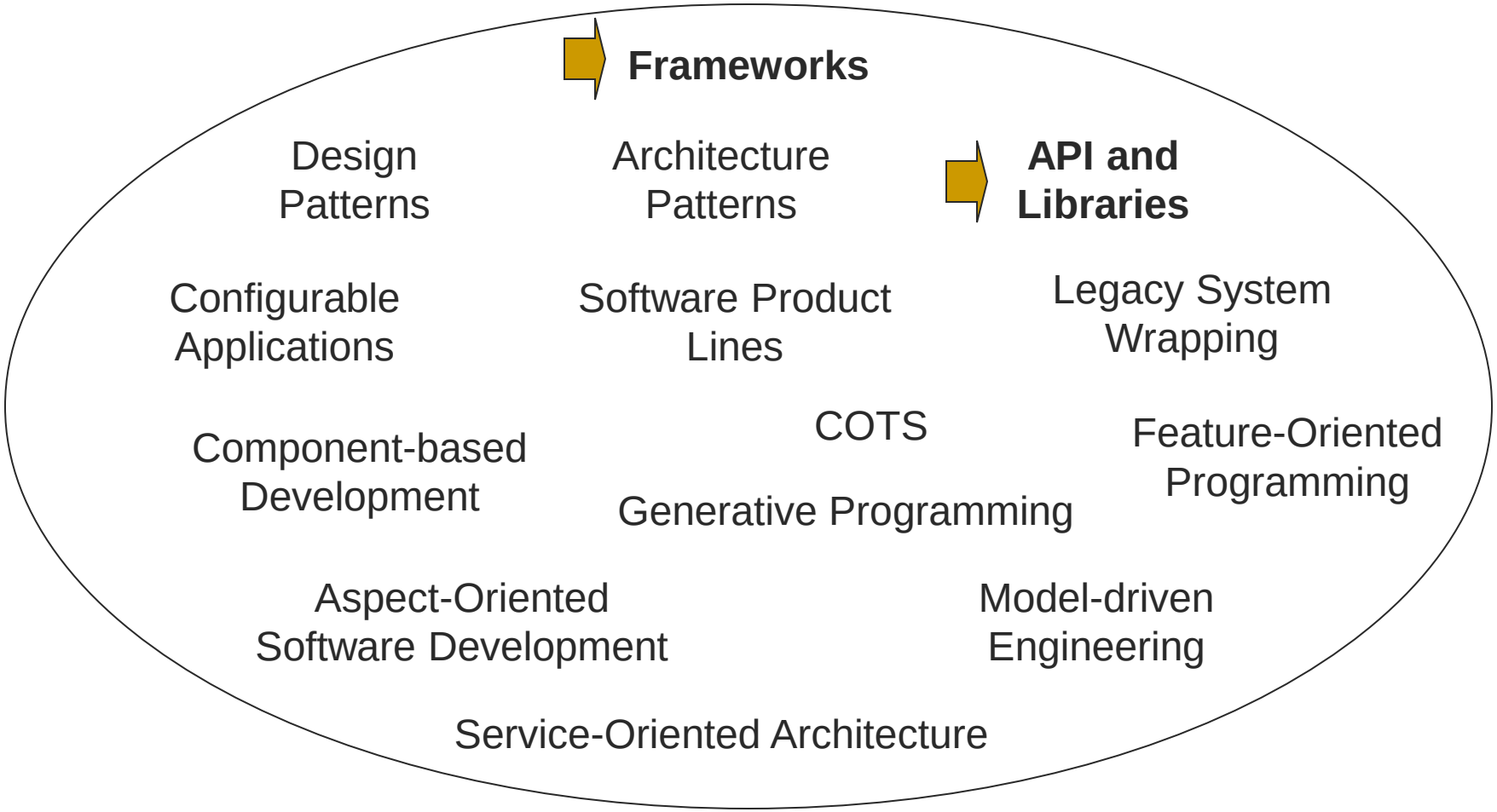
Reuse in Object Oriented Programming

Eduardo Figueiredo

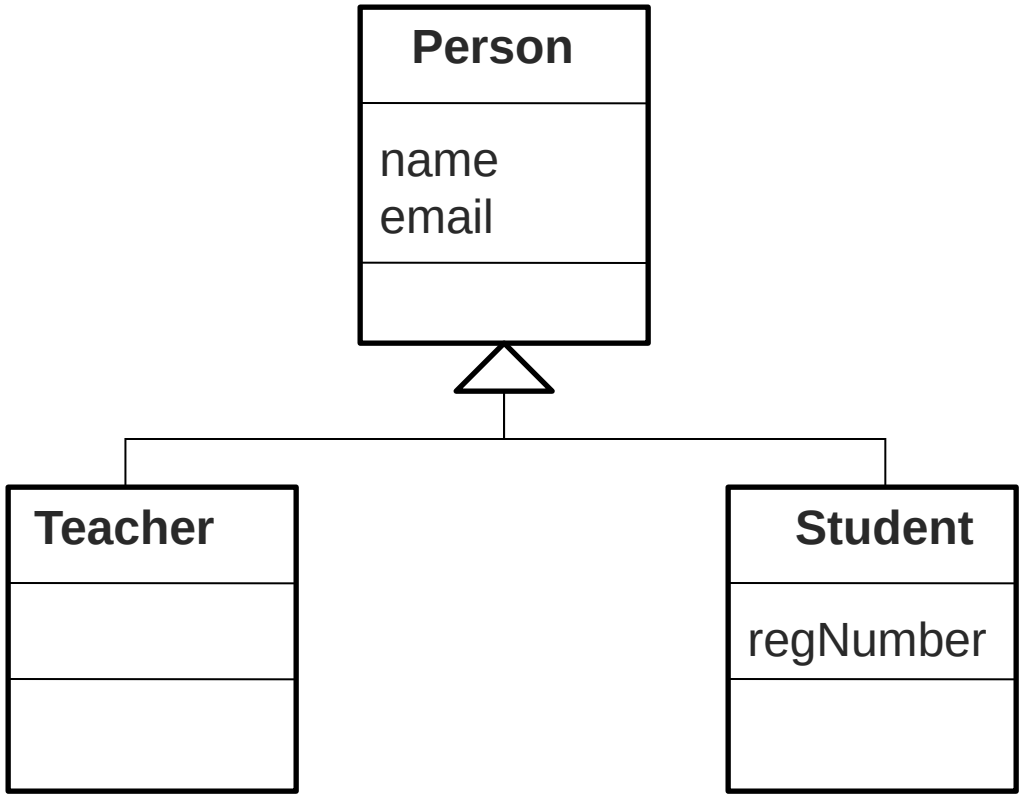
http://www.dcc.ufmg.br/~figueiredo

The Reuse Landscape

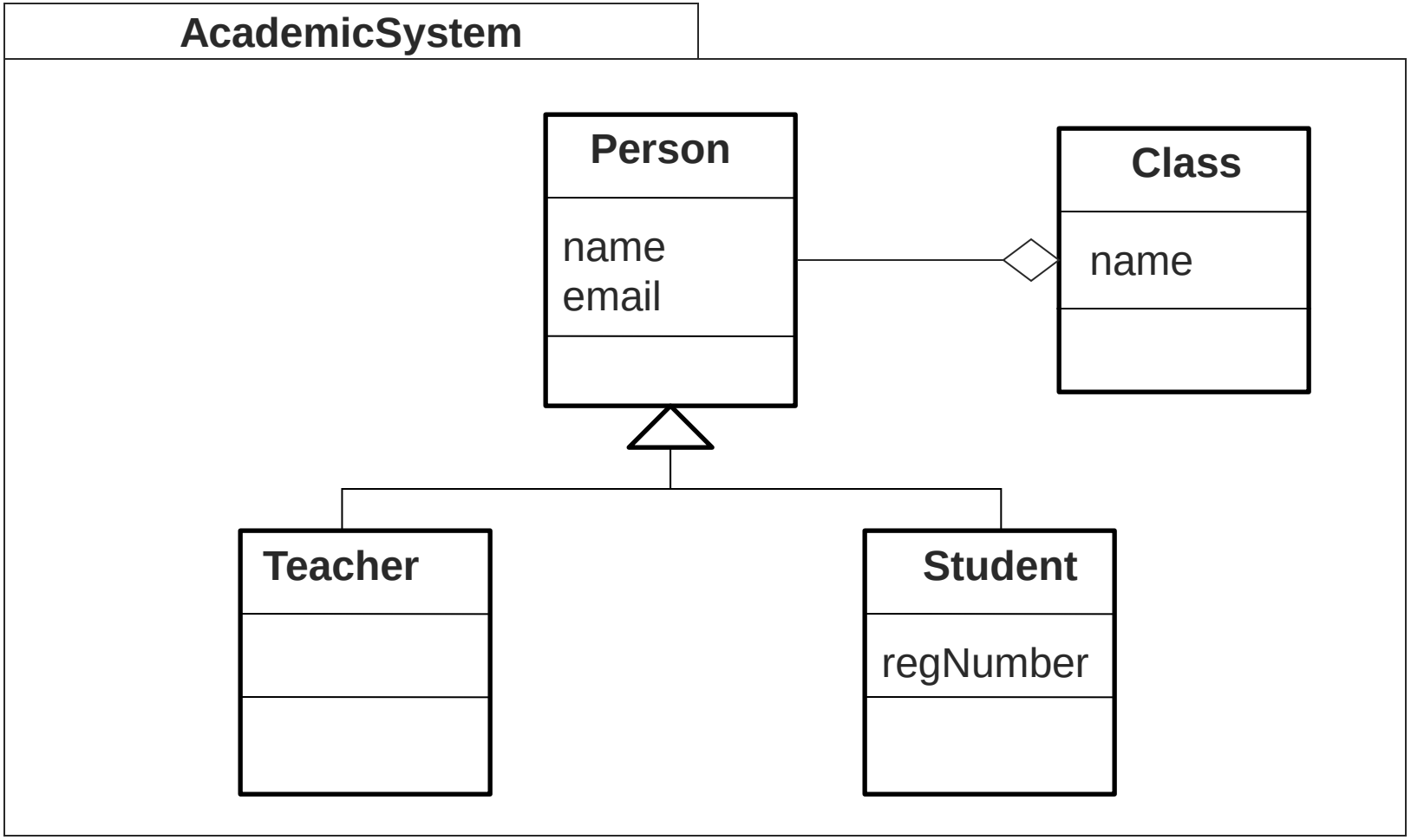


Reuse of Classes

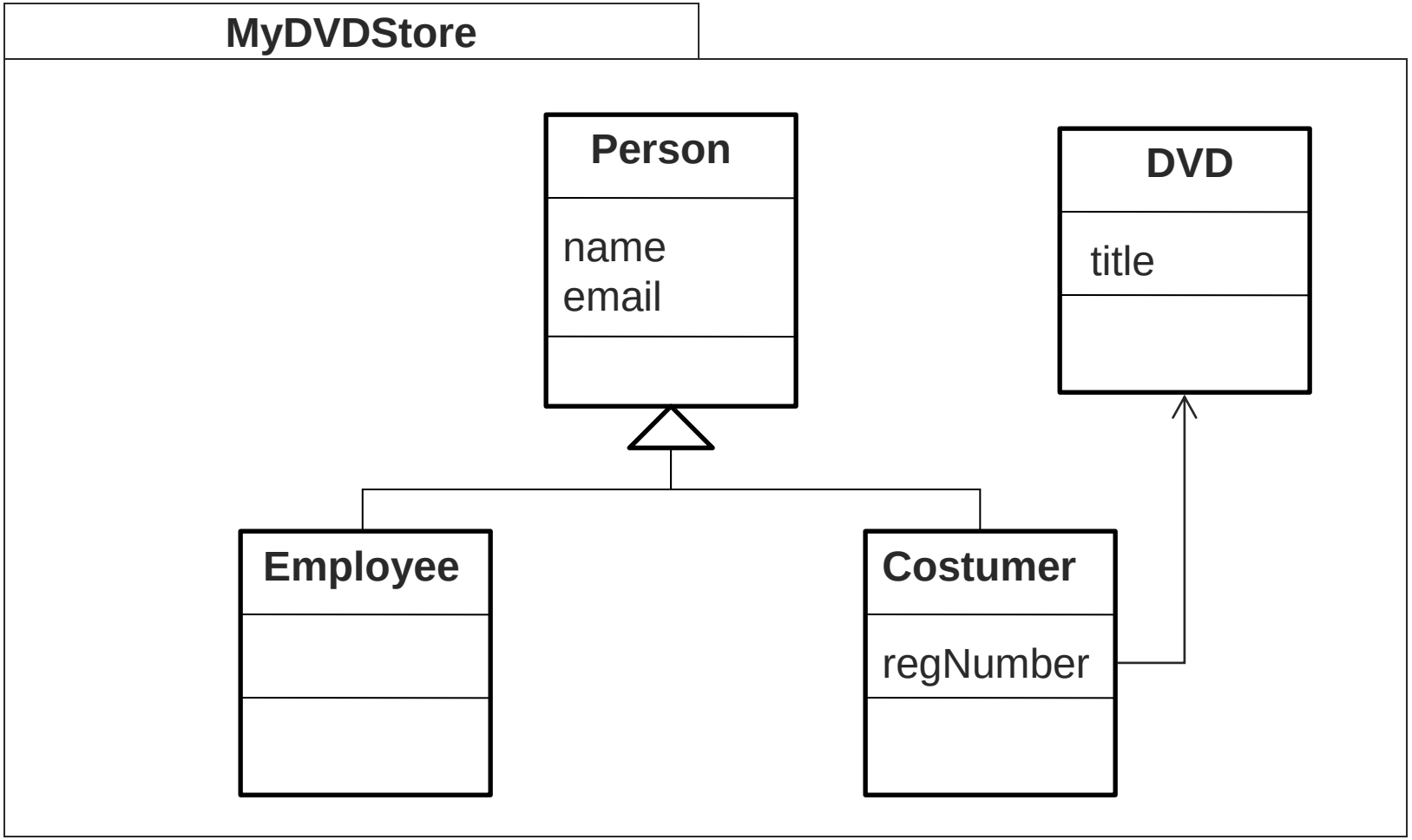
Let's Consider 3 Classes...



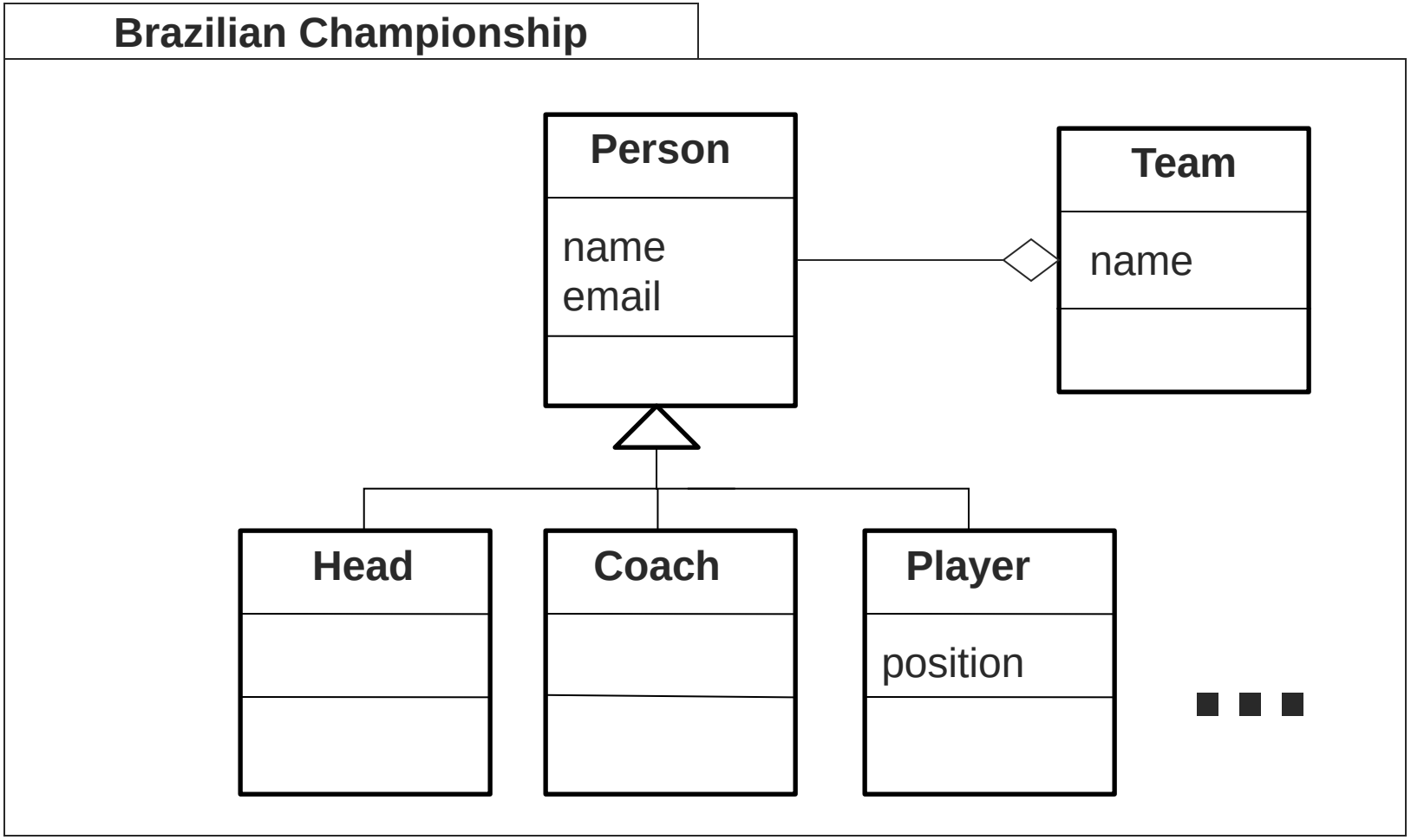
… in a Academic System



... or in a DVD Rental System



... or in a Football Game



Library

Library (API)

- Application Programming Interface (API)
- Libraries (or API) implement services used by several kinds of programs
  - It is a common way of software reuse
- API makes available common functions
  - To convert data from common formats (e.g., string to integer)
  - To access resources, files, DB, etc.
  - Abstract data types, such as queue, stack

Example of Java API Use

```
import java.util.Vector;

public class Customer {

  String name;
  Vector phoneNumbers = new Vector();

  void removePhoneNumber(String c){
    phoneNumbers.removeElement(c);
  }

  void addPhoneNumber(String c){
    phoneNumbers.addElement(c);
  }

  ...
}
```

Import a Library Class

```
import java.util.Vector;

public class Customer {

  String name;
  Vector phoneNumbers = new Vector();

  void removePhoneNumber(String c){
    phoneNumbers.removeElement(c);
  }

  void addPhoneNumber(String c){
    phoneNumbers.addElement(c);
  }

  ...
}
```

The Vector class is now part of the system.

Instantiate a Library Object

```
import java.util.Vector;

public class Customer {

   String name;
   Vector phoneNumbers = new Vector();

   void removePhoneNumber(String c){
     phoneNumbers.removeElement(c);
   }

   void addPhoneNumber(String c){
     phoneNumbers.addElement(c);
   }

   ...
}
```

A Vector instance can be created in the same way as other objects in the system.

Access to Library Funcions

```
import java.util.Vector;

public class Customer {

   String name;
   Vector phoneNumbers = new Vector();

   void removePhoneNumber(String c){
      phoneNumbers.removeElement(c);
   }

   void addPhoneNumber(String c){
      phoneNumbers.addElement(c);
   }

   ...
}
```

Methods like removeElement and addElement can be called, although they are implemented in the API.

Productiveness and Reliability

- Developers do not need to implement everything
  - Many functionalities are already available in the library
- Libraries are extensively tested by several users
  - Even in extreme and unusual situations

Main Drawbacks

- It is hard to change a library if we need specific behavior
  - Best performance
  - More robustness
- It takes time to learn and to find what we need in the library

Frameworks

Motivation

- Objects and functions (in libraries) are often too fine grained and specific
  - It may be useful to reuse larger entities
- Framework is a set of classes and interfaces to form the general structure of an application

Classification of Frameworks

- Infrastructure Frameworks
  - Support the creation of infrastructure for systems, such as compiling environment
- Integration Frameworks
  - Support exchanges of messages among components (communication)
- Application Frameworks
  - Support development of new applications with common characteristics, such as Web applications

Main Properties of Framework

- Inversion of control
  - Unlike in libraries or applications, the program flow of control is dictated by the framework
- Extensibility
  - A framework can be extended by the user
- Non-modifiable framework code
  - The framework code should not be modified
- A framework has a default behavior

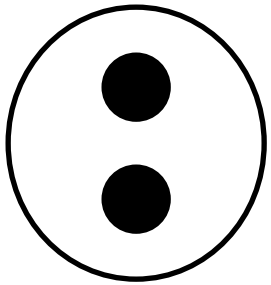
Framework Extension

- Framework is an incomplete system
  - It is composed of classes and interfaces
  - Several applications can be instantiated
- A system is implemented by adding classes to fill in the gaps
  - Include new concrete classes
  - Override methods
  - Include configuration files, such as XML

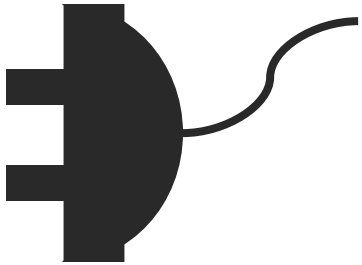
Frozen Spots and *Hot Spots*

- Frameworks have two main parts, called frozen spots and hot spots
- Frozen spots define the general structure of a framework
  - They cannot be changed in applications (they are frozen)
- Hot spots define places which can be extended by programmers in applications

Representation of Hot Spots

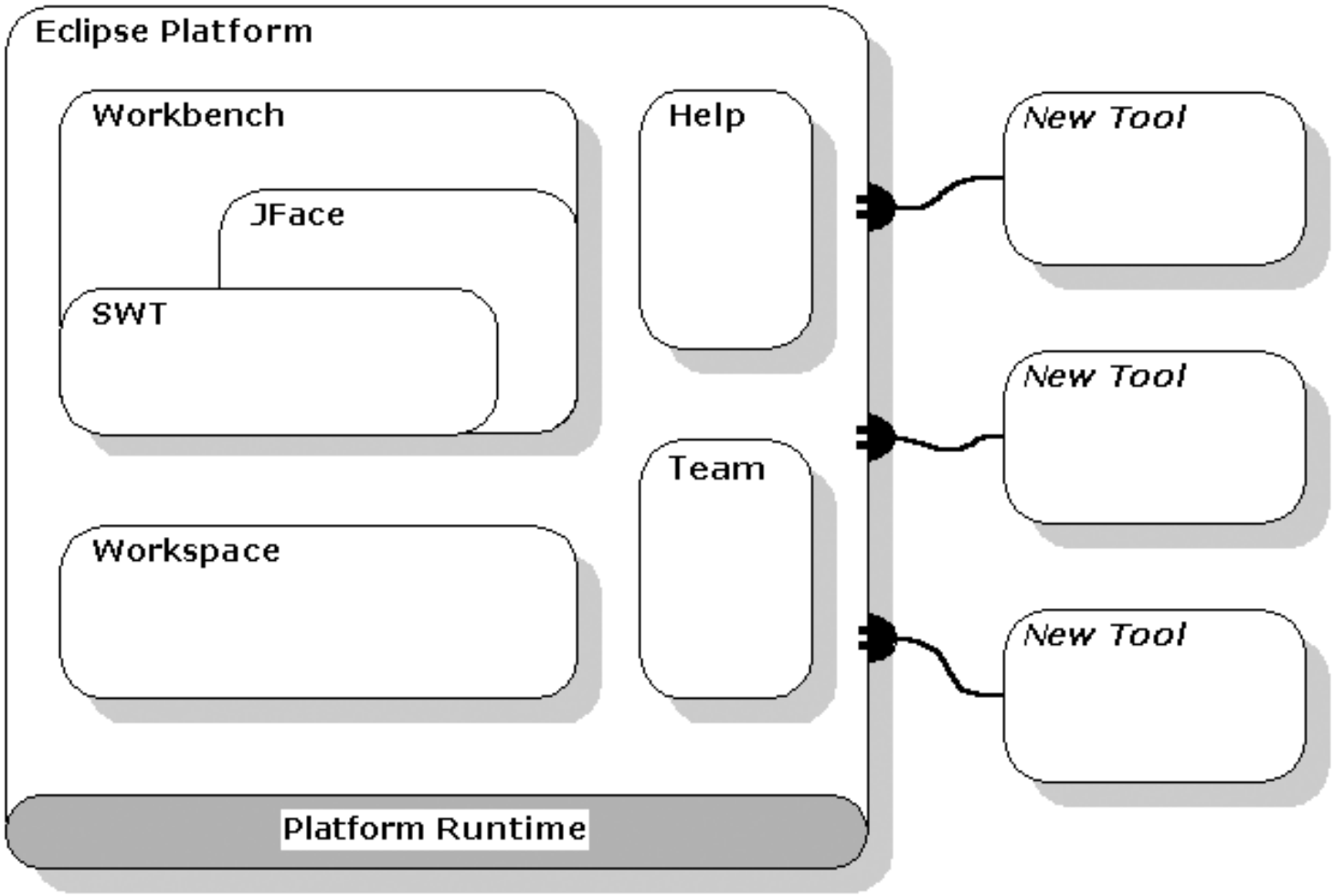


Framework
Extension Point



Application
Extension

Example: Eclipse



Main Drawback

- *Framework* is usually large and complex
  - It is hard to understand
  - It may take long time to be effectively used
  - Developers may need only a simple functionality

Bibliography

- Ian Sommerville. **Software Engineering**, 10th Edition. Pearson Education, 2016.
  - Chap. 15 Software Reuse (mainly Section 15.2)