Brunel University London

Software Development and Management CS2002

Dr Giuseppe Destefanis
giuseppe.destefanis@brunel.ac.uk

Lecture 6

Design Patterns

Learning outcomes

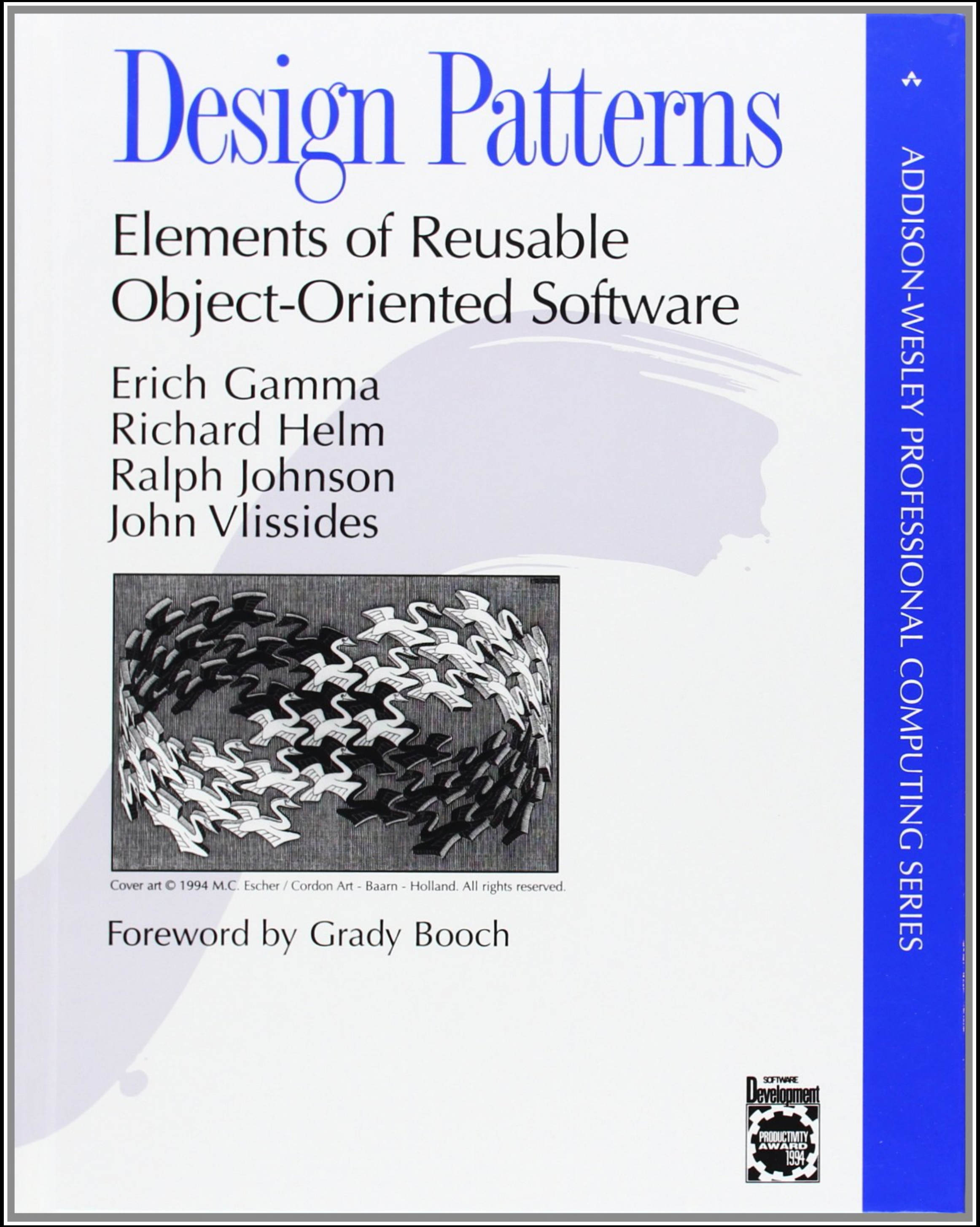
By the end of this session and period of independent study, the successful student should be able to:

- *Define the three main categories of patterns*
- *Apply the concept of patterns to practical situation*
- *Define and use the MVC pattern*

Design Patterns

typical problems often recur

23 Design Patterns


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
Software Development Productivity Award 1994

A pattern has four essential elements:

1. The **PATTERN NAME**
2. The **PROBLEM**
3. The **SOLUTION**
4. The **CONSEQUENCES**

Pattern Name

- is a handle we can use to describe a design problem, its solutions, and consequences in a word or two.
- Naming a pattern immediately increases our design vocabulary. It lets us design at a higher level of abstraction.
- Having a vocabulary for patterns lets us talk about them with our colleagues, in our documentation, and even to ourselves.
- It makes it easier to think about designs and to communicate them and their trade-offs to others. Finding good names has been one of the hardest parts of developing our catalog.

The Problem

- The **PROBLEM** describes when to apply the pattern. It explains the problem and its context.
- It might describe specific design problems such as how to represent algorithms as objects. It might describe class or object structures that are symptomatic of an inflexible design.
- Sometimes the problem will include a list of conditions that must be met before it makes sense to apply the pattern.

The Solution

- The **SOLUTION** describes the elements that make up the design, their relationships, responsibilities, and collaborations.
- The solution doesn’t describe a particular concrete design or implementation, because a pattern is like a template that can be applied in many different situations.
- Instead, the pattern provides an abstract description of a design problem and how a general arrangement of elements (classes and objects in our case) solves it.

The Consequences

- The **CONSEQUENCES** are the results and trade-offs of applying the pattern.
- Though consequences are often unvoiced when we describe design decisions, they are critical for evaluating design alternatives and for understanding the costs and benefits of applying the pattern.

a ***solution*** to a ***problem*** in a ***context***

Creational

Structural

Behavioural

Standing on the shoulders of giants

Design patterns capture the lessons distilled from experience of expert software developers

generalised solutions

to

recurring structural problems

Capture the expertise

Support Reuse

patterns facilitate experience and design reuse

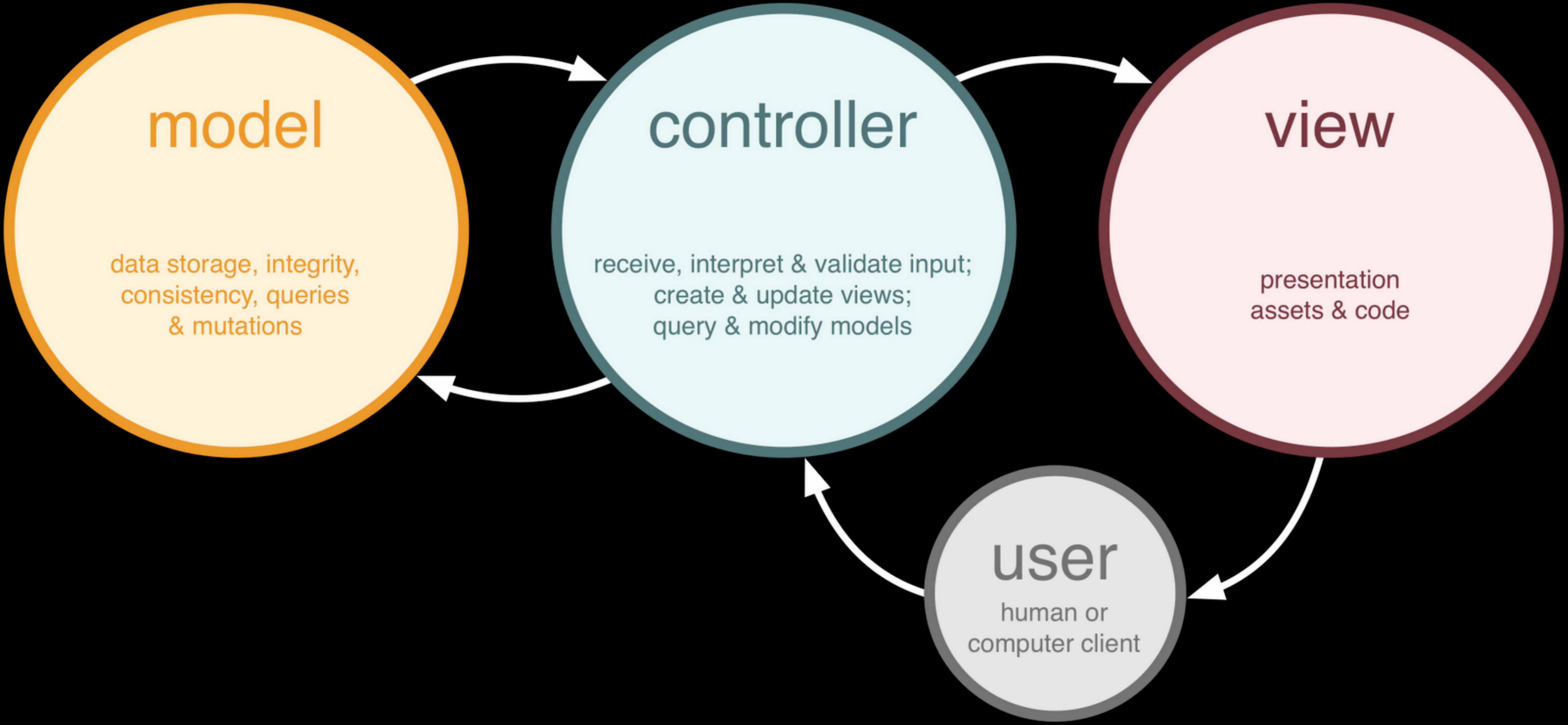
Improve software stability

Patterns describe:

- How software is structured
- How classes and objects interact

Patterns provide a common vocabulary for developers

Model-View-Controller



Model

Responsible for maintaining data

(It can also have logic to update controller if its data changes)

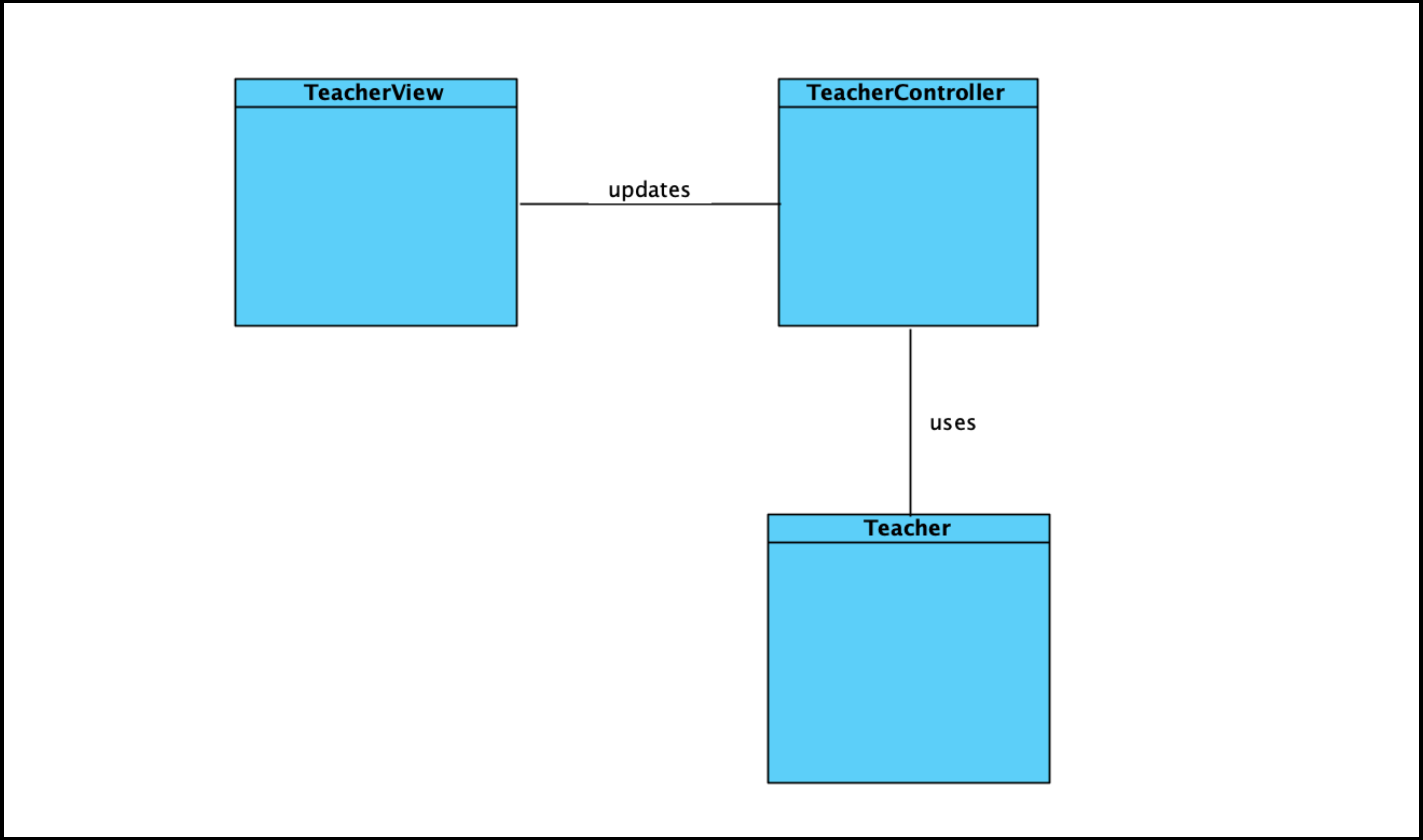
View

represents the visualisation of the data which the model contains

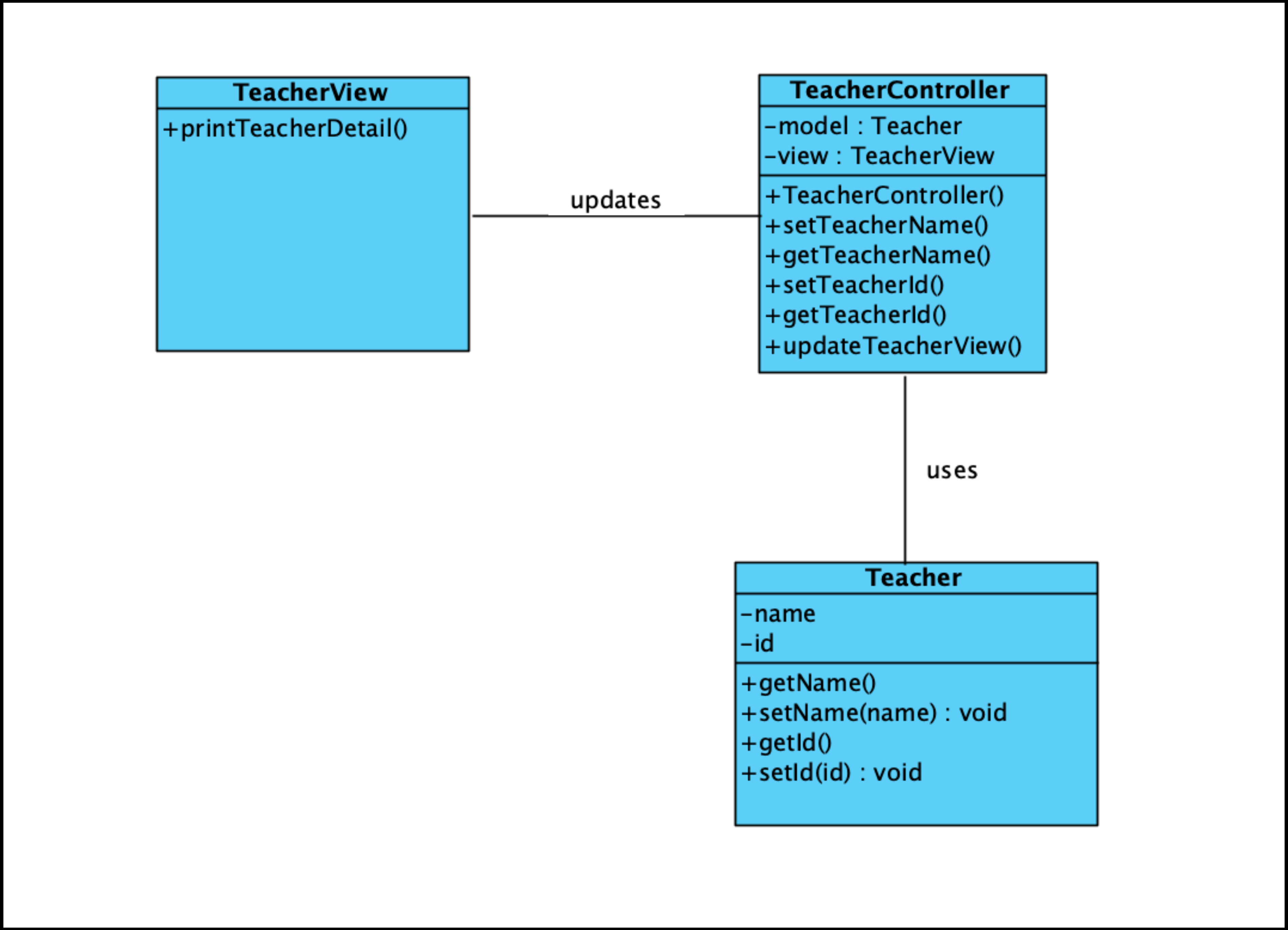
Controller

- acts on both model and view;
- It controls the data flow into model object and updates the view whenever data changes;
- it keeps view and model separate.

Example



Example


TeacherView
+printTeacherDetail()
updates
TeacherController
–model : Teacher
–view : TeacherView
+TeacherController()
+setTeacherName()
+getTeacherName()
+setTeacherId()
+getTeacherId()
+updateTeacherView()
uses
Teacher
–name
–id
+getName()
+setName(name) : void
+getId()
+setId(id) : void

Example and reading

- https://www.oracle.com/technical-resources/articles/javase/mvc.html
- Chapter 6, Sommerville, paragraph 6.3