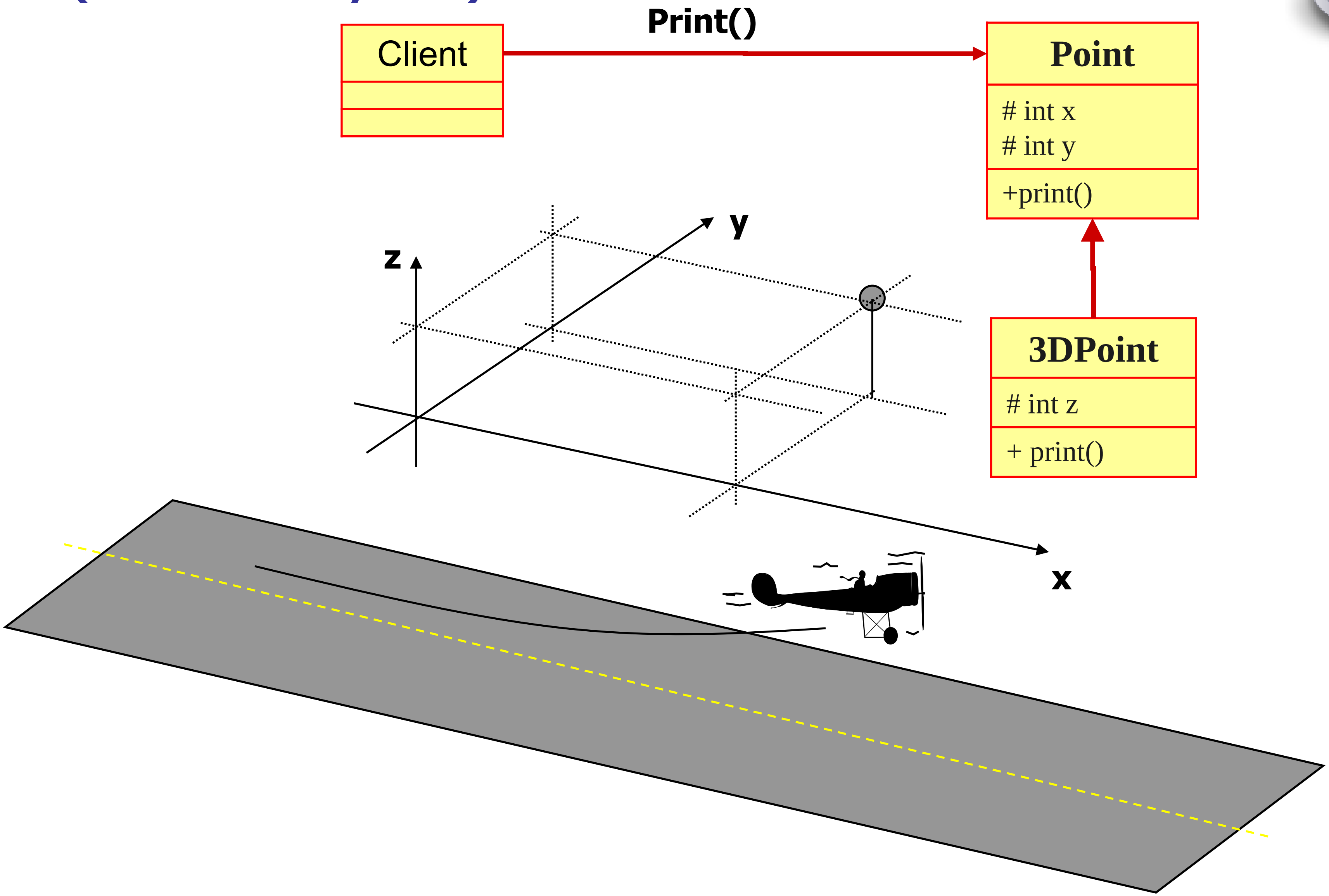
Client

Point

# int x
# int y

+print()

Programming in “Future” tense (Scott Meyers)



Client

Print()

Point

# int x
# int y

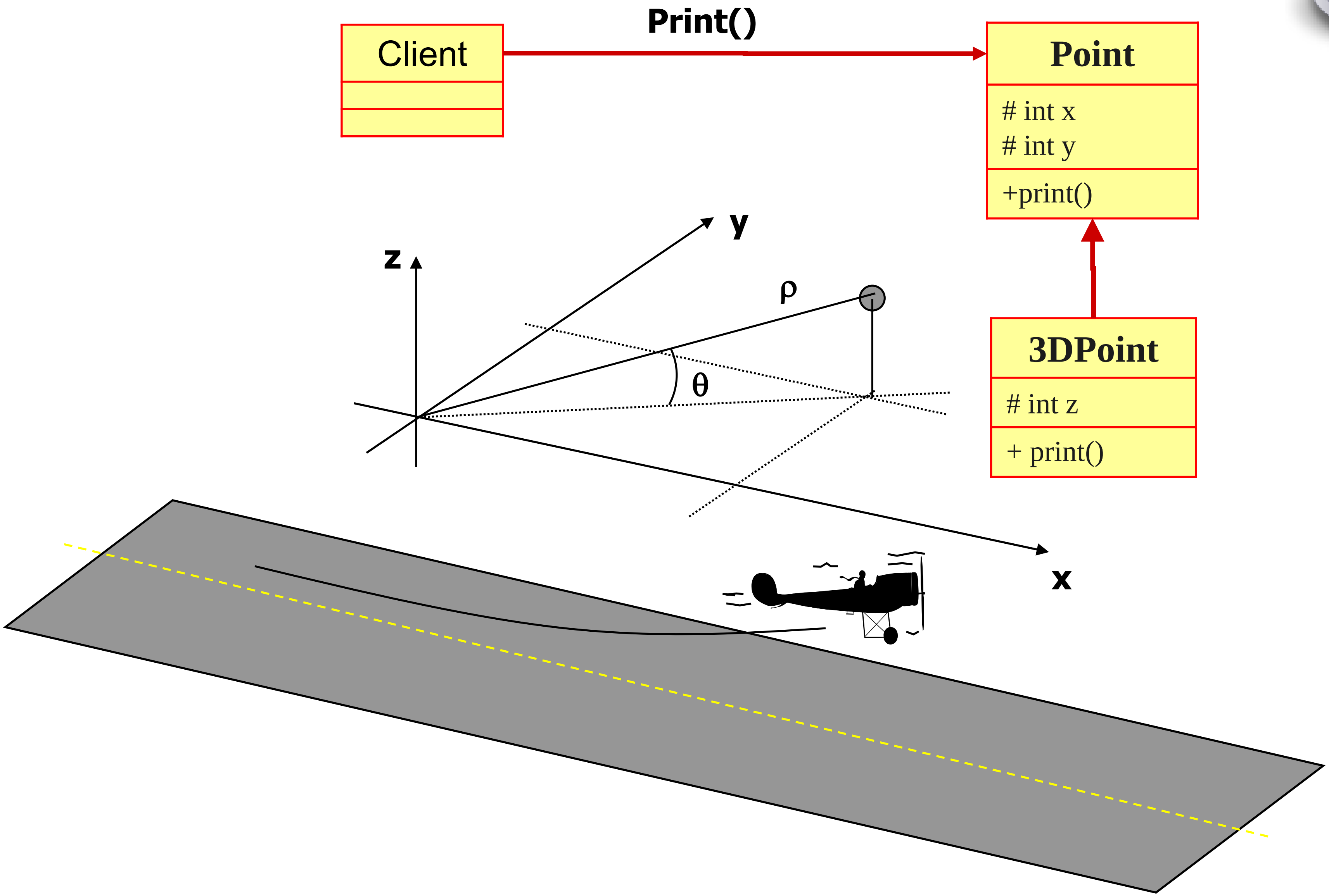
+print()

3DPoint

# int z

+ print()

Programming in “Future” tense



Client

Print()

Point

# int x
# int y

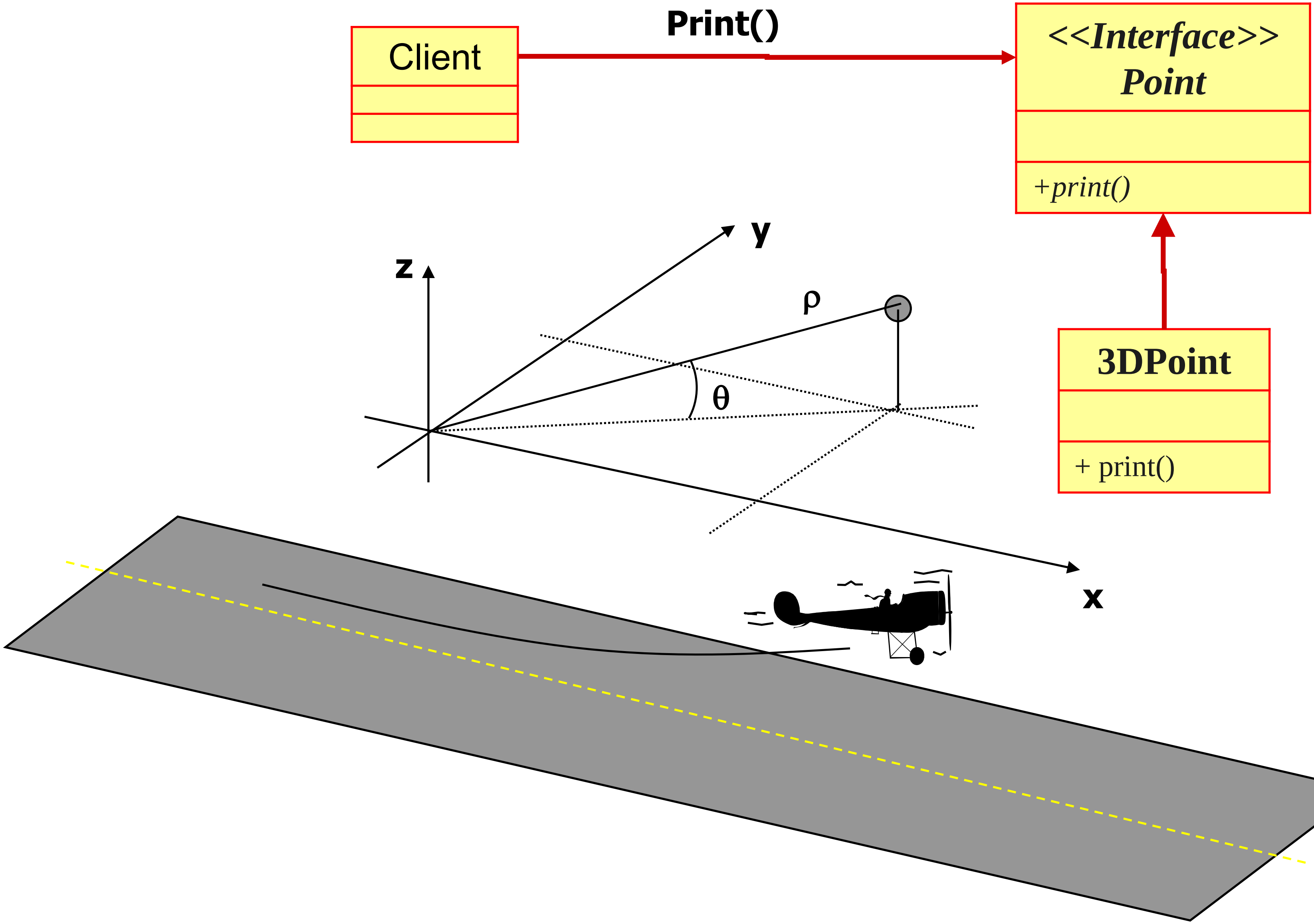
+print()

3DPoint

# int z

+ print()

Programming in “Future” tense



Client

Print()

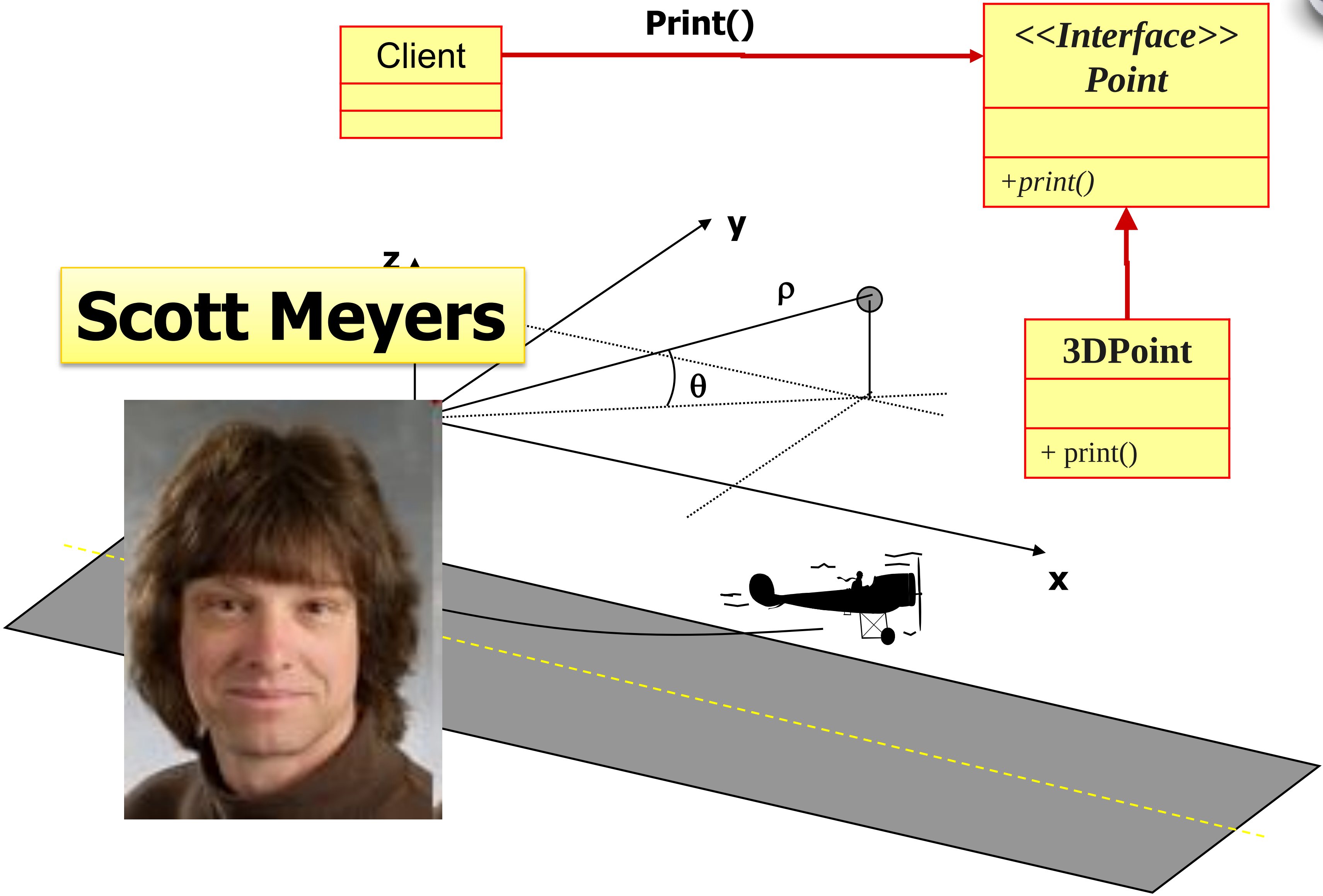
<<Interface>>
Point

+print()

3DPoint

+ print()

Programming in “Future” tense



Client

Print()

<<Interface>>
Point

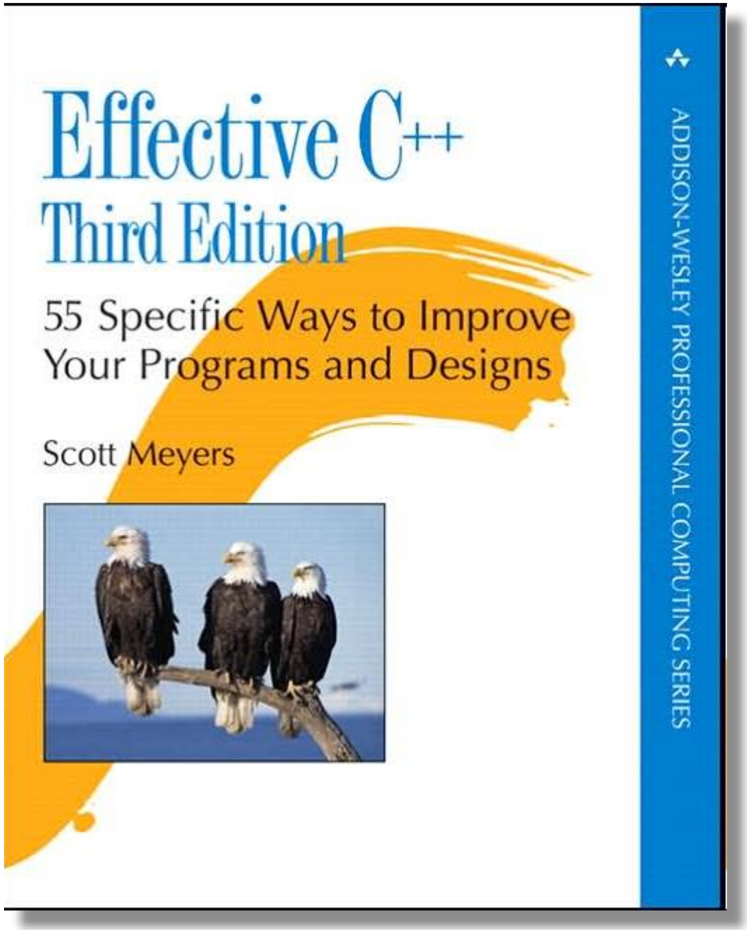
+print()

3DPoint

+ print()

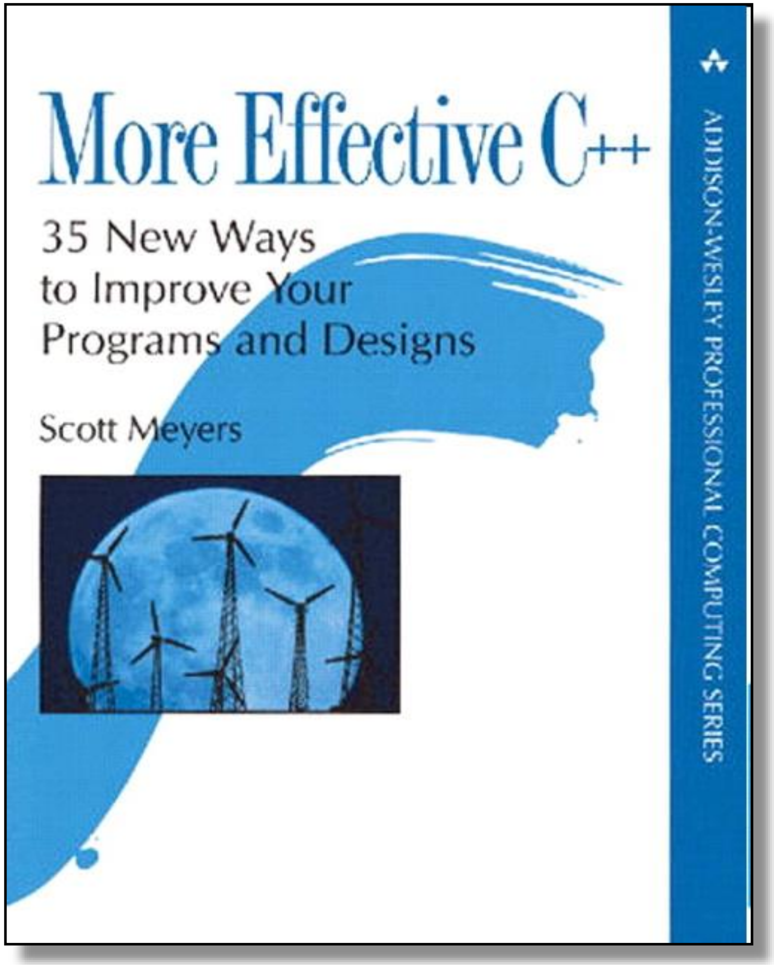
Scott Meyers

Scott Meyers


Effective C++
Third Edition
55 Specific Ways to Improve Your Programs and Designs
Scott Meyers
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Effective C++

Scott Meyers



More Effective C++

35 New Ways to Improve Your Programs and Designs

Scott Meyers

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

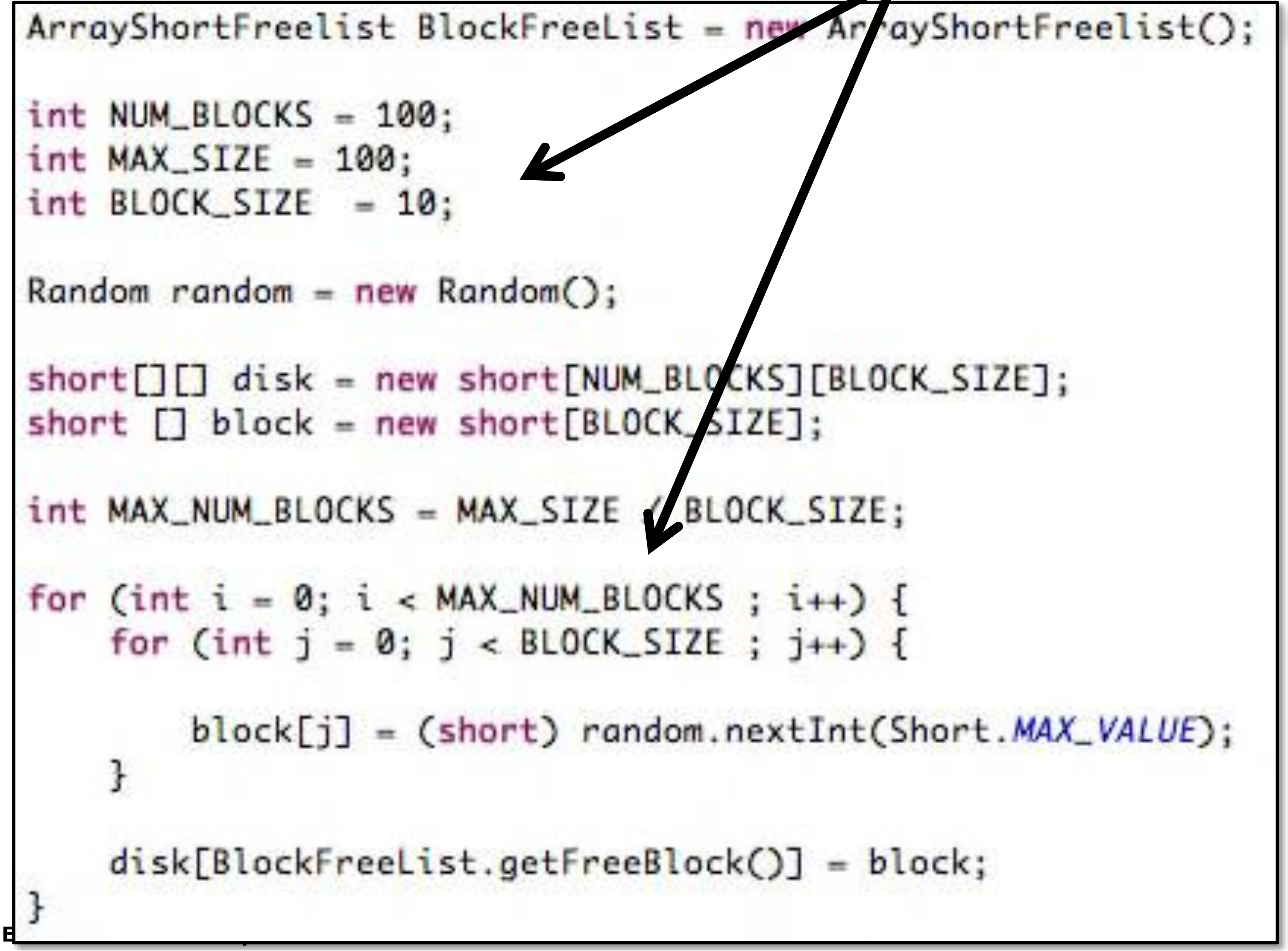
More Effective C++

Managing the changes

- Isolate Likely to Change Items :
  - Modularity : Interchange
  - No Global Items : Change Localization
  - No Hardcode Items (Magic Numbers) : Hide implementation changes
- No Assumption on Implementation !!!!!
  - Design by Interface
- Separation of Concerns
  - Business Services
  - Technical Services

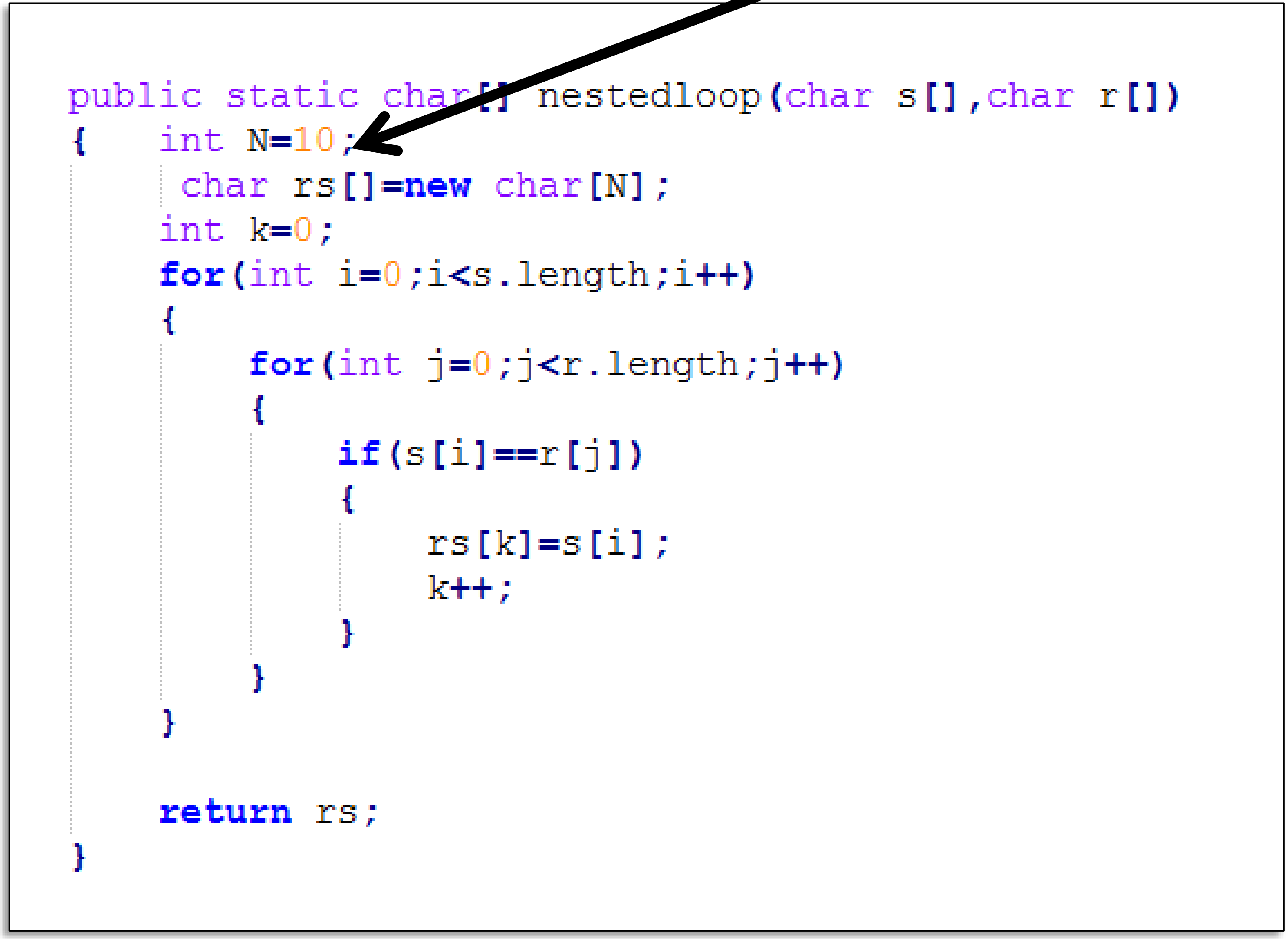
Java snippet

Pas de constantes en dur



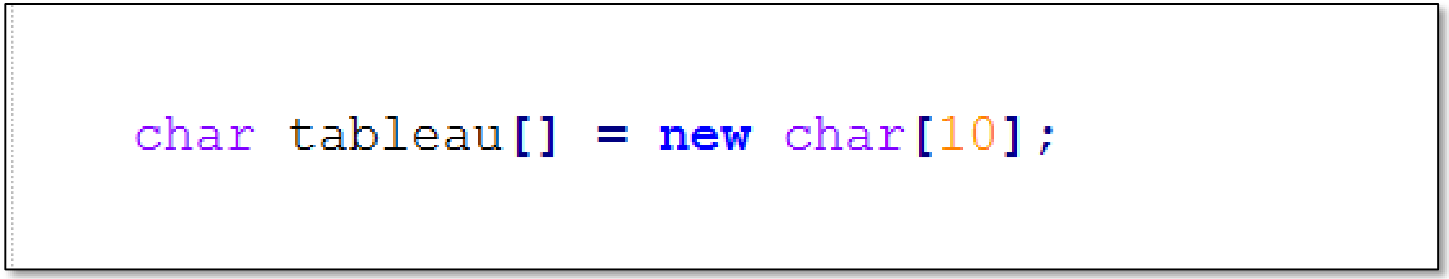
```java
ArrayShortFreelist BlockFreeList = new ArrayShortFreelist();

int NUM_BLOCKS = 100;
int MAX_SIZE = 100;
int BLOCK_SIZE  = 10;

Random random = new Random();

short[][] disk = new short[NUM_BLOCKS][BLOCK_SIZE];
short [] block = new short[BLOCK_SIZE];

int MAX_NUM_BLOCKS = MAX_SIZE / BLOCK_SIZE;

for (int i = 0; i < MAX_NUM_BLOCKS ; i++) {
    for (int j = 0; j < BLOCK_SIZE ; j++) {

        block[j] = (short) random.nextInt(Short.MAX_VALUE);
    }

    disk[BlockFreeList.getFreeBlock()] = block;
}
```

Pas de constantes en dur



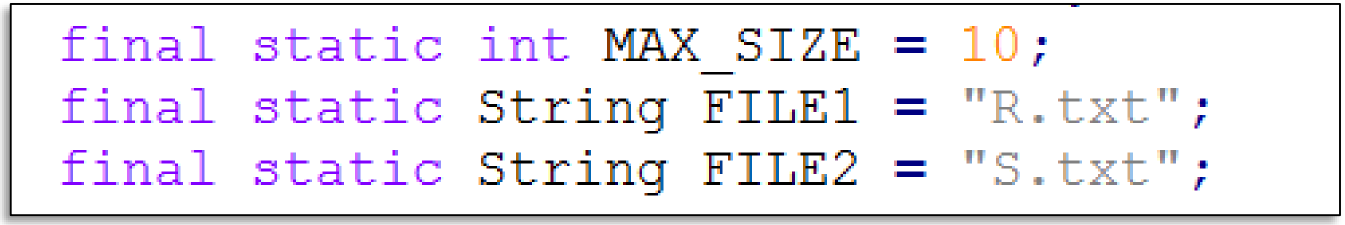
```
public static char[] nestedloop(char s[],char r[])
{    int N=10;
     char rs[]=new char[N];
     int k=0;
     for(int i=0;i<s.length;i++)
     {
         for(int j=0;j<r.length;j++)
         {
             if(s[i]==r[j])
             {
                 rs[k]=s[i];
                 k++;
             }
         }
     }

     return rs;
}
```

Constantes en dur (magic number)

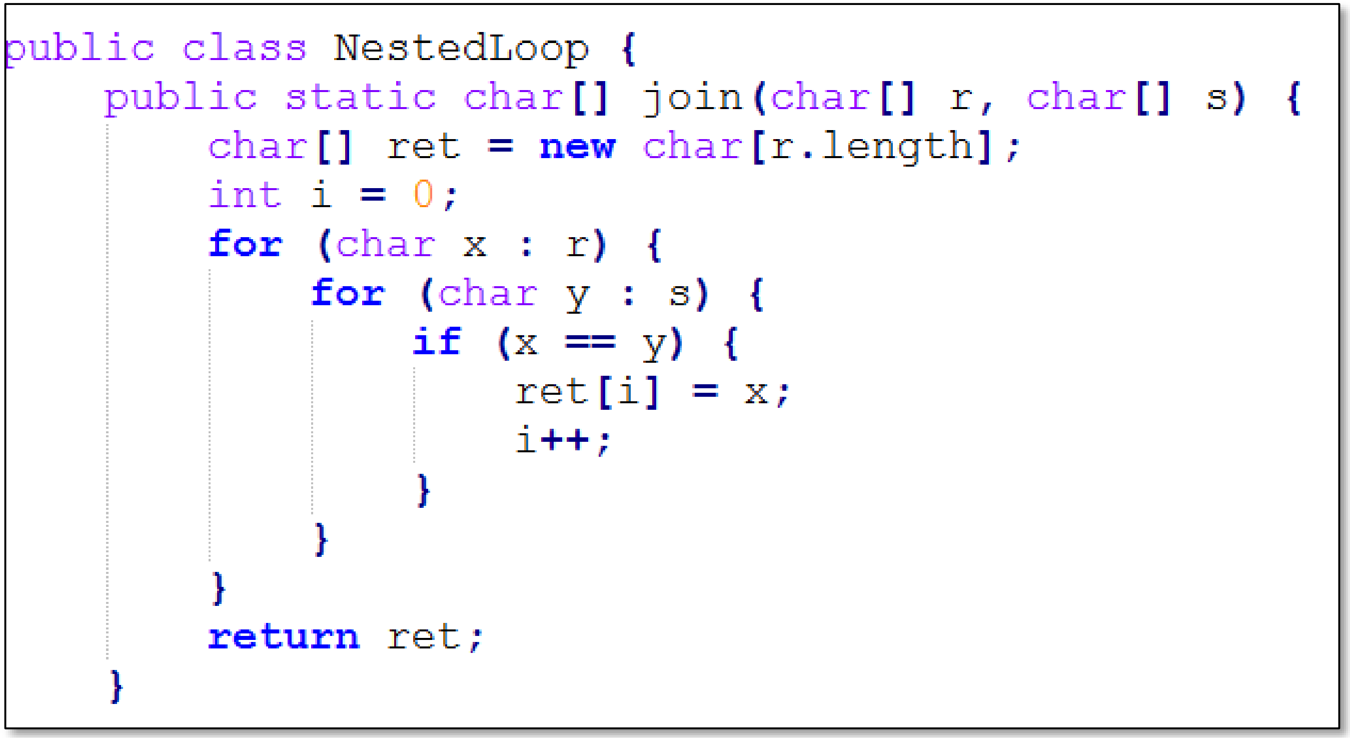


```
char tableau[] = new char[10];
```

C'est mieux !

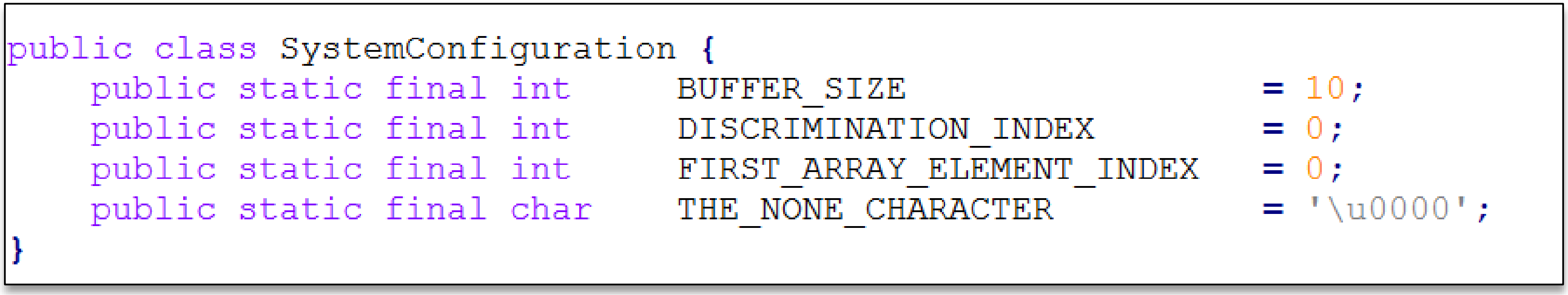


```
final static int MAX_SIZE = 10;
final static String FILE1 = "R.txt";
final static String FILE2 = "S.txt";
```



```
public class NestedLoop {
    public static char[] join(char[] r, char[] s) {
        char[] ret = new char[r.length];
        int i = 0;
        for (char x : r) {
            for (char y : s) {
                if (x == y) {
                    ret[i] = x;
                    i++;
                }
            }
        }
        return ret;
    }
```

C’ est très bien !!!



```
public class SystemConfiguration {
    public static final int     BUFFER_SIZE                 = 10;
    public static final int     DISCRIMINATION_INDEX        = 0;
    public static final int     FIRST_ARRAY_ELEMENT_INDEX   = 0;
    public static final char    THE_NONE_CHARACTER          = '\u0000';
}
```

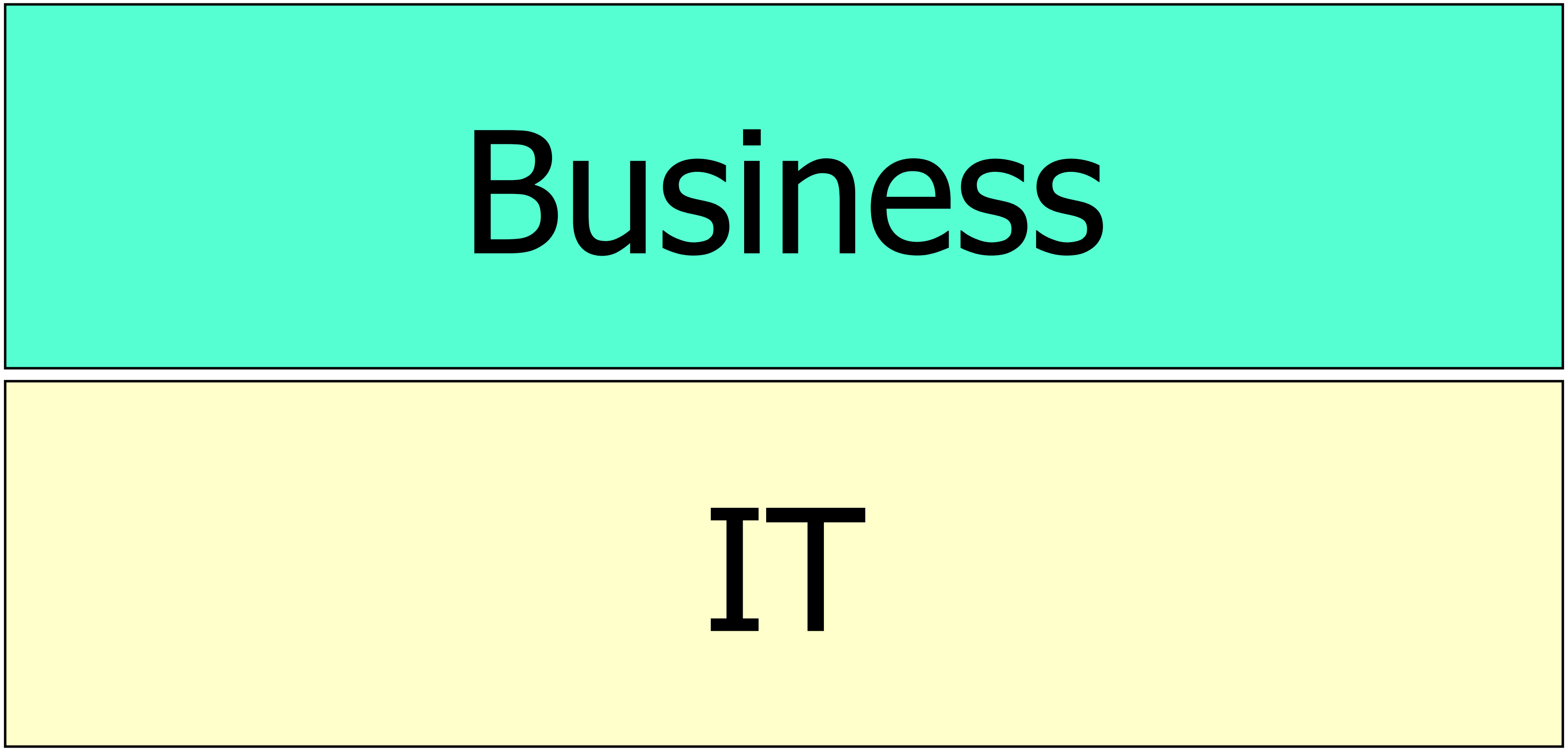
Programmer pour le futur

- Les constantes ne doivent pas être « hardcodées »

- Les évolutions sont gérées uniquement par le changement des valeurs des constantes.

- Solutions possibles : .properties, .xml, JavaBeans …

Separation of Concerns (views)

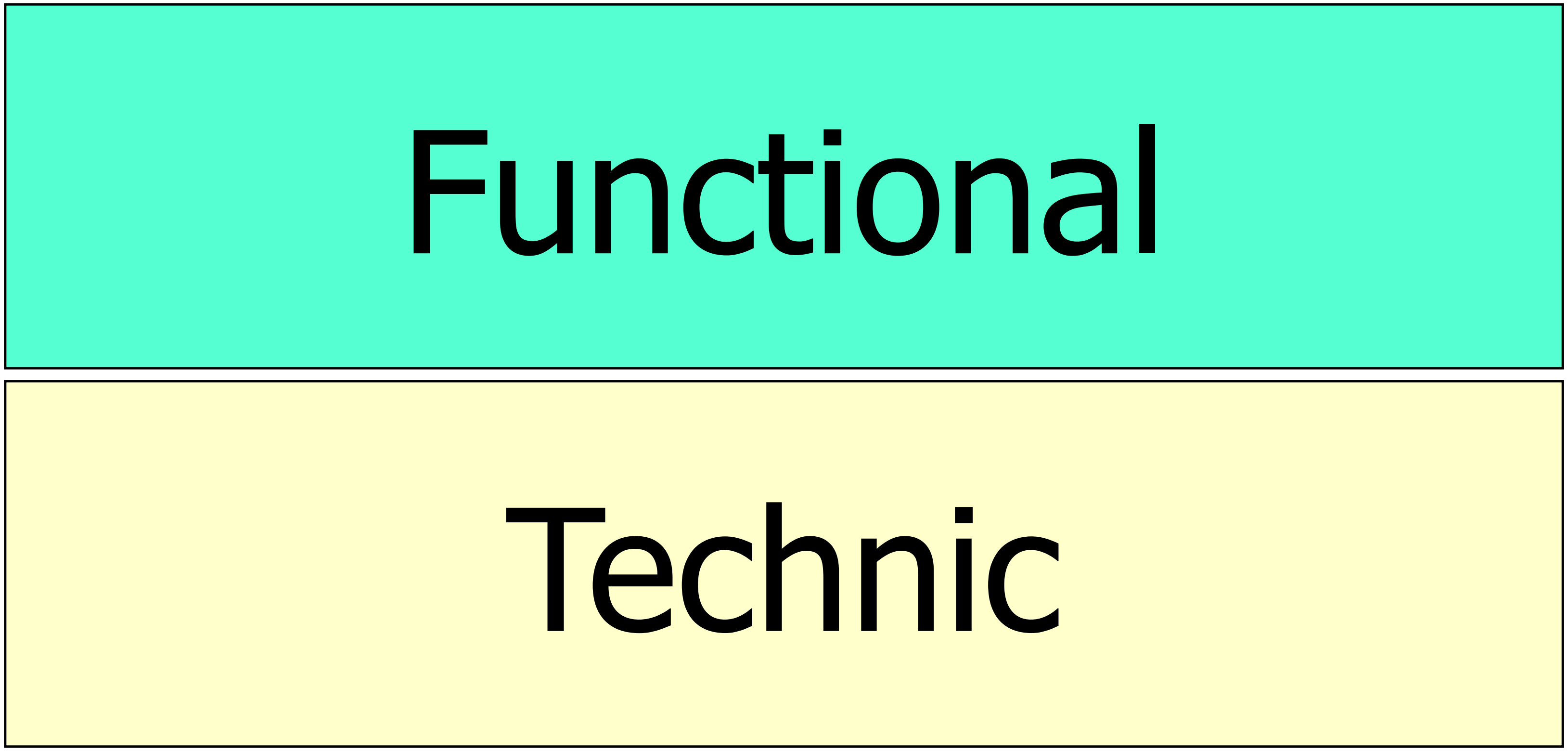


Business

IT

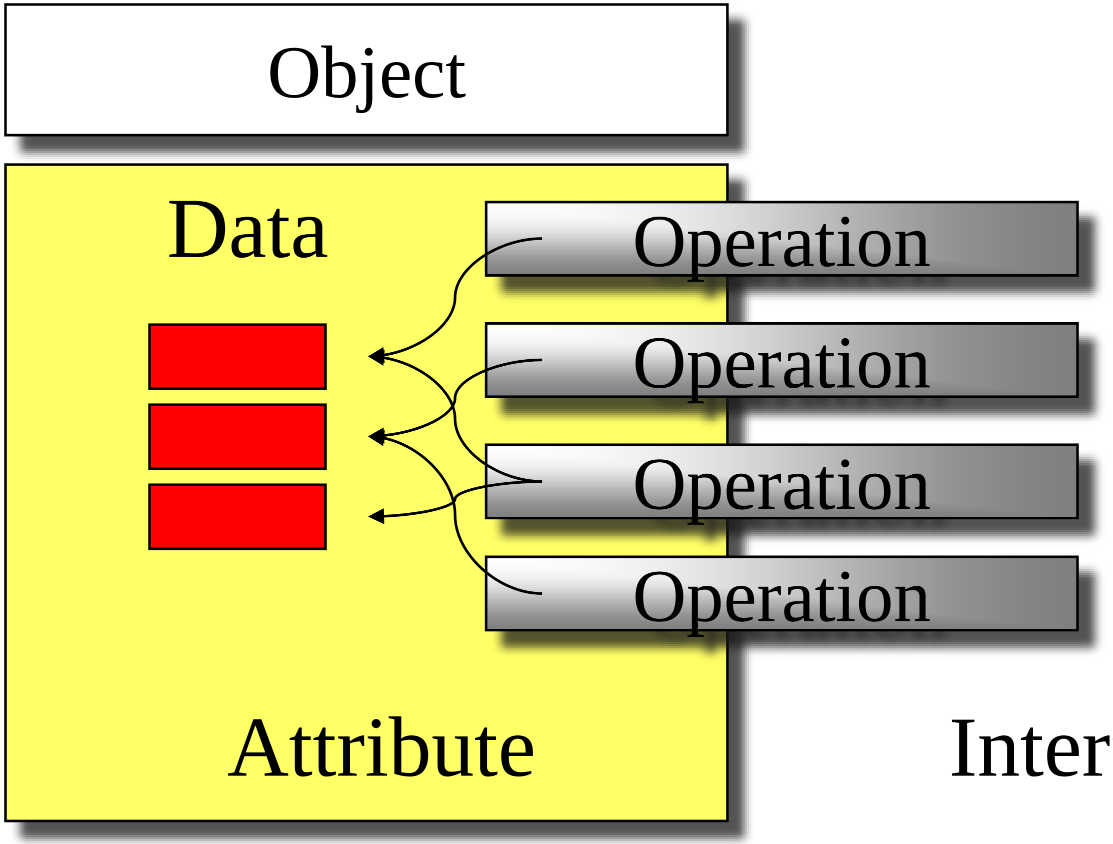
Traduction en cours

Separation of Concerns (views)

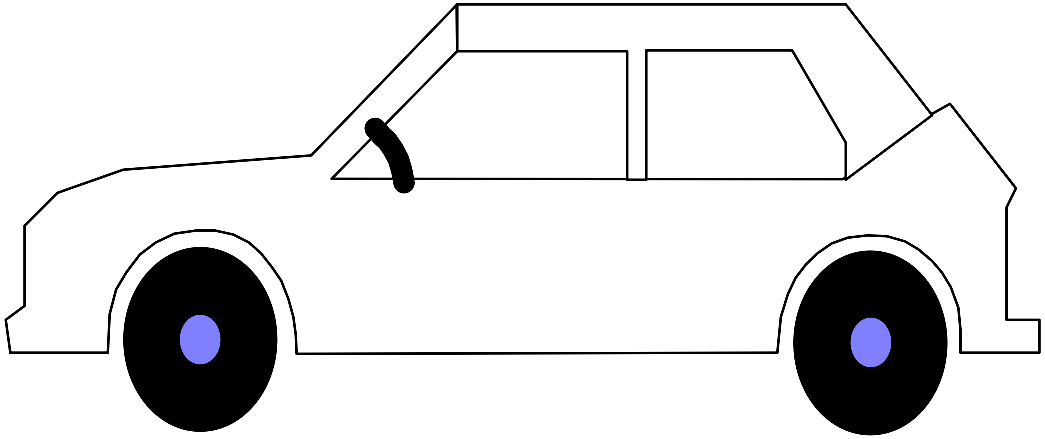

Functional

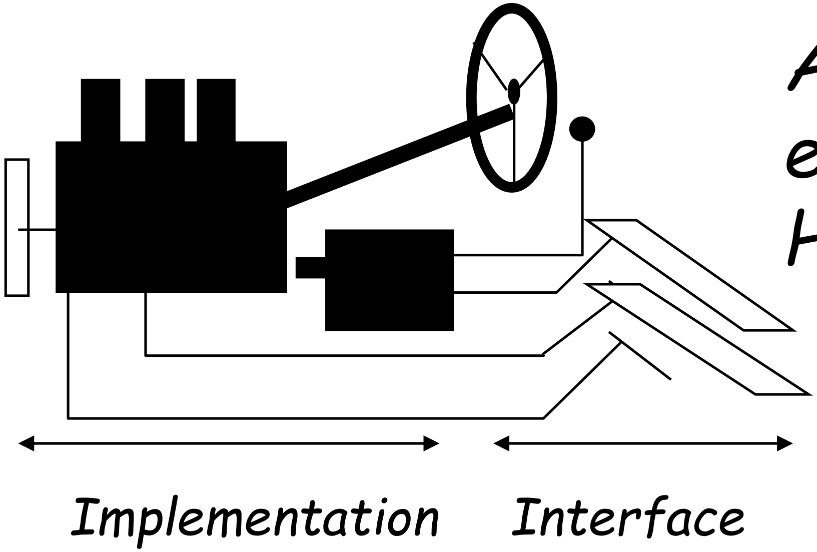
Technic

Object Paradigm



car analogy





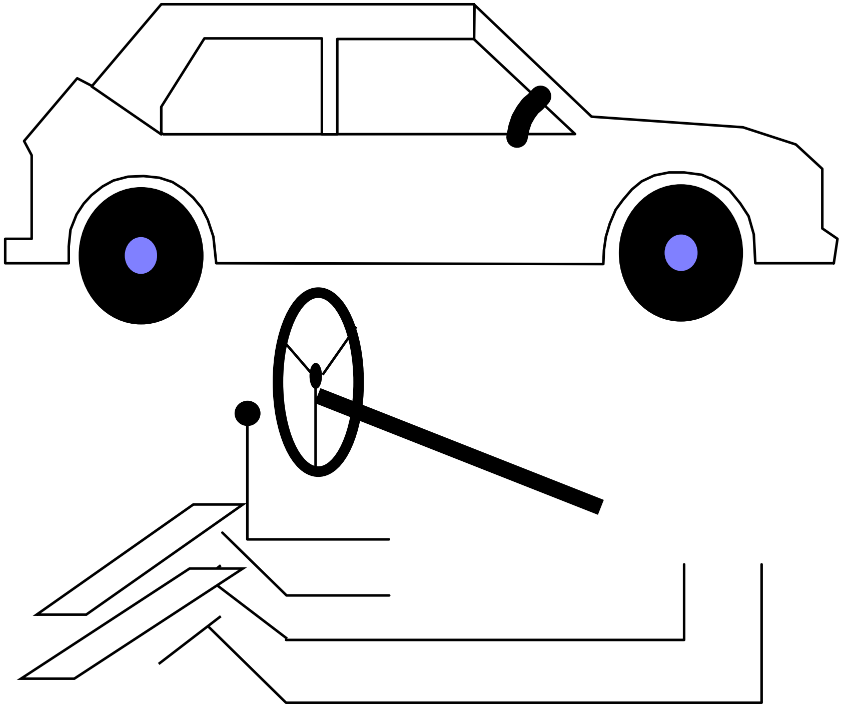
A driver doesn't care of engine's internal working. He only knows the interface

Interface

```
vehicle {

        attributes engine

        interface car {

        start()
        accelerate()
        stop()

        }
}
```

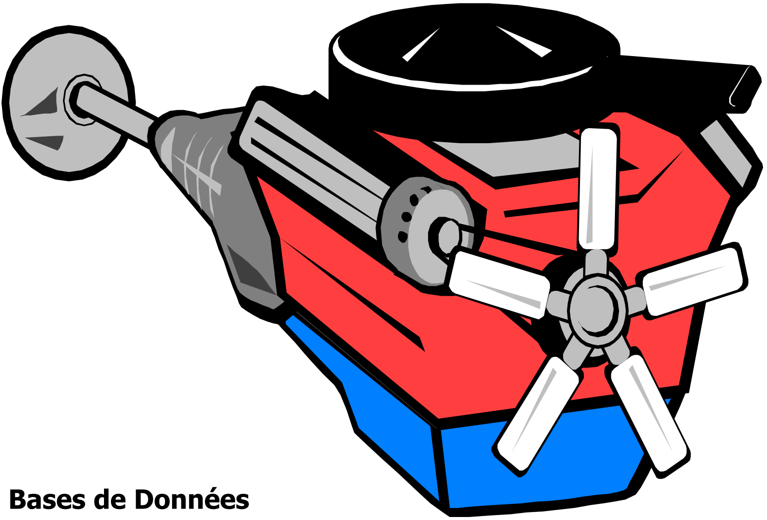
car → start()

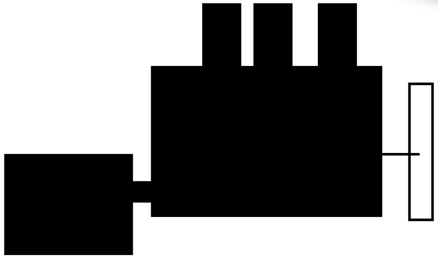
Interface VS implementation

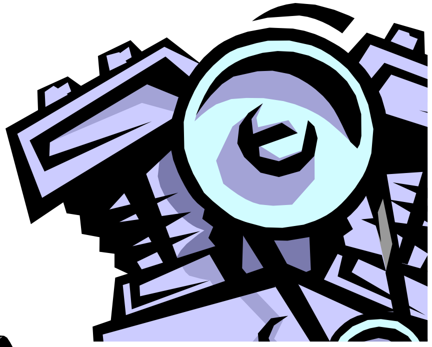


Interface

(specification)







Implementation

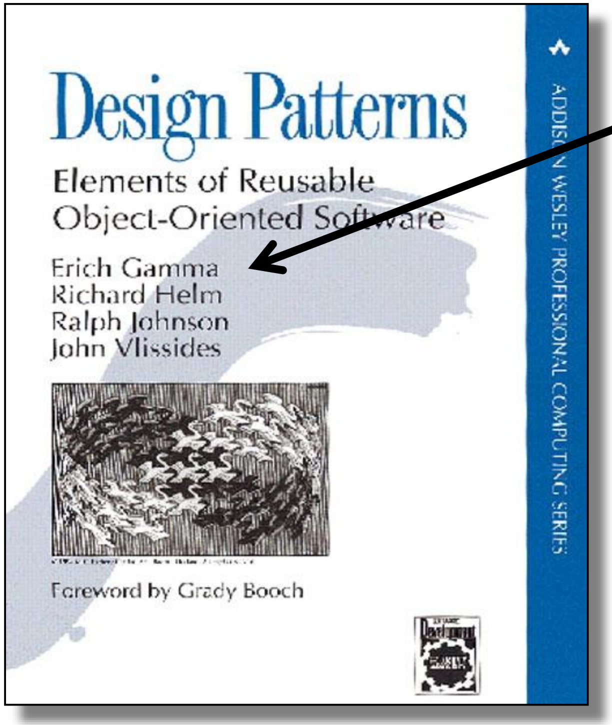
(body)

GOF

“program to an interface, not an implementation”

http://en.wikipedia.org/wiki/Design_Patterns

Moodle



Design Patterns

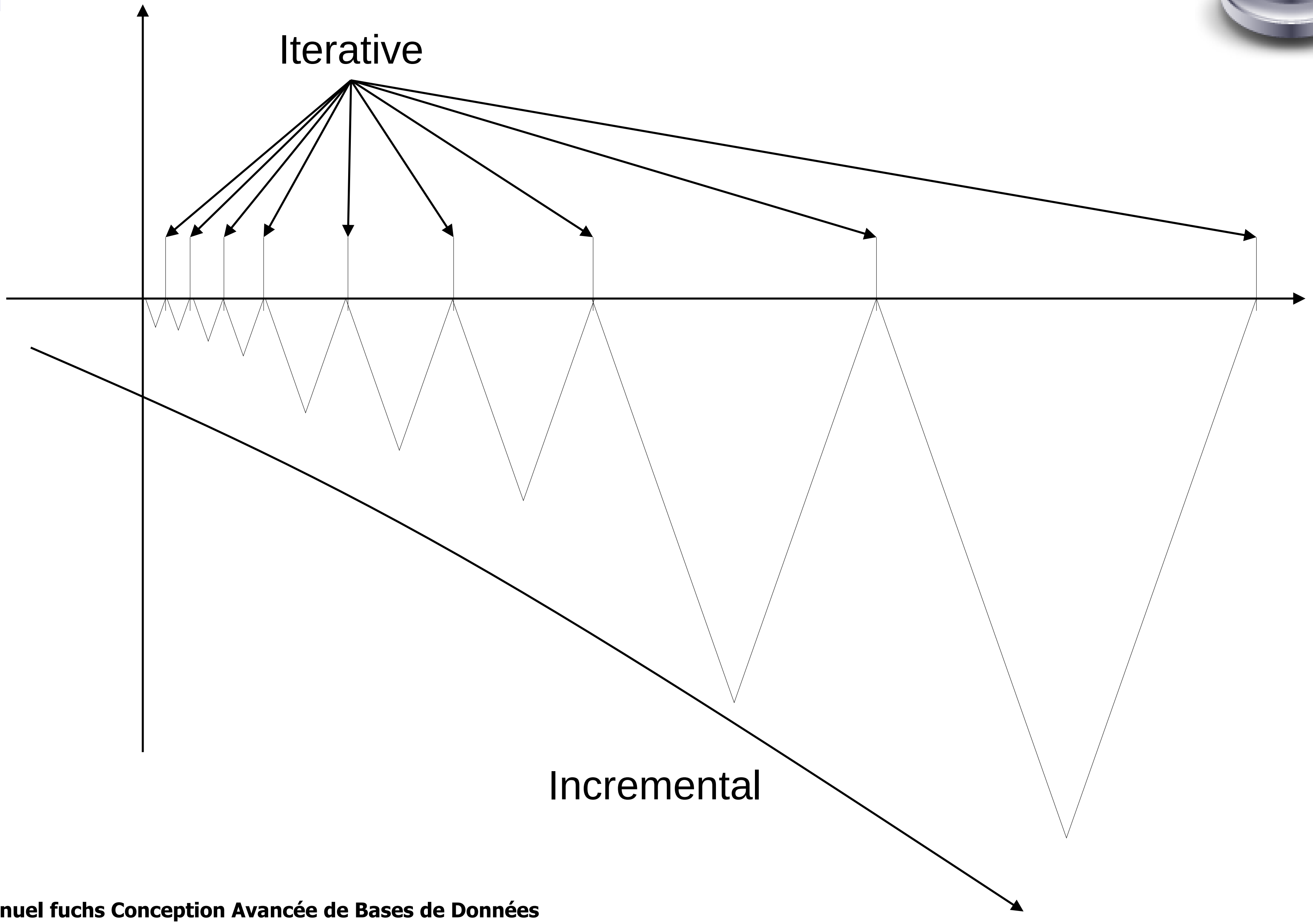
Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

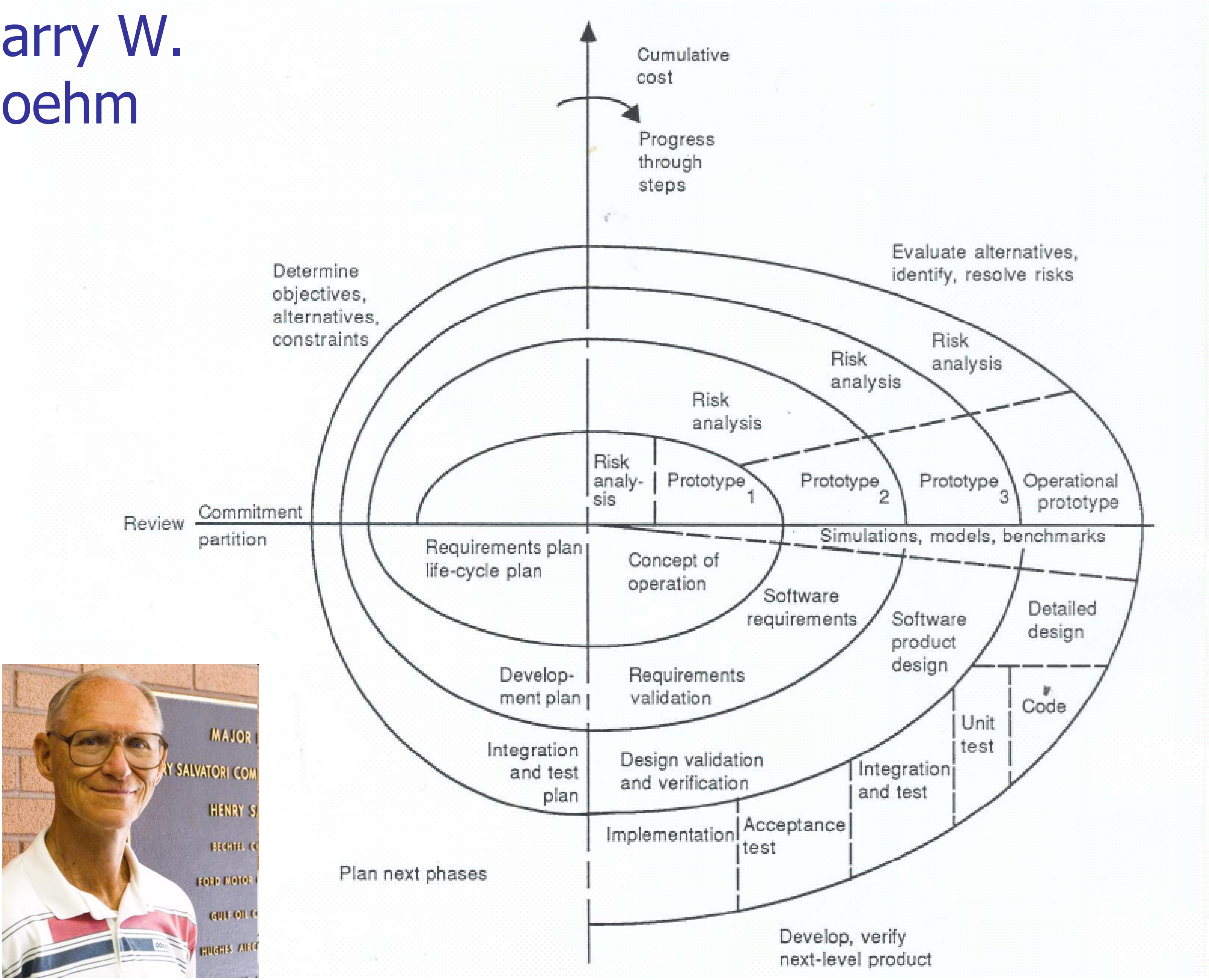
Foreword by Grady Booch

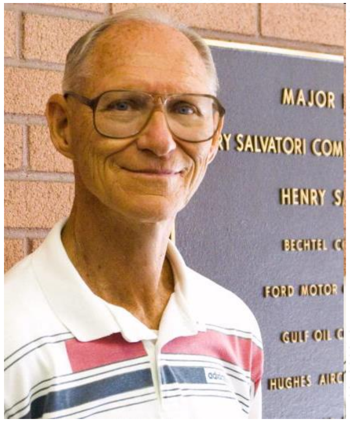
ADDISON WESLEY PROFESSIONAL COMPUTING SERIES

Iterative and Incremental development process

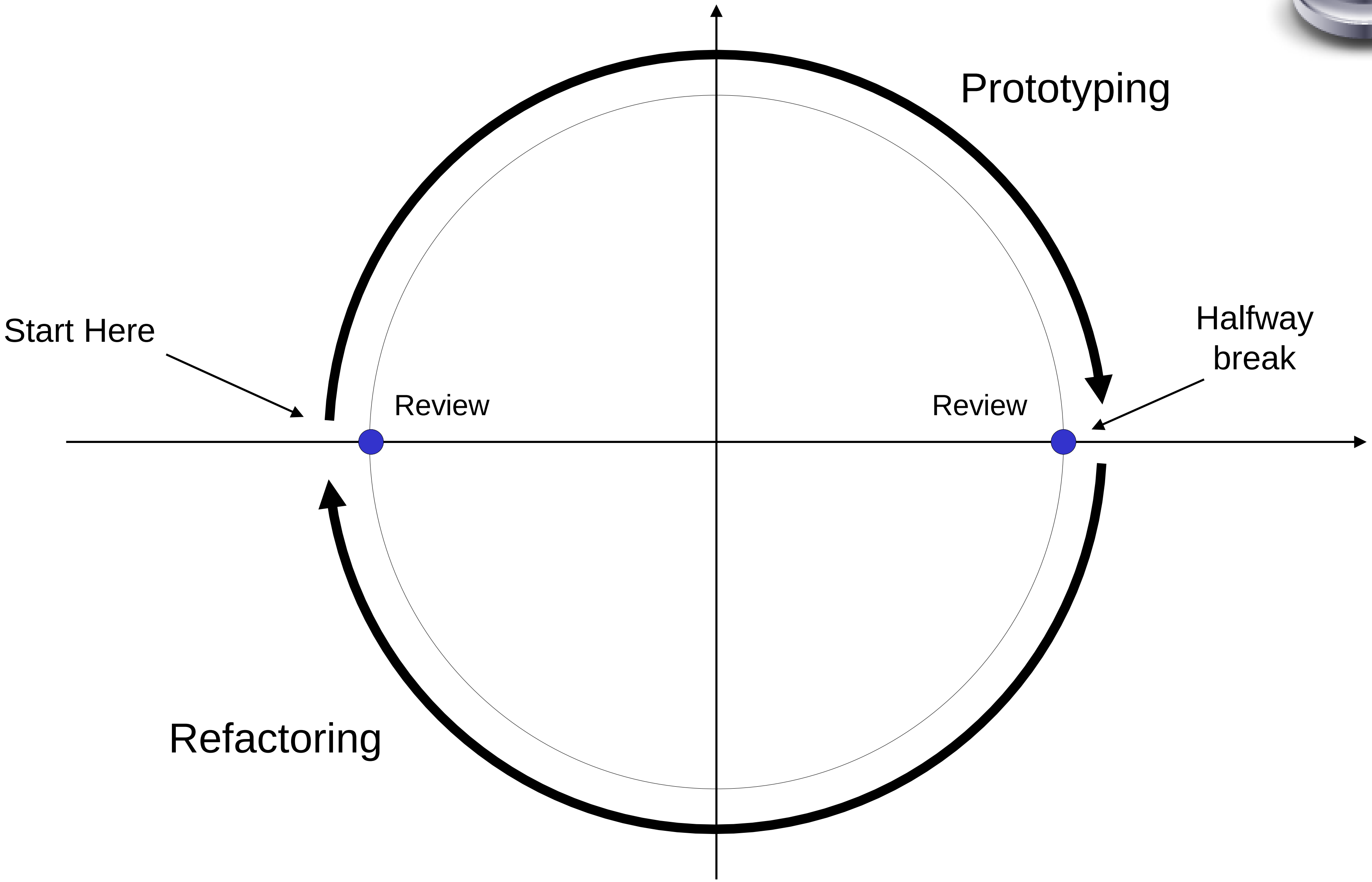


Barry W. Boehm





Use Case Prototyping Cycle


Prototyping
Start Here
Halfway break
Review
Review
Refactoring