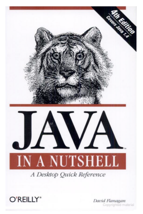
Introduction

Alexandre Bergel
http://bergel.eu
12/03/2018

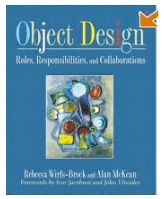
Recommended Texts

Java in Nutshell: 6th edition,

David Flanagan, O’Reilly, 2014.



Object Design - Roles, Responsibilities and Collaborations,



Rebecca Wirfs-Brock, Alan McKean, Addison-Wesley, 2003.

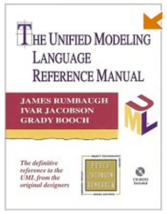
Recommended Texts

Design Patterns: Elements of Reusable Object-Oriented Software,

Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Addison Wesley, Reading, Mass., 1995.

The Unified Modeling Language Reference Manual,

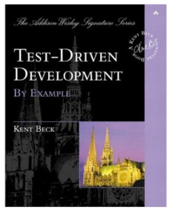
James Rumbaugh, Ivar Jacobson, Grady Booch, Addison-Wesley, 1999



Recommended Texts

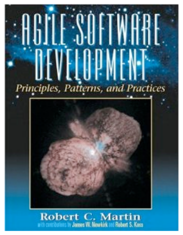
Test-Driven Development,

Kent Beck, Addison-Wesley Professional, 2002.



Agile Software Development, Principles, Patterns, and Practices

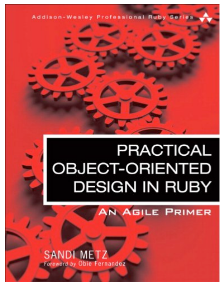
Robert C. Martin, Prentice Hall, 2002



Recommended Texts

Practical object-oriented design in ruby,

Sandi Metz, Addison-Wesley Professional, 2012.



Java Precisely

Peter Sestoft, MIT Press, 2016

Outline

1. Programming
2. Programming is complex!
3. Object-Oriented Design
4. Java

What constitutes programming?

- ★Understanding requirements
- ★Design
- ★Testing
- ★Debugging
- ★Developing data structures and algorithms
- ★User interface design
- ★Profiling and optimization
- ★Reading code
- ★Enforcing coding standards
- ★...

Question

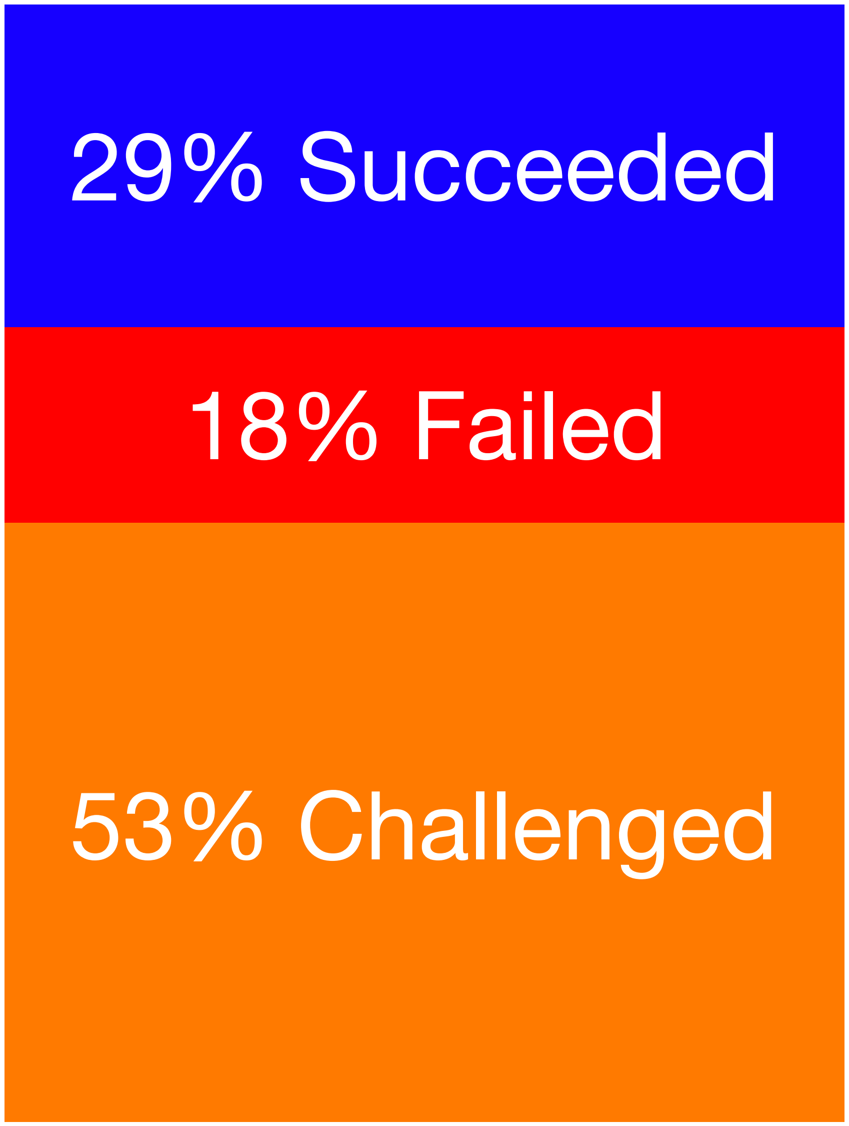
What is the easy and hard part of programming?

Question

What is the easy and hard part of programming?

The easy part: telling a computer what it should do
The hard part: telling other programmers what a program does

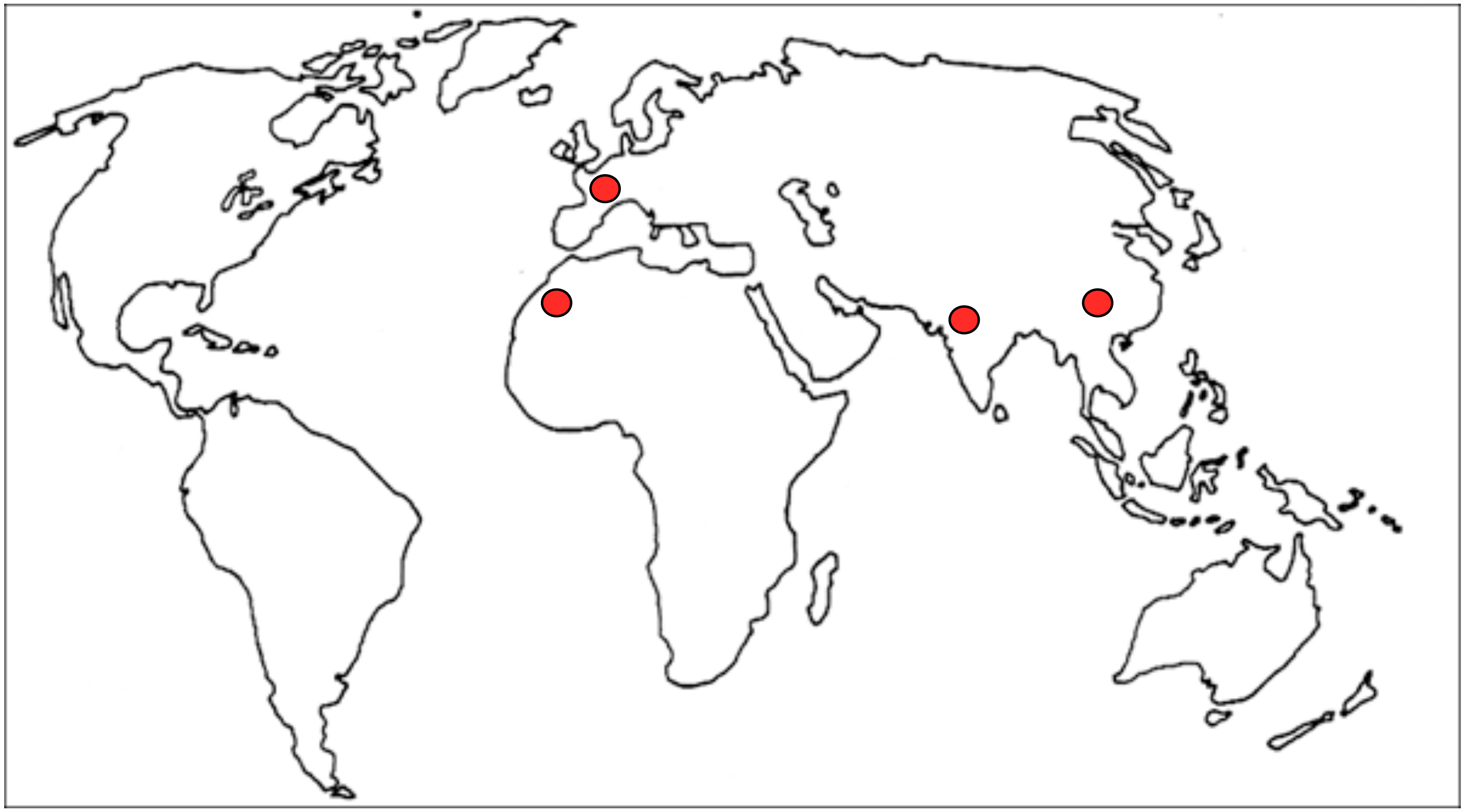
Software is complex



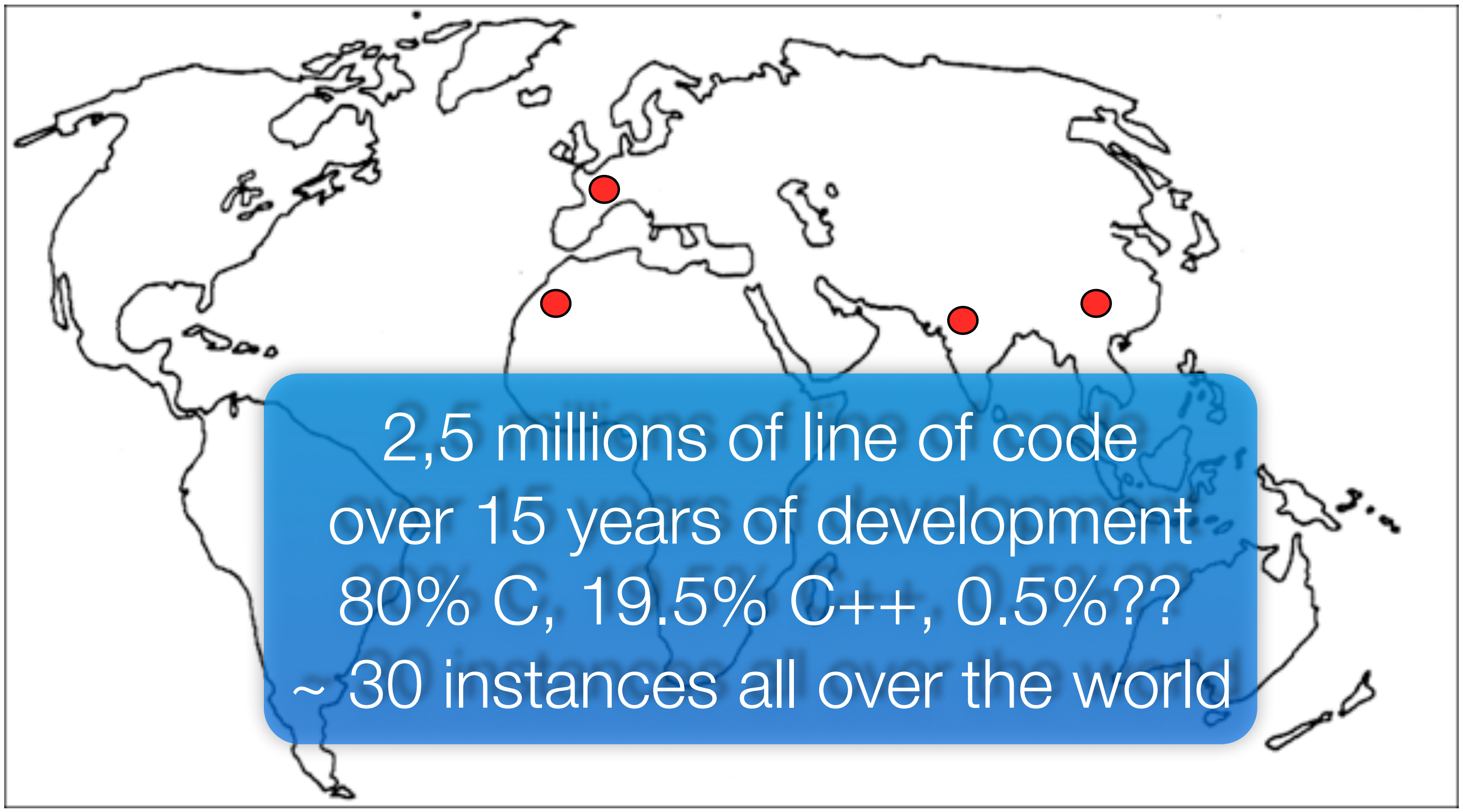
The Standish Group, 2004

Let’s study some real world examples

Construction sites for an European truck maker

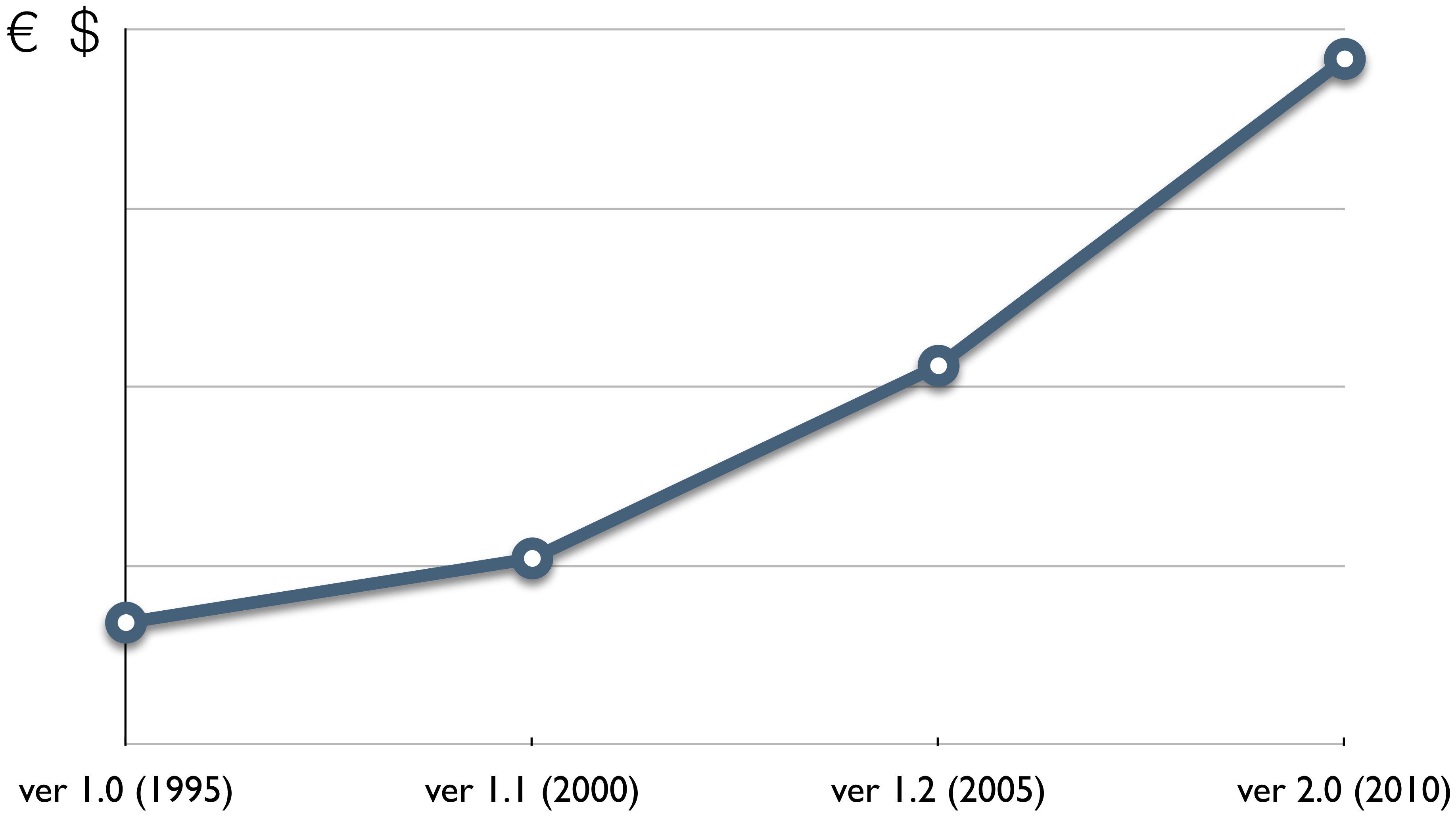


Construction sites for an European truck maker

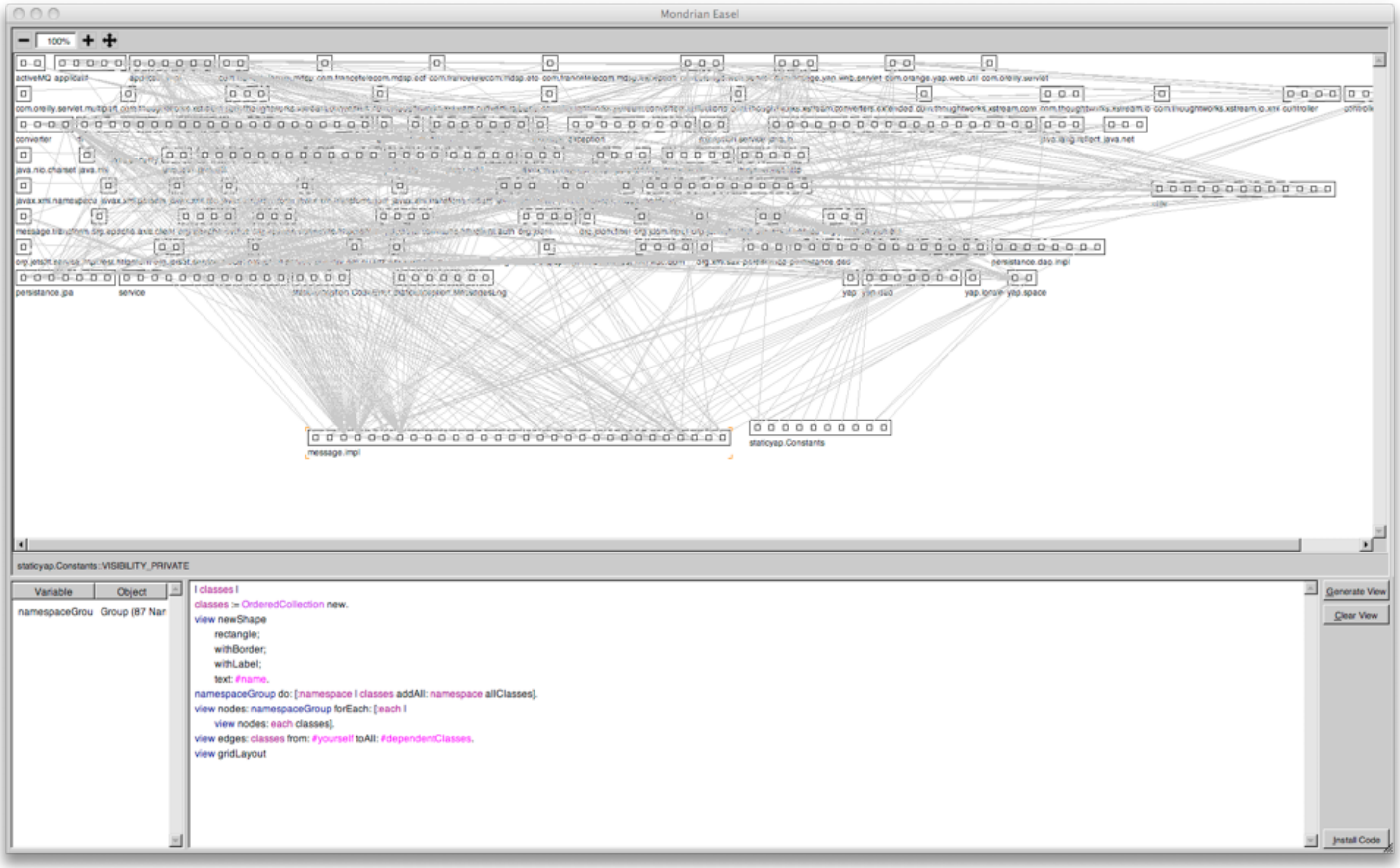


2,5 millions of line of code
over 15 years of development
80% C, 19.5% C++, 0.5%??
~ 30 instances all over the world

Cost of feature addition



Large software in a French telecom company



staticyap.Constants::VISIBILITY_PRIVATE

Variable	Object
namespaceGrou	Group (87 Nar

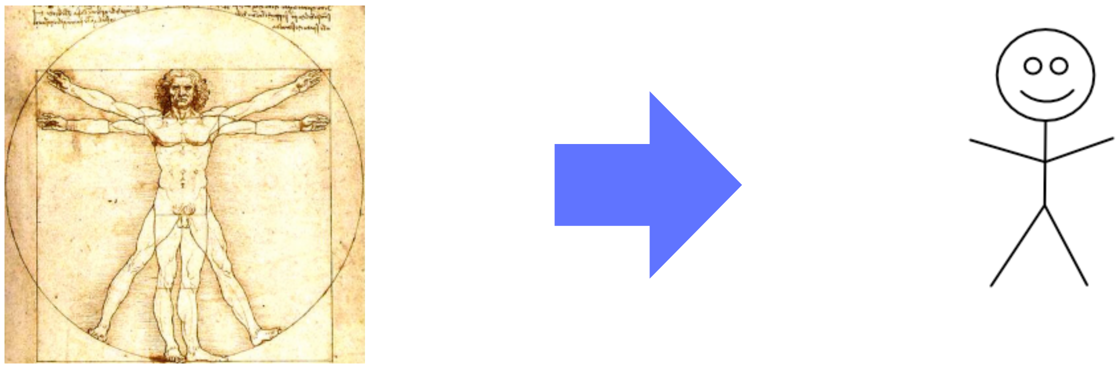
```
| classes |
classes := OrderedCollection new.
view newShape
    rectangle;
    withBorder;
    withLabel;
    text: #name.
namespaceGroup do: [:namespace | classes addAll: namespace allClasses].
view nodes: namespaceGroup forEach: [:each |
    view nodes: each classes].
view edges: classes from: #yourself toAll: #dependentClasses.
view gridLayout
```

~100 packages
~ 500 classes

Typical large scale long living systems

- Large
  - thousands of classes
  - hundreds of packages
- Undocumented - knowledge loss
- Lack of structure overview (layers, cycles, core)
- Possibly written in ADA or Cobol
- Multi developers
- Multi years development

How can we simplify programming?



Key insights

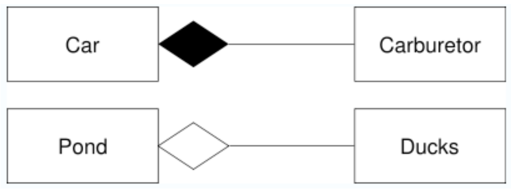
Real programs changes!

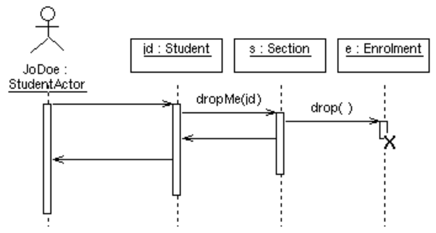
Development is incremental

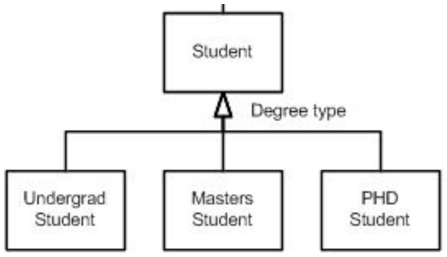
Design is iterative

What is Object-Oriented Programming?

Encapsulation	Abstraction & Information Hiding
Composition	Nested Objects
Distribution of Responsibility	Separation of concerns (e.g., HTML, CSS, JavaScript)
Message Passing	Delegating responsibility
Inheritance	Conceptual hierarchy, polymorphism and reuse







A procedural design

Problem: compute the total area of a set of geometric shapes

```
public static long sumShapes(Shape[] shapes) {
    long sum = 0;
    for (int i=0; i<shapes.length; i++) {
        if (shapes[i] instanceof Rectangle) {
            Rectangle r = (Rectangle)shapes[i];
            sum += (r.width * r.height);
            break;
        }
        if (shapes[i] instanceof Circle) {
            Circle r = (Circle)shapes[i];
            sum += (Math.PI * r.radius * r.radius);
            break;
        }
        // more cases
    }
    return sum;
}
```

An object-oriented approach

A typical object-oriented solution:

```
public static long sumShapes(Shape[] shapes) {
    long sum = 0;
    for (Shape s : shapes) {
        sum += s.area();
    }
    return sum;
}
```

What are the advantages and disadvantages of the two solutions?

Object-oriented design

A proper Object-oriented design is

- easy to understand
- easy to extend

However, getting such a design is not trivial

Unit testing, Design Patterns, Refactoring are commonly employed to improve software quality

What is Java?

Java is a platform for application development

Developed at Sun Microsystems in 1995

Java comprises

- a compiler
- a running execution support
- a programming environment (usually provided by a tierce)

Some features of Java

Object oriented

Both interpreted and natively compiled

- Key for portability. Java runs on many many devices

Relatively secure

Multi-threaded

High-performance

- Intense research on making Java works on multicore CPU

Some characteristics of Java

Cross platform, most of the time

Basic principles are easy *understandable*

- but Java remains a complicated language

Java guarantees a form of *safety*

- it cannot easily crash your machine
- automatic memory management garbage collector

Widely used in industry

If you understand Java, you will probably understand *C#, PHP, Ruby,* and many more

Why Java?

Special characteristics

- Resembles C++ minus the complexity
- Clean integration of many features
- Dynamically loaded classes
- Large, standard class library

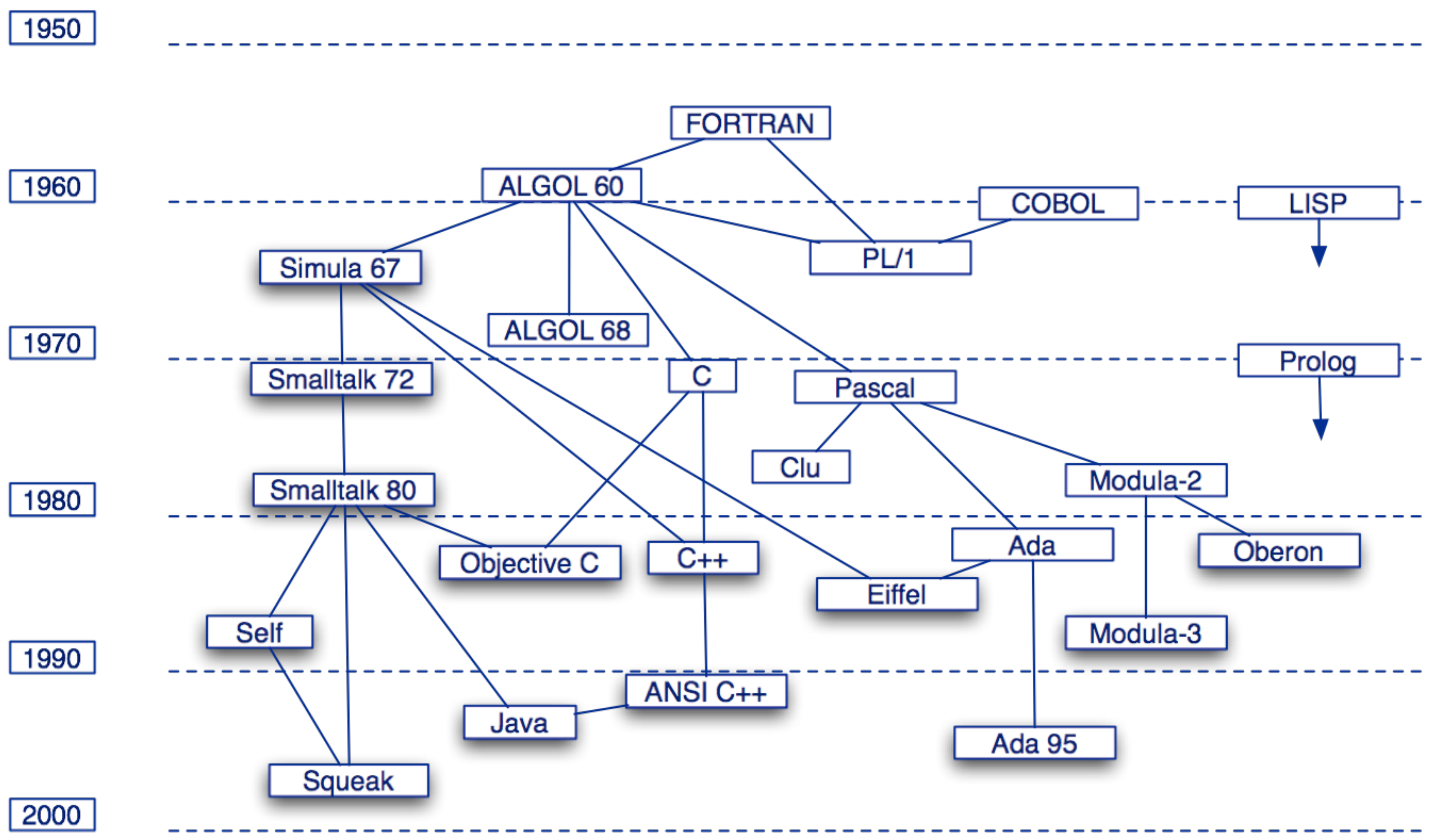
Why Java?

Simple Object Model

- “Almost everything is an object”
- No pointers
- Garbage collection
- Single inheritance; multiple subtyping
- Static and dynamic type-checking

Few innovations, but reasonably clean, simple and usable

History


1950
1960
1970
1980
1990
2000
FORTRAN
ALGOL 60
COBOL
LISP
PL/1
Simula 67
ALGOL 68
Smalltalk 72
C
Pascal
Prolog
Clu
Modula-2
Smalltalk 80
Objective C
C++
Ada
Oberon
Eiffel
Self
Modula-3
ANSI C++
Java
Ada 95
Squeak

What you should know!

What is meant by “separation of concerns” ?

Why do real programs change?

How does object-oriented programming support incremental development?

Can you answer these questions?

What are good and bad uses of inheritance?

What does it mean to “violate encapsulation”?

Why is strong coupling bad for system evolution?

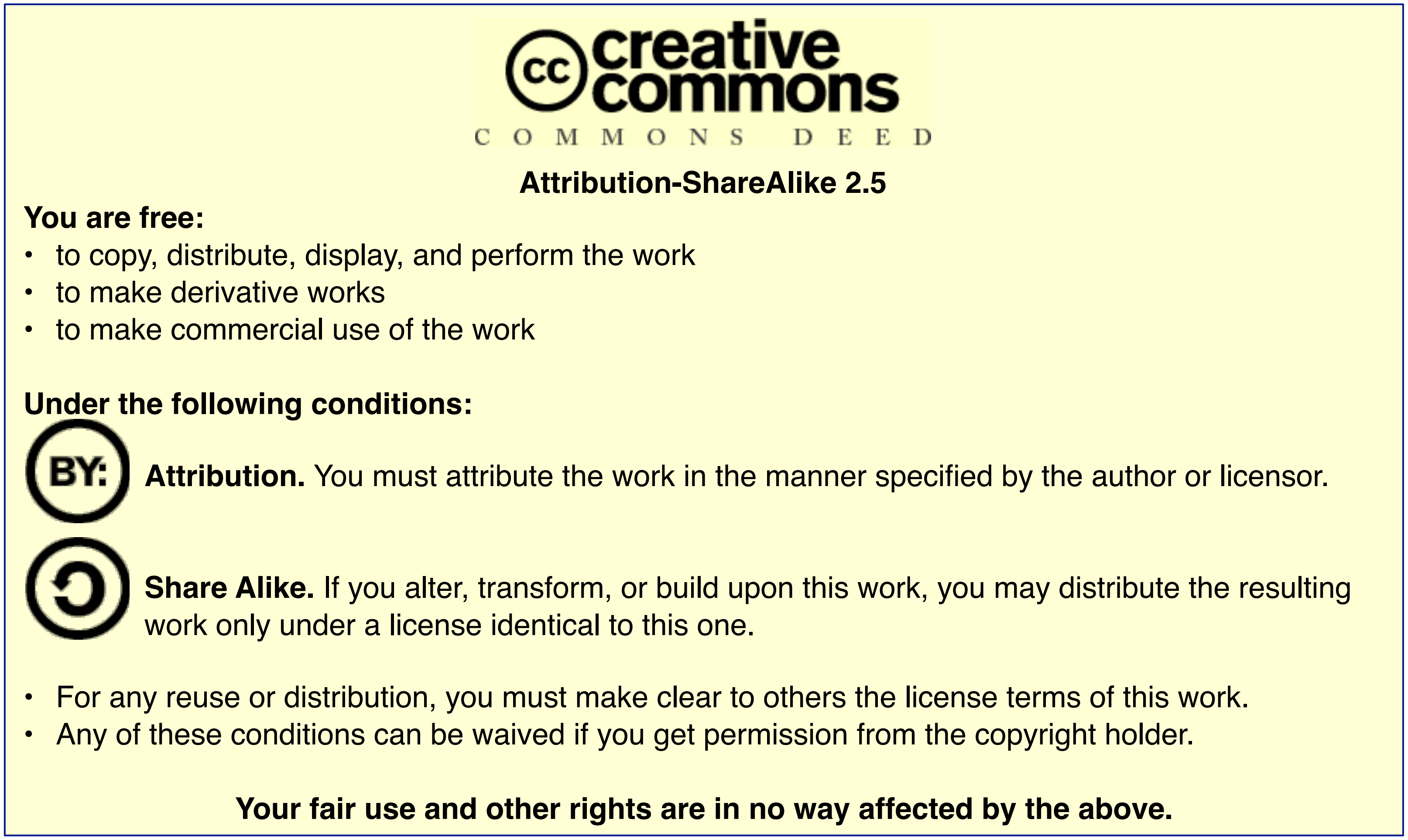
How can you transform requirements into tests?

How would you eliminate duplicated code?

When is the right time to refactor your code?

License

http://creativecommons.org/licenses/by-sa/2.5



creative commons

COMMONS DEED

Attribution-ShareAlike 2.5

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.