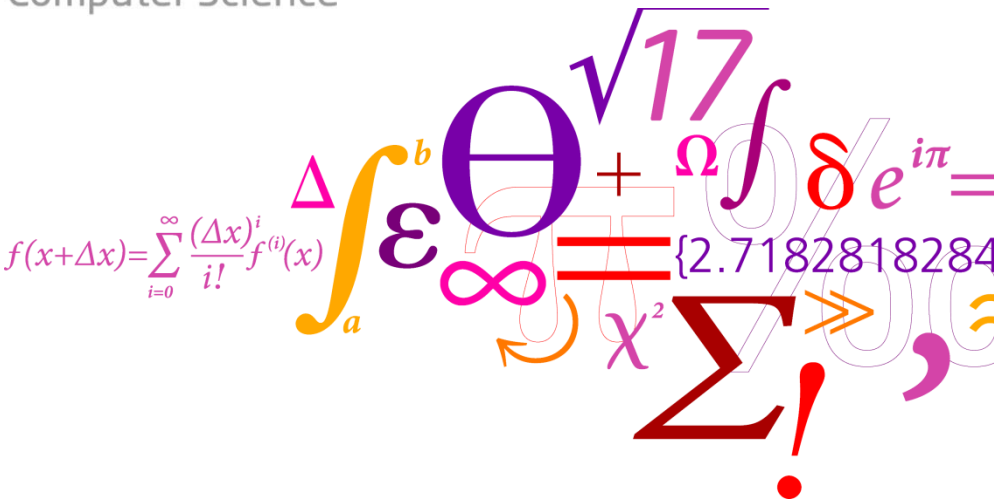
Model-based Software Engineering
(02341, spring 2016)

Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



II. Modelling with a Purpose

Brief recapitulation of lecture 2

DTU Compute
Department of Applied Mathematics and Computer Science

1. Models to which end

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Understanding the world (conceptual models, domain models)
- Understanding what the software is supposed to do (requirements)

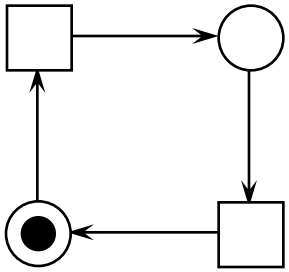
- Understanding and finding your way round in existing software
- Outline the idea of how to realize the software (architecture)
- Overview of componenents and their interplay
- Detailed design and realization of the software

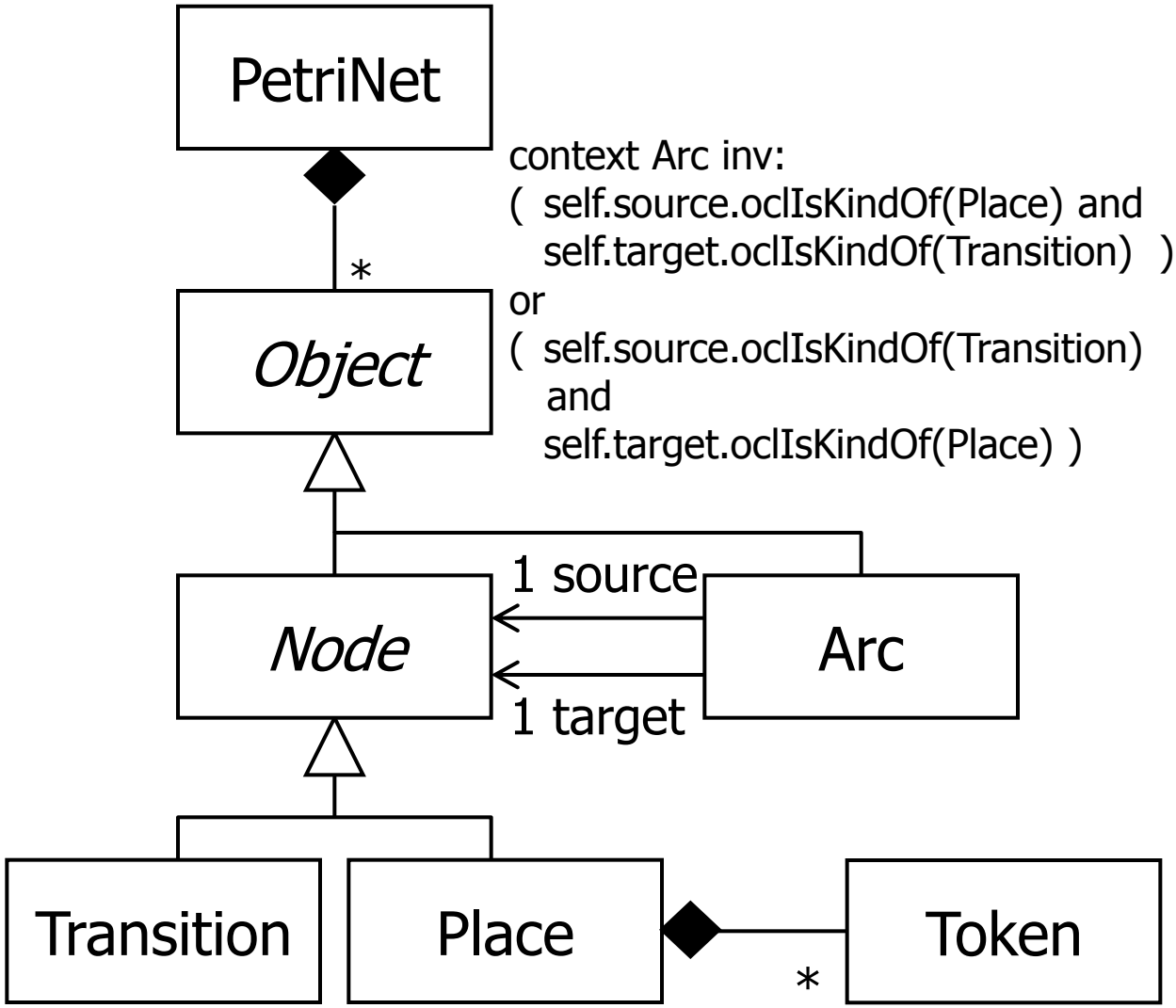
Models to which end (cntd.)

- Generate parts of the software automatically
- Define data representations (XML, database schemas, ...)
- Define interfaces between different components of the software
- ...

2. Domain model

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

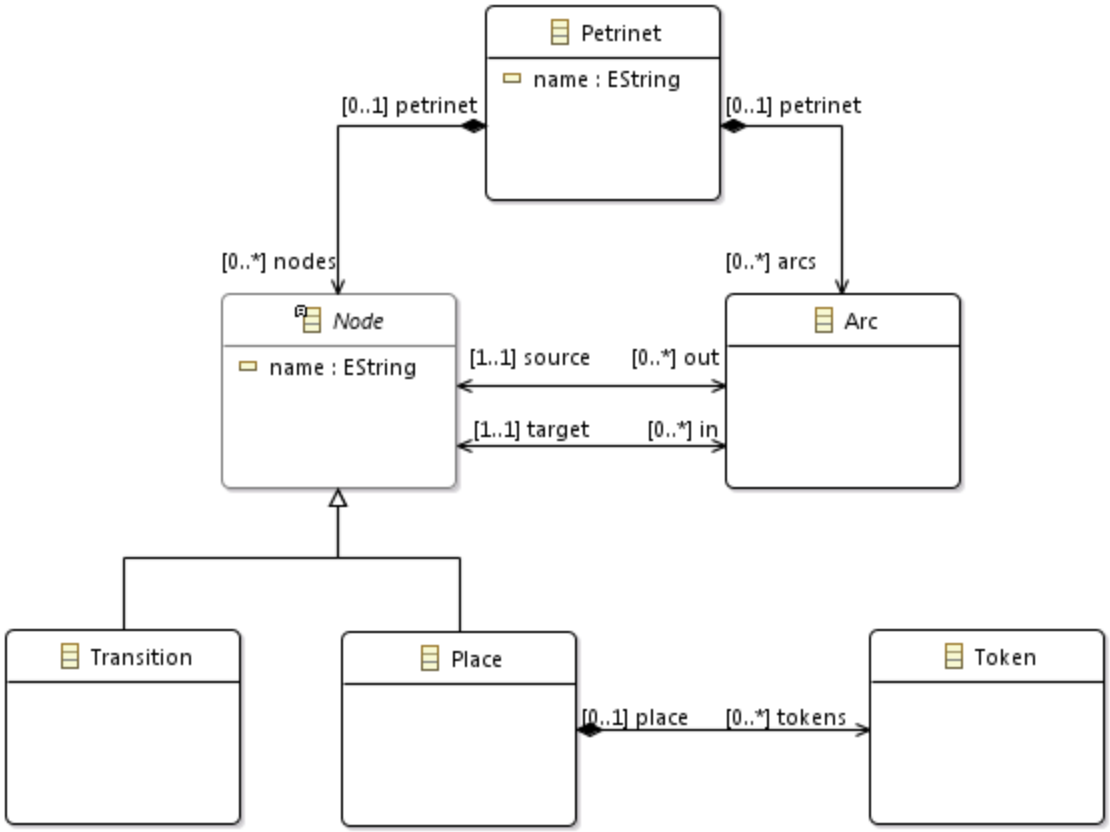





```
context Arc inv:
( self.source.oclIsKindOf(Place) and
  self.target.oclIsKindOf(Transition)  )
or
( self.source.oclIsKindOf(Transition)
  and
  self.target.oclIsKindOf(Place) )
```

Ecore model

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Representations of model

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Same model can have different representations:

- Graphical / tree (as of Tutorial 1)
- Java
- Ecore
- XML Schema (XSD)

Actually, in our EMF technology, Ecore models can be imported from XML Schema and from annotated Java classes (see Java example on the next slides).

Different representation might serve different purposes and have a different focus!

What would the focus for XSDs, Java and Ecore be?

Also JPA can be considered a model represented in Java (with annotations mapping it to a database schema).

3. Software Models

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

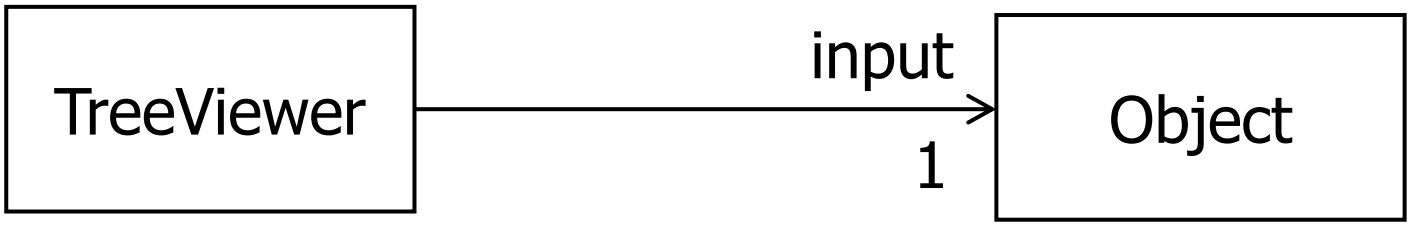
Two objectives:

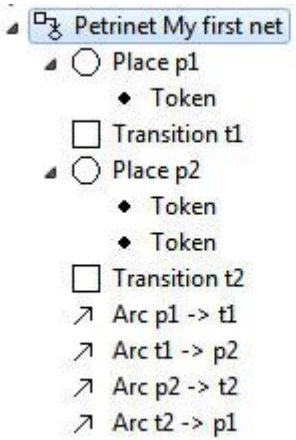
- Understand (a bit) the generated code and the framework behind it
- See how models can be used for that purpose

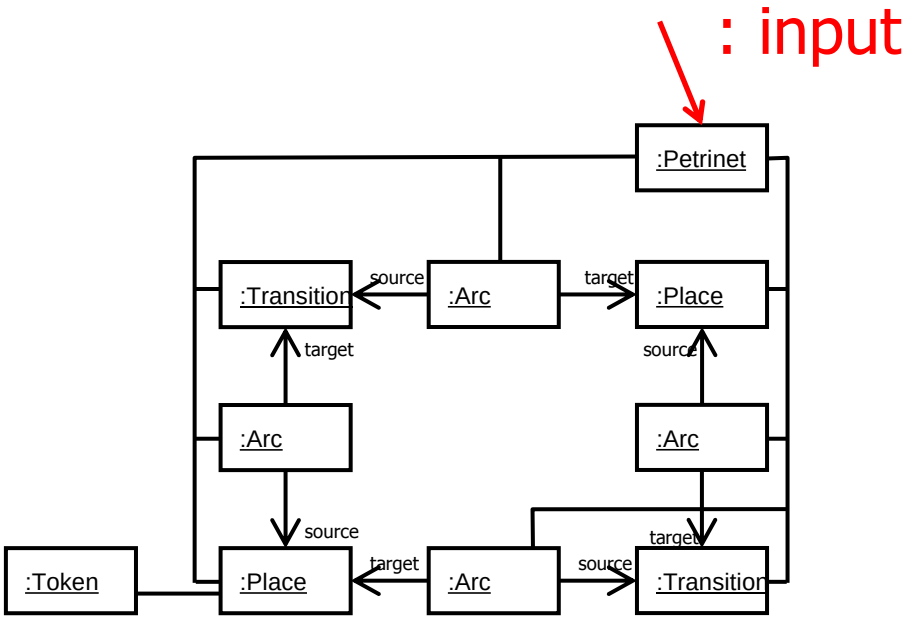
How???

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- How could the TreeViewer, which does not know anything about Petri nets (and the classes representing the concepts of Petri nets), know how this tree should be shown?

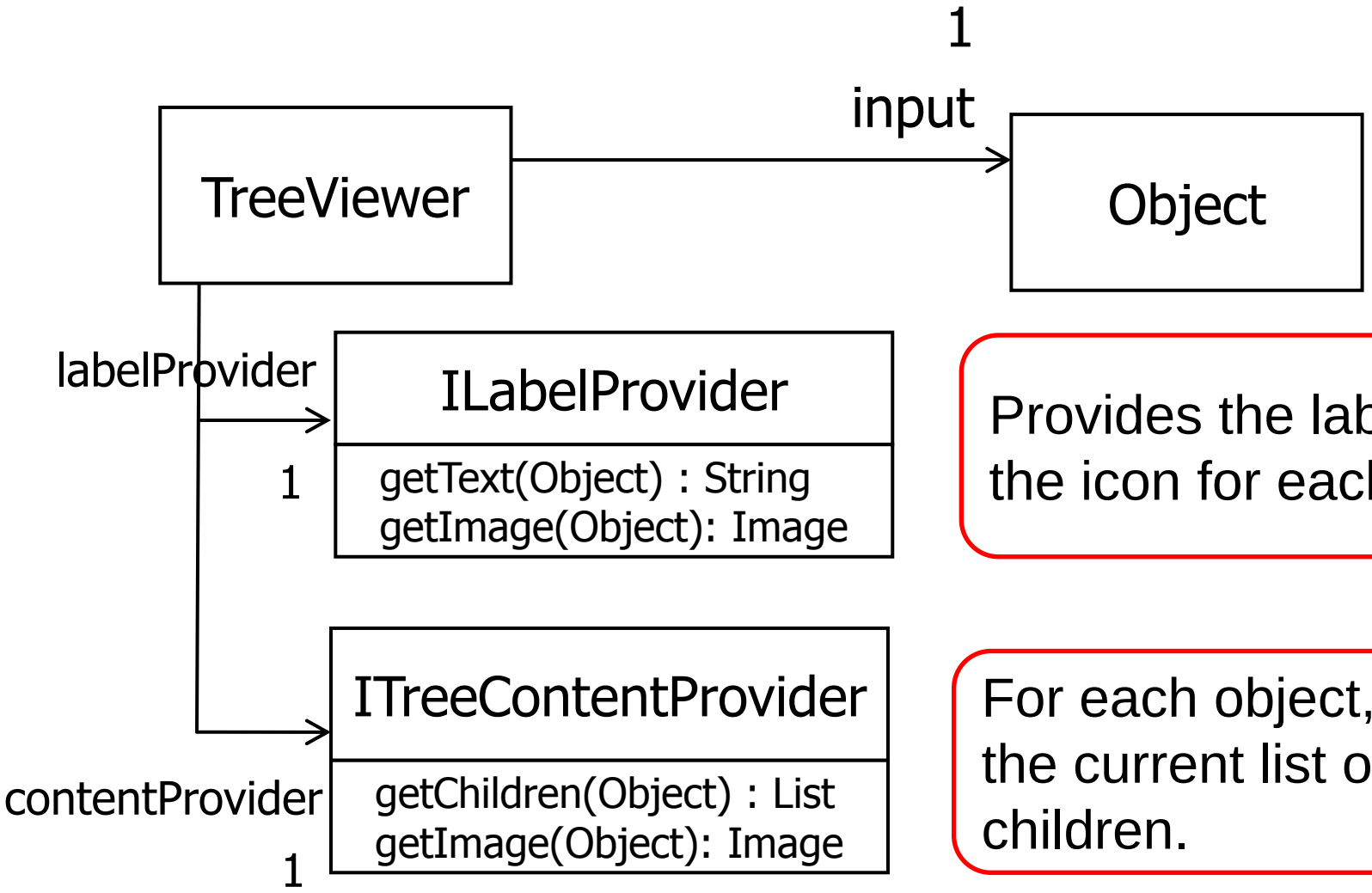






TreeViewer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



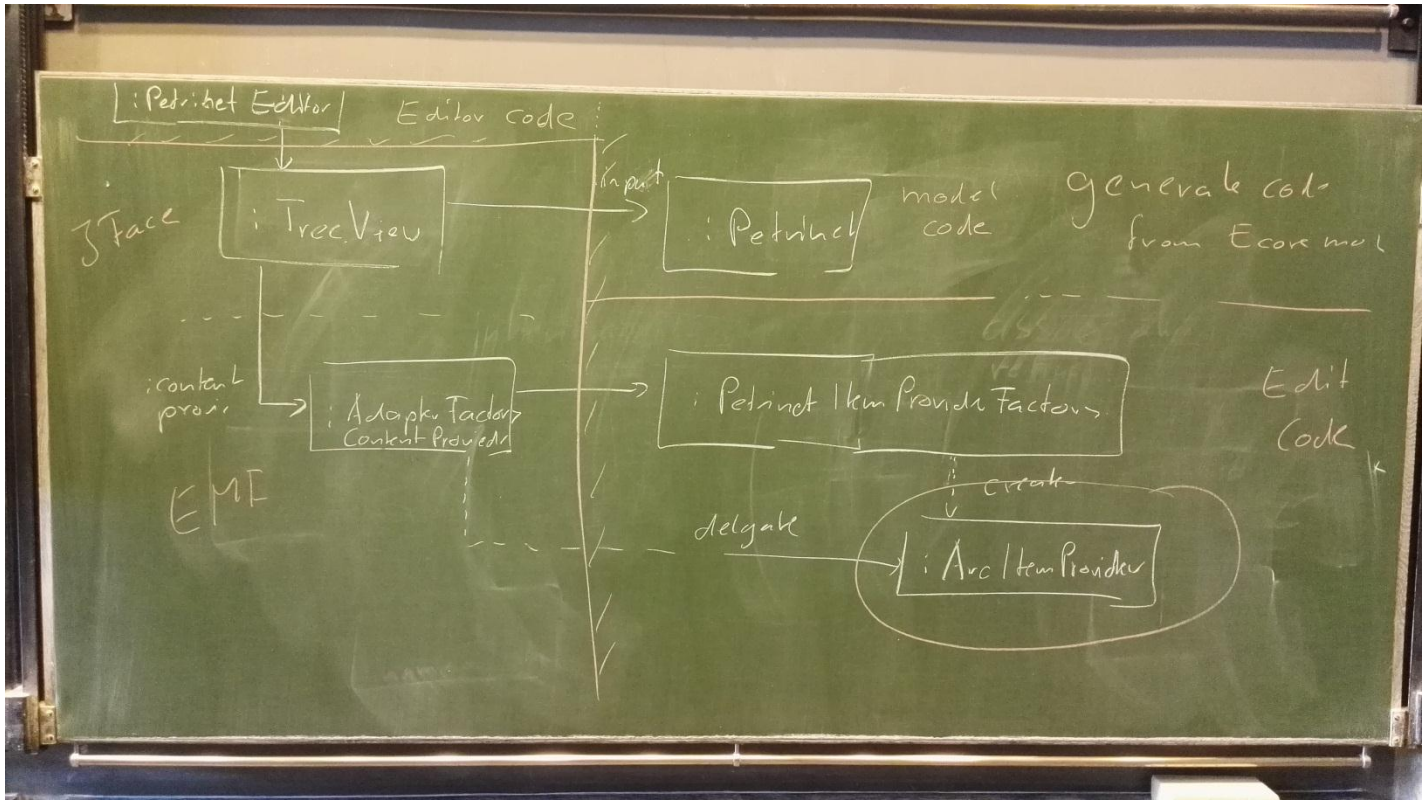
Provides the label and the icon for each object

For each object, provides the current list of children.

3.2 EMF: Use of TreeViewer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

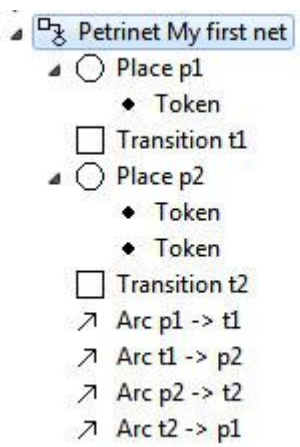
- In EMF, this is even more complicated: using a generic ContentProvider, which creates the respective ItemProviders and delegates to them

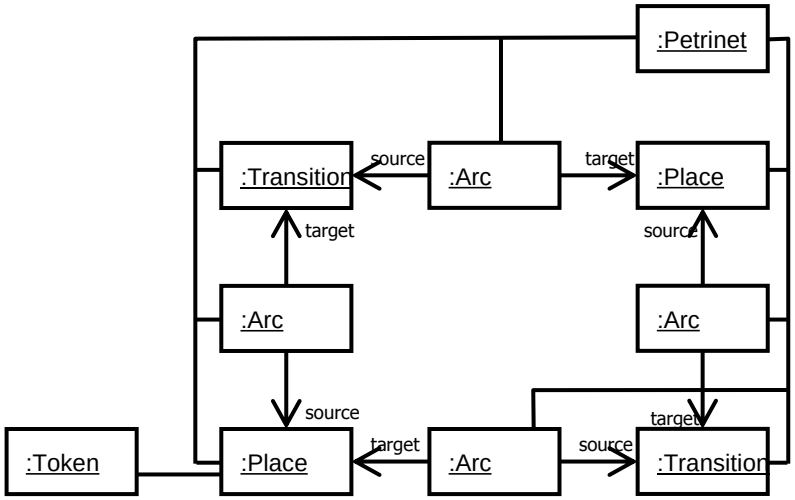


3.3 Update Viewer on Changes

- In order to make sure that the viewer properly updates, whenever changes occur, it registers itself as listener to the respective elements (actually to their ItemProviders.

Idea is discussed on blackboard (BBD); more details next time





Discussion

- Domain models

vs

- Software models

III. Design Patterns (How / Software Models)

DTU Compute
Department of Applied Mathematics and Computer Science

1. Introduction

Originally, the term was introduced in architecture: Alexander et al. 1977.

Design patterns (in software engineering) are the distilled experience of software engineering experts on **how** to solve standard problems in software design.

Freeman & Freeman call this "experience reuse"!

The generated code and the underlying framework of EMF extensively use design patterns.

Design Patterns in SE:

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Often called the “Gang of Four” (GoF / Go4).

- Gamma, Helm, Johnson, Vlissides: Design Patterns. Addison-Wesley 1995.
- Eric Freeman, Elisabeth Freeman: Head First Design Patterns. O’Reilly 2004 [FF]
- …

Disclaimer

- Design patterns are a topic of their own, worth being taught as a separate course (e.g. seminar/special course)
- This lecture gives just a glimpse of the general idea and some patterns, which are important to understand and use EMF

III. Design Patterns (How / Software Models)

2. Examples

DTU Compute
Department of Applied Mathematics and Computer Science

Example: Observer (GoF)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Name and classification

Observer, object, behavioural

Intent

”Define a one-to-many dependency between objects so that an object changes all its dependents are are notified and updated automatically” [GoF].

Also know as

Dependents, Publish-Subscribe, Listener

Example: Observer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Motivation

[...] maintain consistency between related objects without introducing tight coupling (which increases reusability) [...]

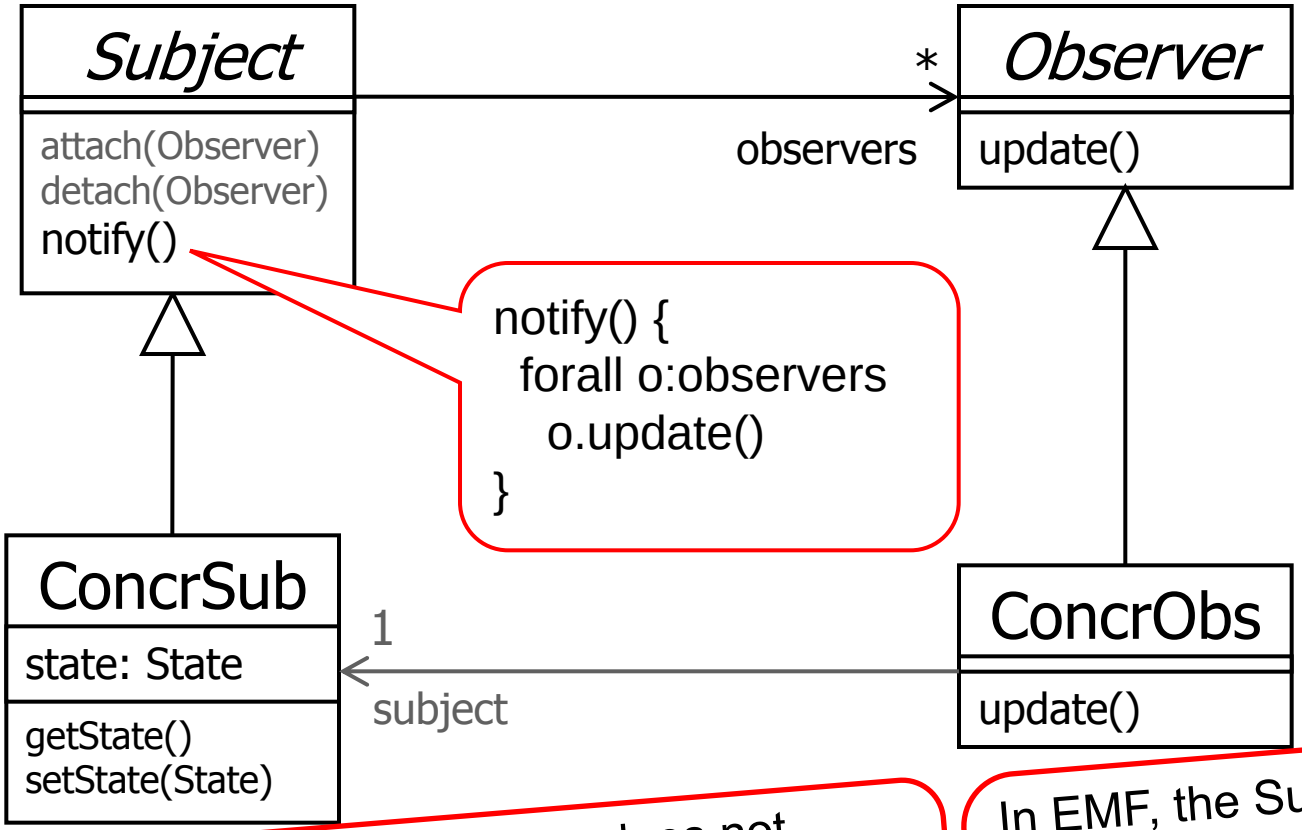
Typical Example

... update views when the underlying model changes ...

Roughly following GoF

Example: Observer

Structure



The concrete observer does not necessarily need to “know” the concrete subject; then the subject is passed as a parameter to update()

In EMF, the Subject is (roughly) EObjectImpl, the Observer is Adapter (it is called adapter since it is also an Adapter in the sense of GoF and more).

Example: Observer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Participants (see structure)

~ GoF

- **Subject**
  - knows its observers
  - provides an interface for attaching and detaching Observer objects
- **Observer**
  - defines the updateing interface for being notified
- **ConreteSubject**
  - stores the state (of interest)
  - sends notifications
- **ConreteObserver**
  - Implements the Observer‘s updating interface to keep its state consistent

Example: Observer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Collaboration

Black board discussion

Scheme (GoF)

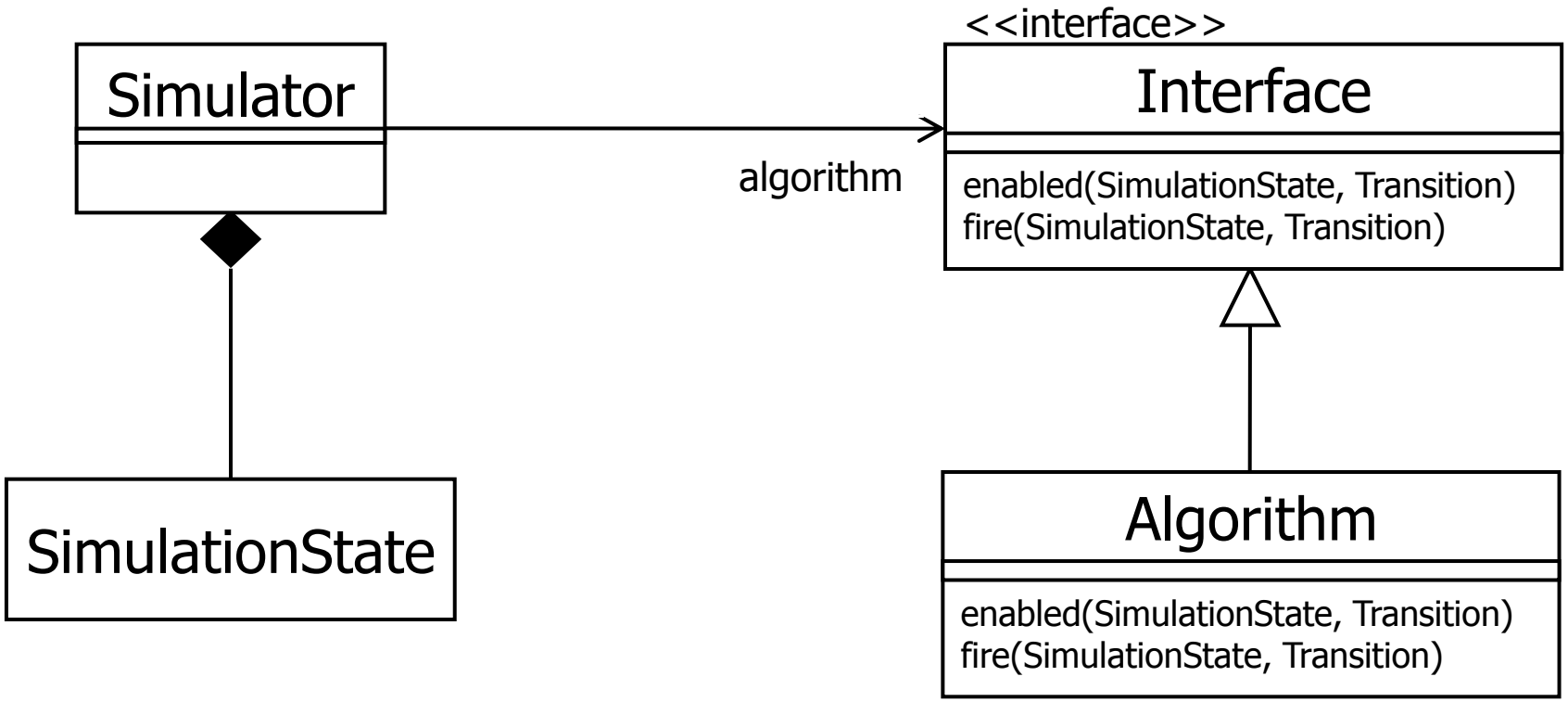
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Name
- Classification
- Intent
- Also known as (aka)
- Motivation
- Application
- Structure
- Participants
- Collaboration
- Consequences
- Implementation
- Sample code
- Known uses
- Related patterns

Sometimes there is more:
Variants, “Counter indications”,
…

Example: Simulation Algorithm

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Pattern: Strategy (GoF)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Name and classification

Strategy, object-based, behavioural

Intent

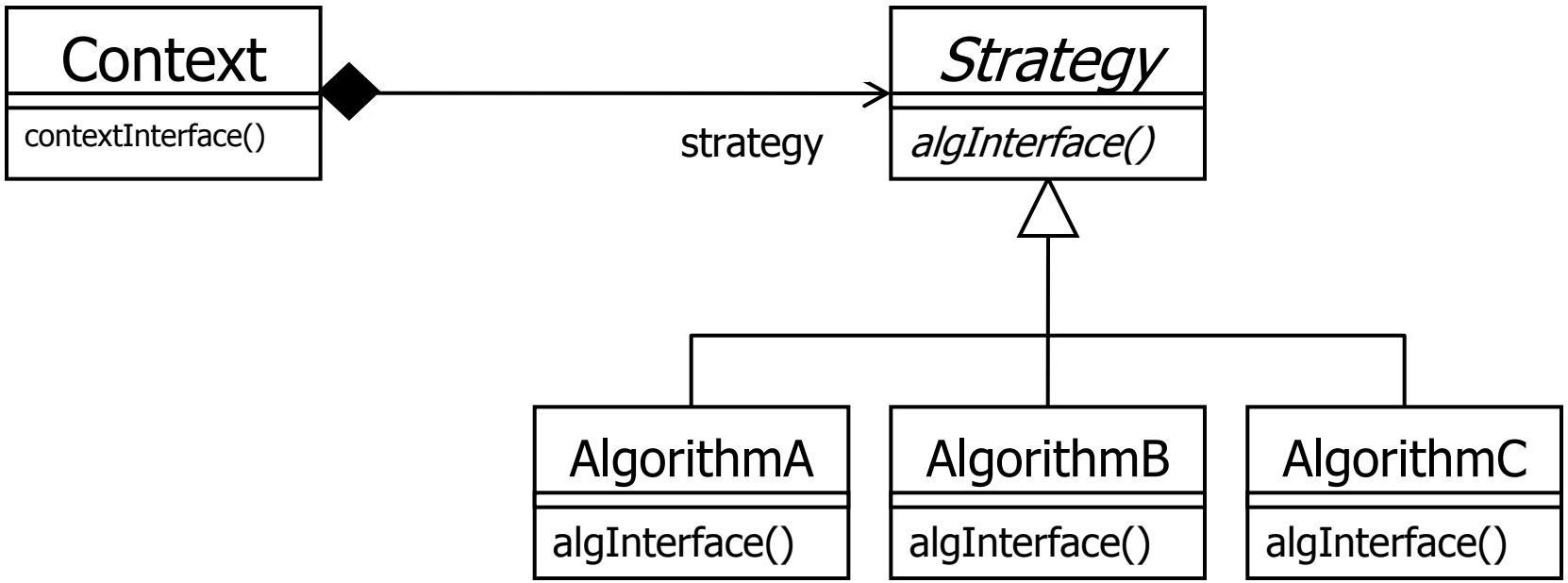
Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it [GoF]

Motivation

Avoid hard-wiring of algorithms for making it easier to change the algorithm …

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Structure



Pattern: Strategy (cntd)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

We skip the rest of the GoF scheme here.

Questions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Is the “simulation algorithm” a strategy?

Patterns should not be applied too mechanically!
But sometimes details make a difference (e.g. State Pattern vs. Strategy Pattern)

Pattern: Abstract Factory

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

What we called Factory up to now.

Name and classification

Abstract factory, object, creational

Intent

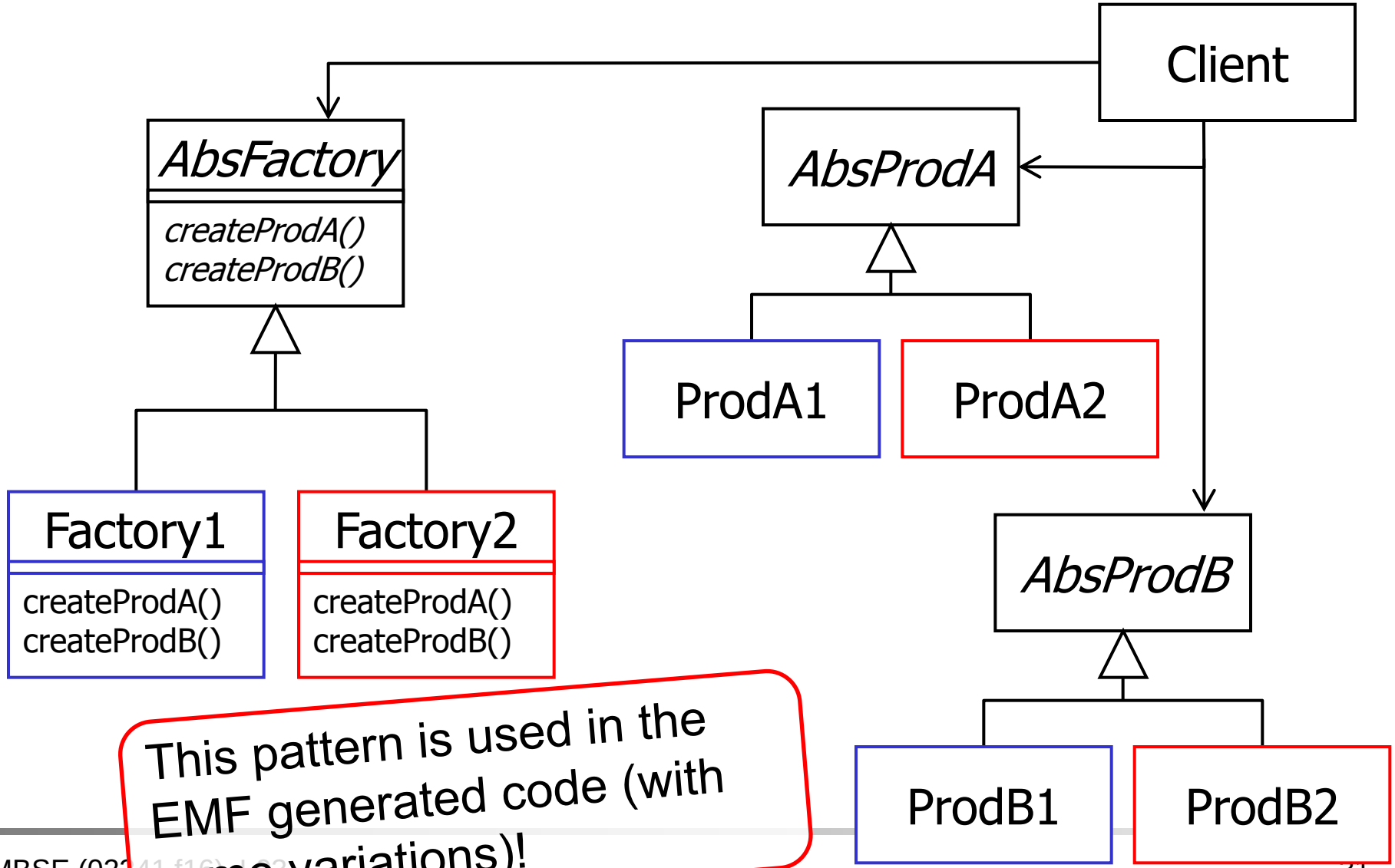
Provide an interface for creating families of related or dependent objects without specifying their concrete classes [GoF]

Motivation

Use of different implementations in different contexts with easy portability …

Pattern: Abstract Factory (cntd)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



This pattern is used in the EMF generated code (with some variations)!

Pattern: Singleton (GoF)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Name and classification

Singleton, object-based, creational

Intent

Ensure that a class has only one instance, and provide a global point of access to it [GoF]

Motivation

…

See [GoF] or [FF] for details.

Other relevant patterns

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Factory Method
- Command
  (see Tutorial 2)
- Adapter

The Factory Method pattern is different from the Abstract Factory.

The EMF Commands are commands in this sense (actually very sophisticated ones).

The Eclipse command is **not** a command in the sense of GoF! An Eclipse command is basically just a name, which is than implemented by a handler (which in some way is a command in the sense of GoF and a bit more).

3. Summary

- GoF present 23 patterns
- There are many more (and more complex combinations of patterns, e.g. MVC --)

- “Pattern terminology” can be used to communicate design!
- Patterns should **not** be used to schematically
- Generated code, typically, makes use of many patterns. Automatic code generation “saves us making some design decisions” (observer, singleton, factory, and adapters are part of the EMF-generated code)

Scheme (GoF)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Name
- Classification
- Intent
- Also known as (aka)
- Motivation
- Application
- Structure
- Participants
- Collaboration
- Consequences
- Implementation
- Sample code
- Known uses
- Related patterns

Sometimes there is more:
Variants, “Counter indications”,
…

4. Model View Controller (MVC)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

The domain models are an (the) essential part of the software

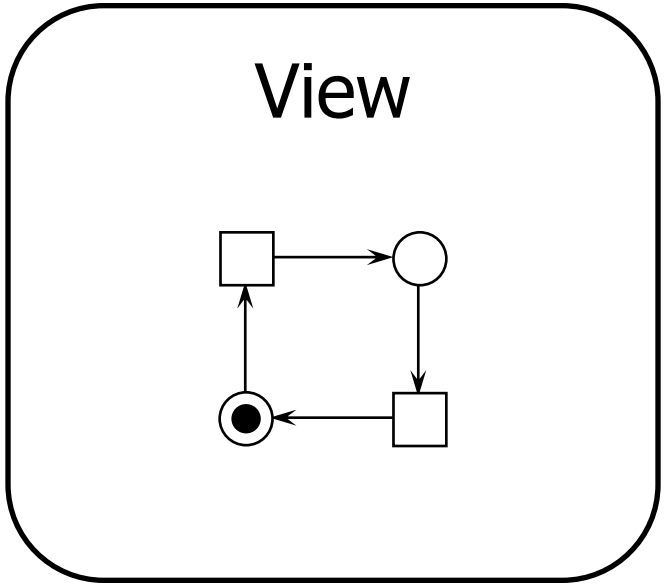
In addition to that we need

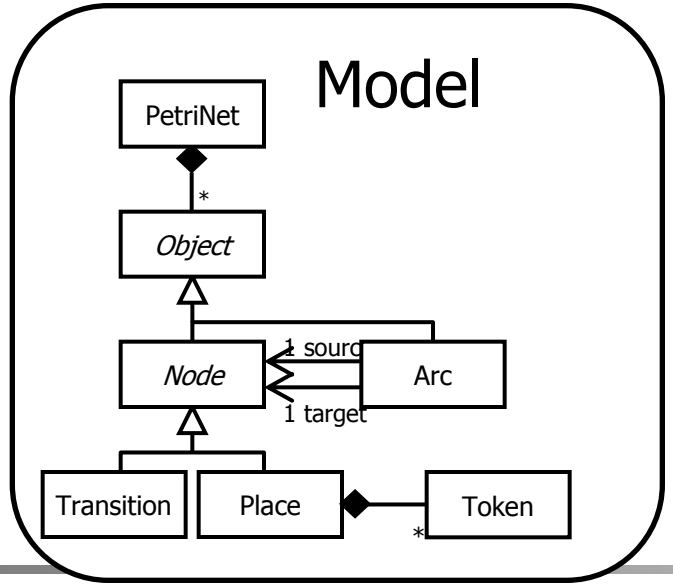
- Information about the presentation of the model to the user
- The coordination with the user

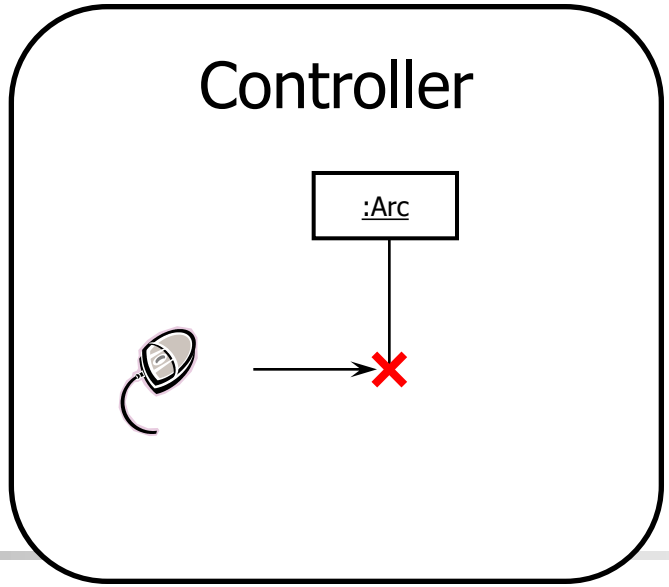
Note: These parts of the software can be modelled too (don't get confused: „models are everywhere"); domain model vs. software model

Modelle View Controller (MVC)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



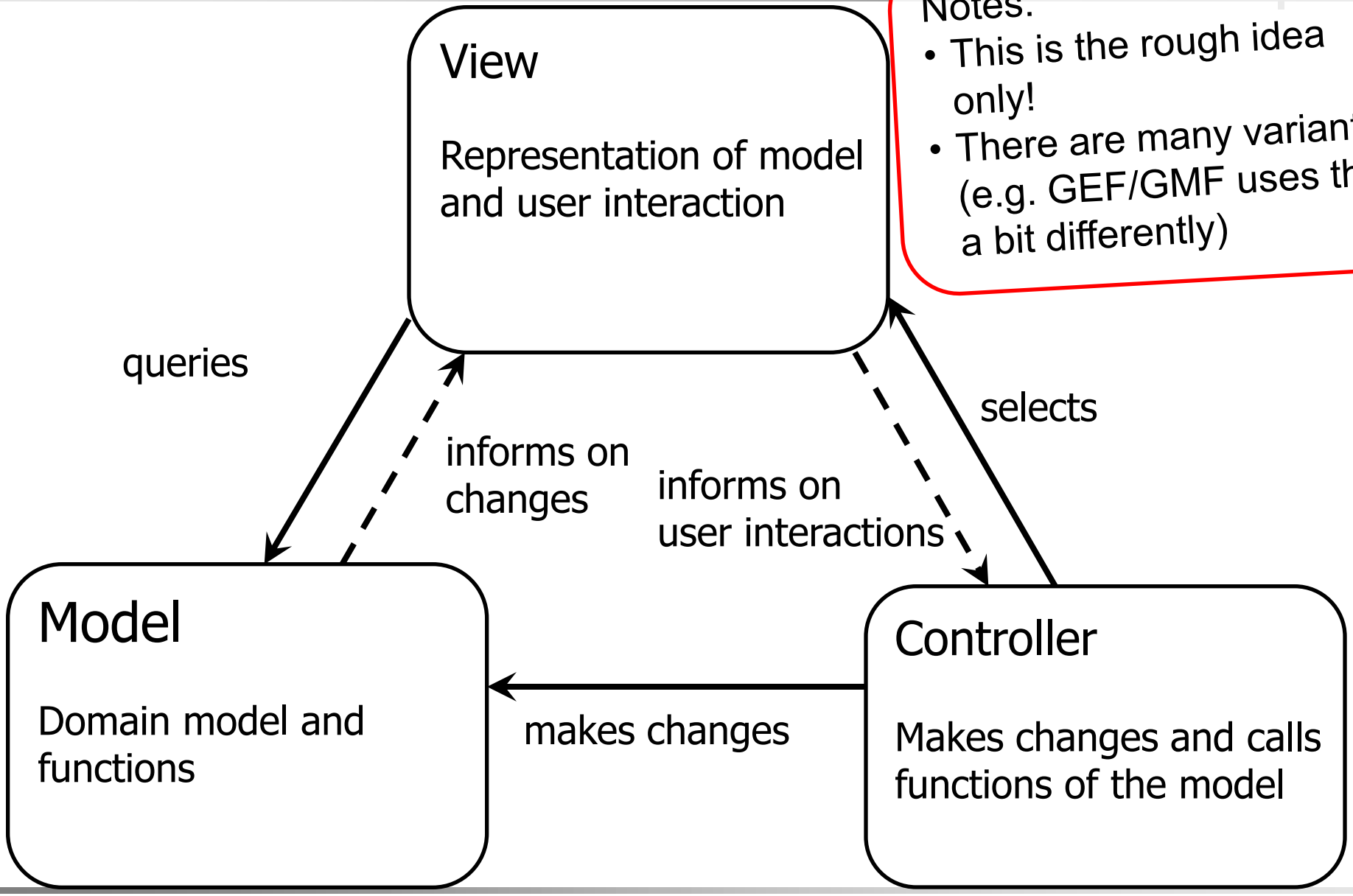




MVC

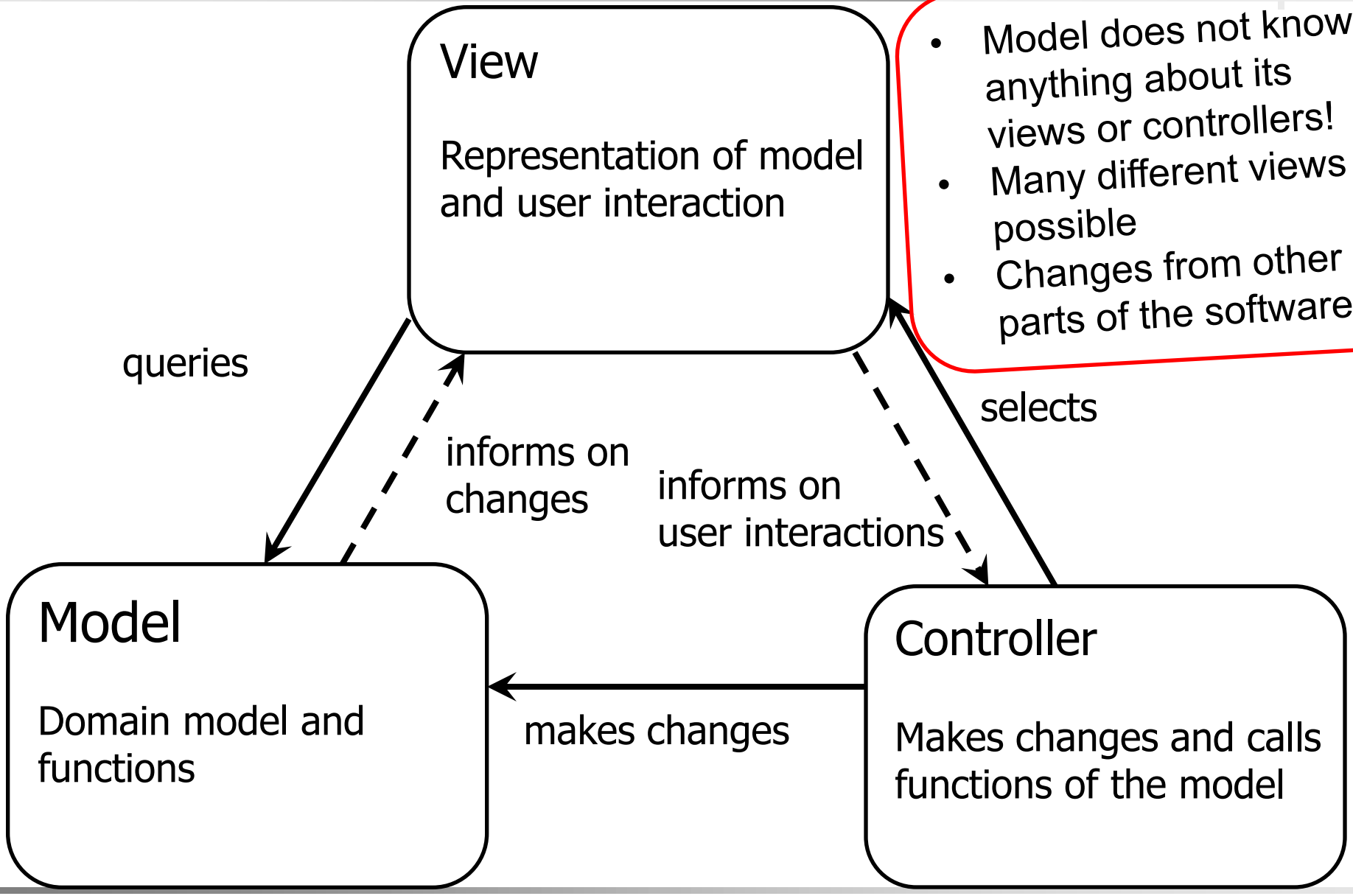
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Notes:
- This is the rough idea only!
- There are many variants (e.g. GEF/GMF uses this a bit differently)



MVC

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



- Model does not know anything about its views or controllers!
- Many different views possible
- Changes from other parts of the software

MVC

MVC is a principle (pattern / architecture) according to which software should be structured

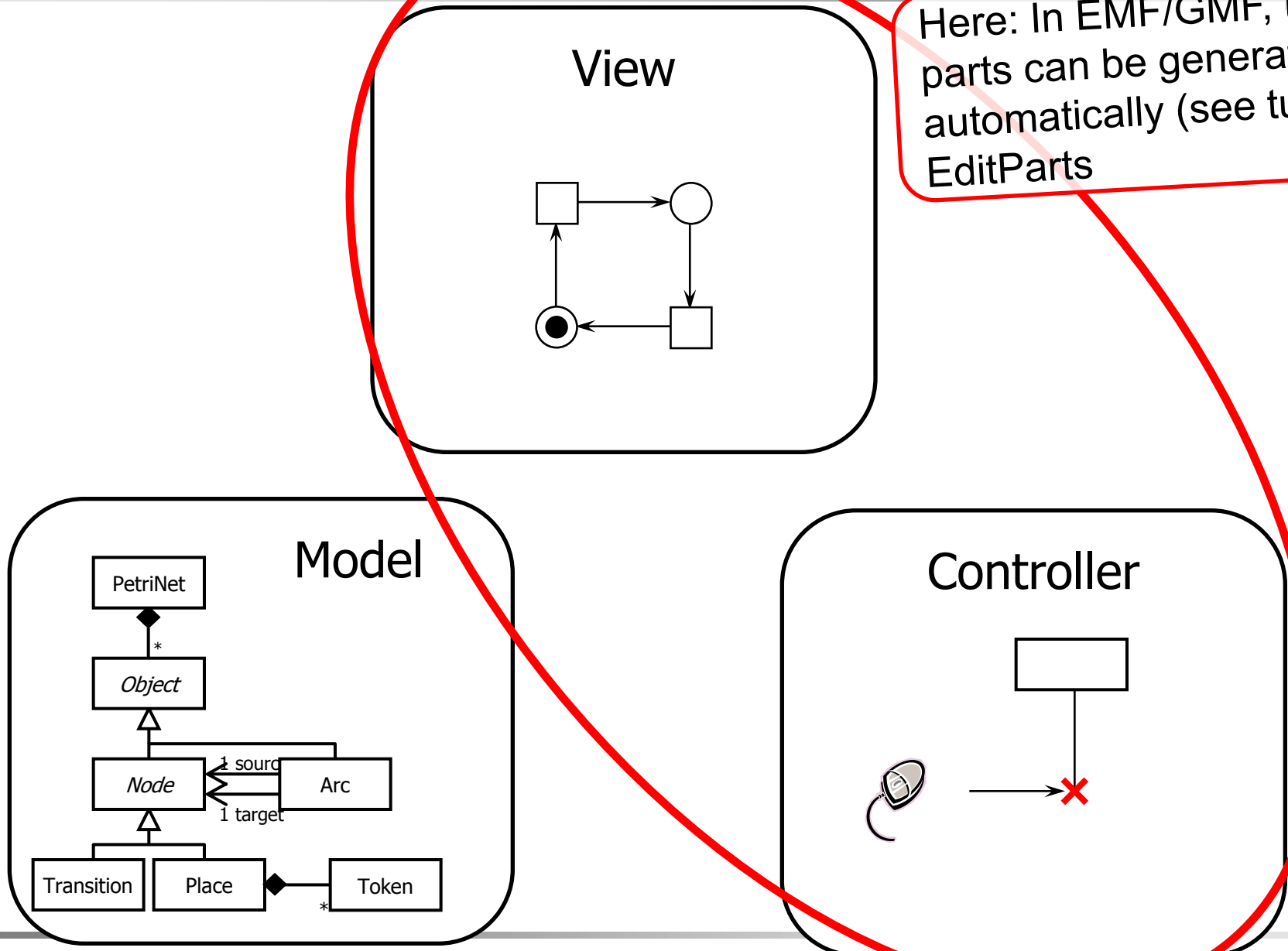
Eclipse and GEF (as well as GMF) are based on this principle and guide (force) you in properly using it

If things do not work out with EMF for you, you might have messed with the MVC pattern.

EMF, GMF and MVC

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Here: In EMF/GMF, these parts can be generated automatically (see tutorials EditParts



Tutorial 2: Q & A

DTU Compute
Department of Applied Mathematics and Computer Science

Tutorial 3: Discussion Assignment 3

DTU Compute
Department of Applied Mathematics and Computer Science