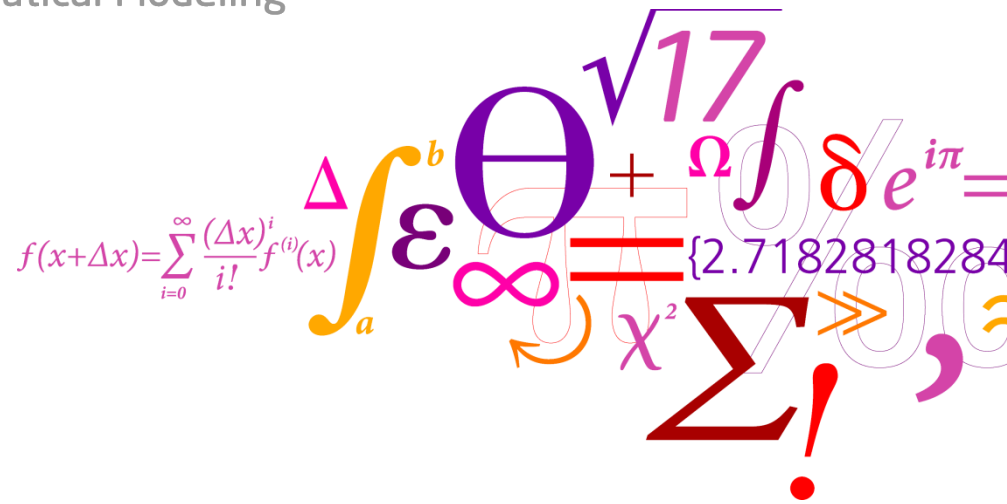
Software Engineering 2
A practical course in software engineering

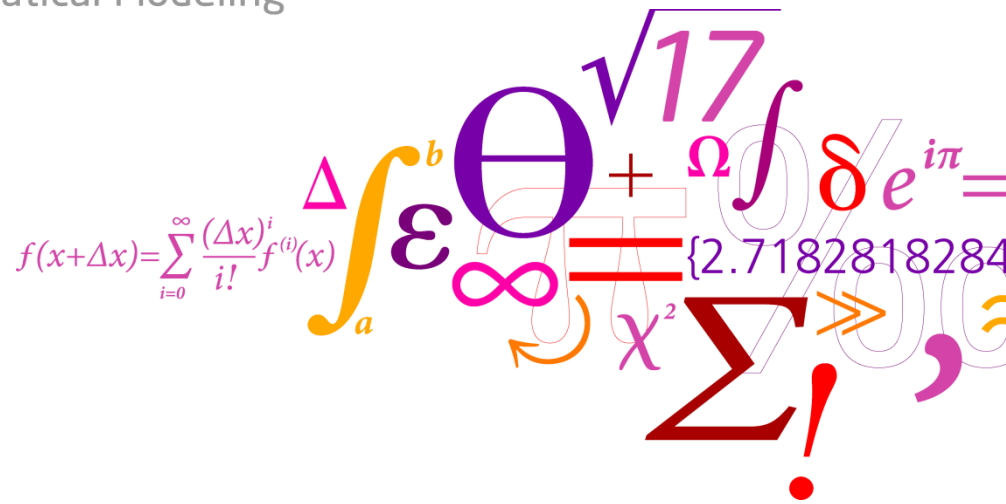
Ekkart Kindler

DTU Informatics
Department of Informatics and Mathematical Modeling



I. Introduction

DTU Informatics
Department of Informatics and Mathematical Modeling



Introduction

- Motivation
- The role of models in software engineering
- Software engineering & management

- Organisation of this course

- Project & tutorials
  - The task
  - Technology tutorials
  - Forming the groups

Weekly Schedule (roughly)

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

	Mon	Tue	Wed	Thu	Fri
8-10		lecture			
10-12					
13-15					
15-17					project

Plus actual work on the project!!

lecture

tutorial

project

There will be many exceptions from the rule! See web pages for details!

1. Motivation

- Objectives of this course:
  Skills in software engineering!

- What is “software engineering”?
- What is “software”?

- software = program
- software engineering = programming

Program vs. Software

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

is much more than

Software >> Program

Software Engineering >>> Programming

is much much more than

Programmi

Software

Software Engineer

Software Engineering

If somebody has built a garage, would we let him built a skyscraper? No, never!

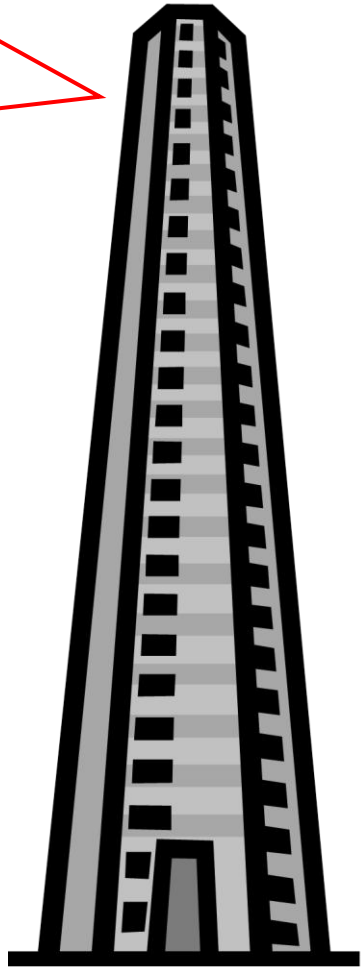
If somebody has written a program, would we let him built software? Yes we would!

Program

Programmer

Programming

But, of course we should not!



Software Engineering is

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

… much more than programming!

… listening and understanding!
… analytic and conceptual work!
… communication!
… a social process!
… acquiring and using new technologies!
…

but, still evolving

… a discipline with proven concepts, methods, notations, and tools!
… and ever new technologies emerging!

Experience

Software Engineering requires much experience!

This experience

- can not be taught theoretically!
- will be provided in this course!

→ project
→ tutorial (new technologies)
→ and (only) backed by the lectures

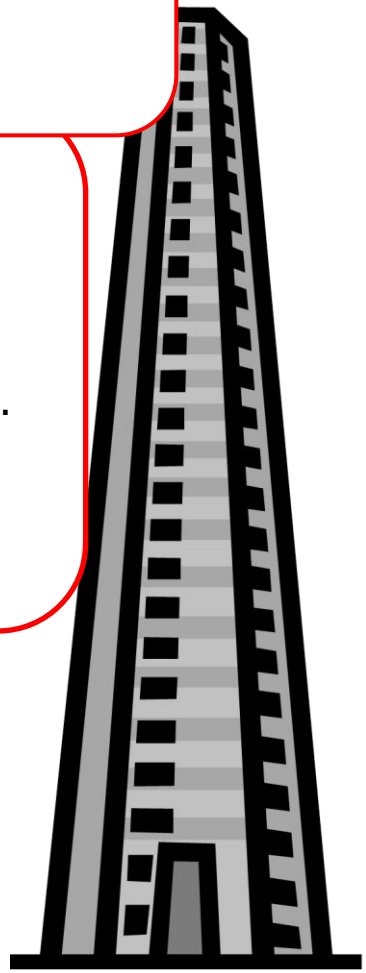
Analogy revisited

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

Effort per participant

- 10 ECTS = ca. 270h work
- ca. 20h/week

The experience of a big project cannot be replaced by the experience of many small ones.



Objective

Practice the concepts, methods, notations and tools for software engineering

- improve programming skills
- understanding of the software engineering process
- experiences with problems and concepts for solving them
- writing and creating documents and models
- use of methods and tools
- practice communication and presentation skills
- capability of teamwork and leadership
- acquire new technologies
- …

If in doubt:
1. think
2. search (“google”)
3. ask

Excursion: CDIO

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

Conceive

Design

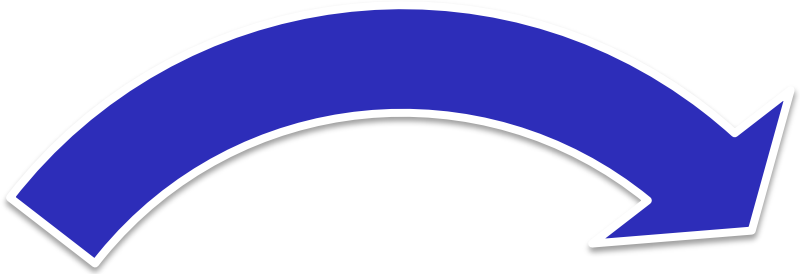
Implement

Operate

Co-evolution

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

"What" should the software do?



WHAT HOW

Reality is even more complicated, but keeping "what" and "how" seperate in your mind is a first step.

"How" is it realized?

Questions

- Why do so many software projects fail?
- Why is software development so hard (or at least harder as we believe)?
- BTW: What is software?

Software

Die Menge aller **Programme**, **Prozeduren** und **Objekte**, zusammen mit den zugehörigen **Daten** und der **Dokumentation**, die für eine lauffähige Anwendung nötig oder wünschenswert sind.

[frei nach Informatik DUDEN und Hesse]

Software

The sum of all **programs**, **procedures** and **objects** along with the associated **data** and **documentation**, which are necessary (or at least desirable) for running an application on a computer system.

Software …

is becoming more and more complex!

Exponential growth of software (in „lines of code“ LOC) within the same product line:

- Apollo (NASA‘s Apollo programme)
- Cars (automotive software)
- …

Technology needs to keep up with that!

Software …

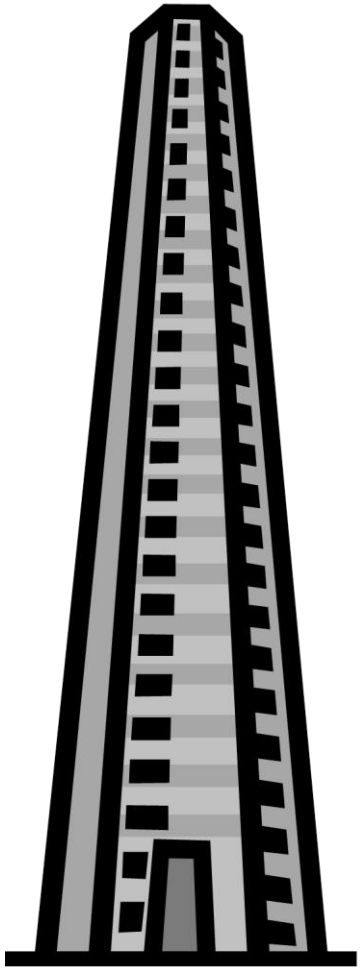
cannot be „programmed“ by a single person anymore; a single person cannot fully comprehend all details of software any more.

Efforts of 10 to 100 person years (PYs) are quite standard in software development.

Software …

is intangible.

You cannot touch, see or feel software. Humans lack a "natural feeling" of software and its complexity.



Software …

does not wear out,
but becomes of age anyway
(in relation to the environment it is running in and the expectations of the end user)!

Software needs „maintenance“! But, this does not mean the same as in traditional engineering (e.g. in mechanical engineering, where systems physically wear out).

Actually, maintenance is a big factor in the cost of IT systems.

Software …

„lives“ longer than its creators expect it to live.

Software …

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

is everywhere and many lifes depend on it.

Software Engineering is

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

… much more than programming!

… listening and understanding!
… analytic and conceptual work!
… communication!
… a social process!
… acquiring new technologies!

Problems

- imprecise requirements
- mistakable and unclear requirements
- inconsistent requirements
- changing requirements

- changing environments (software / hardware)
- different versions and configurations
- changing tools, notations, languages, methods, concepts, technologies

- collective knowledge only
- communication
- …

Software engineering

is the sum of all means, facilities, procedures, processes, notations, methods, concepts for developing, operating and maintaining a software system.

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

Branches:

- **Development**:
  actual development of the software product
- **Management**:
  Manage (control and improve) the development process
- **Quality management**:
  Planning and implementing measures that guarantee that the software meets the required quality
- **Software maintenance**:
  Remove faults occurring in operation, adapt software to changing requirements and environments

Process models

Process models (life cycle models) are the „distilled“ experience of successful software projects.

They define a functional procedure along with appropriate documents.

▪ **What** should be done	document, notation
▪ **when**,	phase
▪ by **whom** and	role
▪ **how**!	method

Process models

Problem: Often process models are used very mechanical and in a „meaningless“ way.

- → documents just for the sake of the process
- → (UML) diagrams just for the sake of UML
- → comments just for the sake of comments

Therefore:

1. Think!
2. What is reasonable?

Rule of thumb

When producing and compiling a document, ask yourself:

- What should the document be good for?
- Who should be addressed?
- Which information is expected?
- What is the common „pragmatics“?
- …

In short: What is reasonable?

Lectures and discussions will give some guidelines, though!

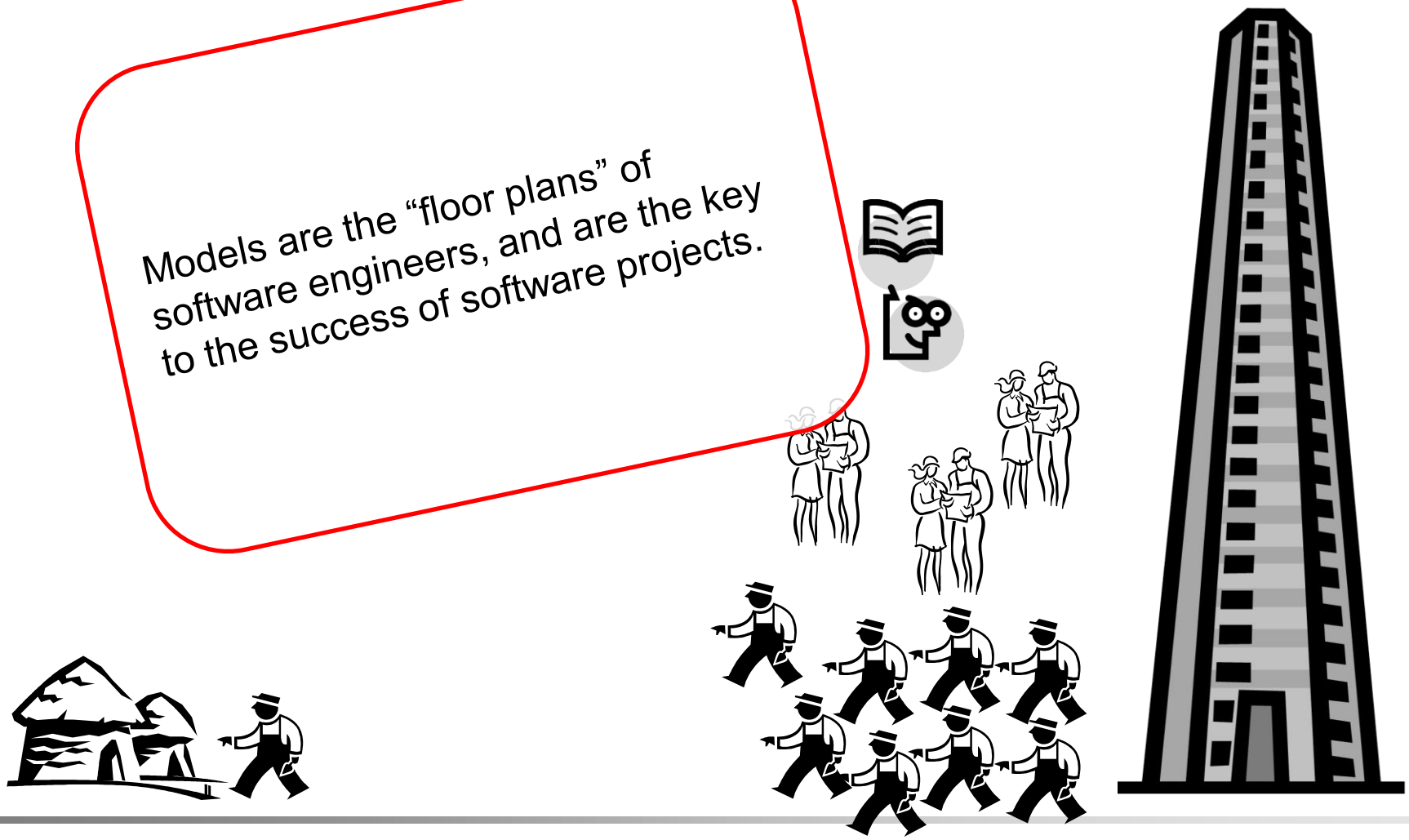
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

… and a glimpse of how software can be developed by using models – without doing any programming at all.

Modelling

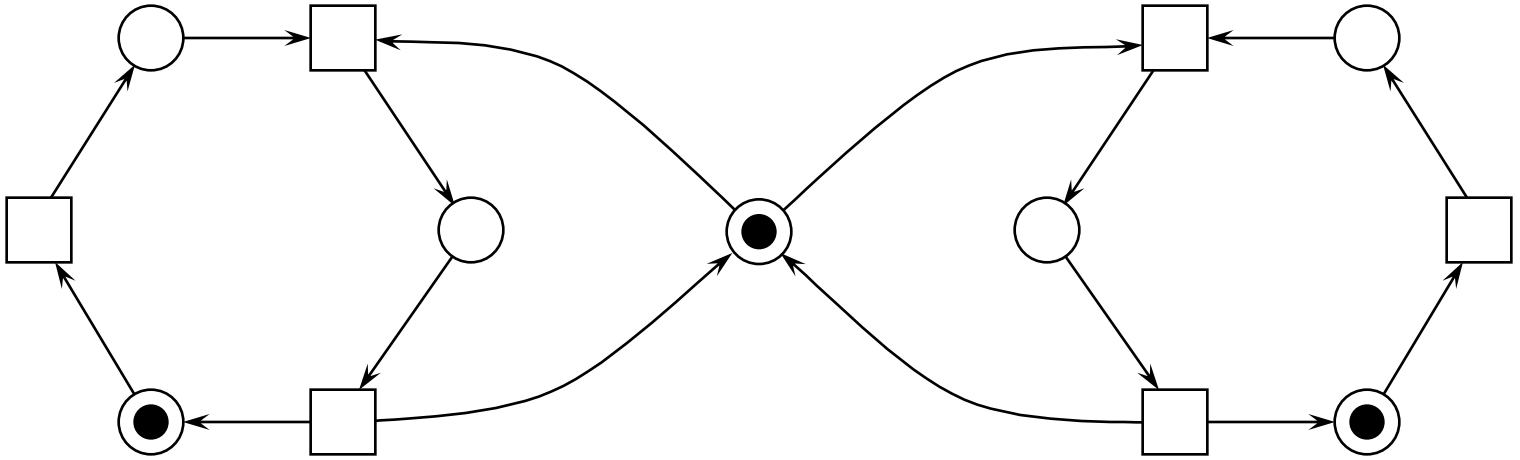
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

Models are the “floor plans” of software engineers, and are the key to the success of software projects.



A Model (Petri net)

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



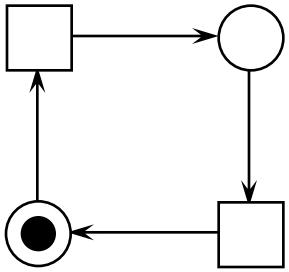
Stages

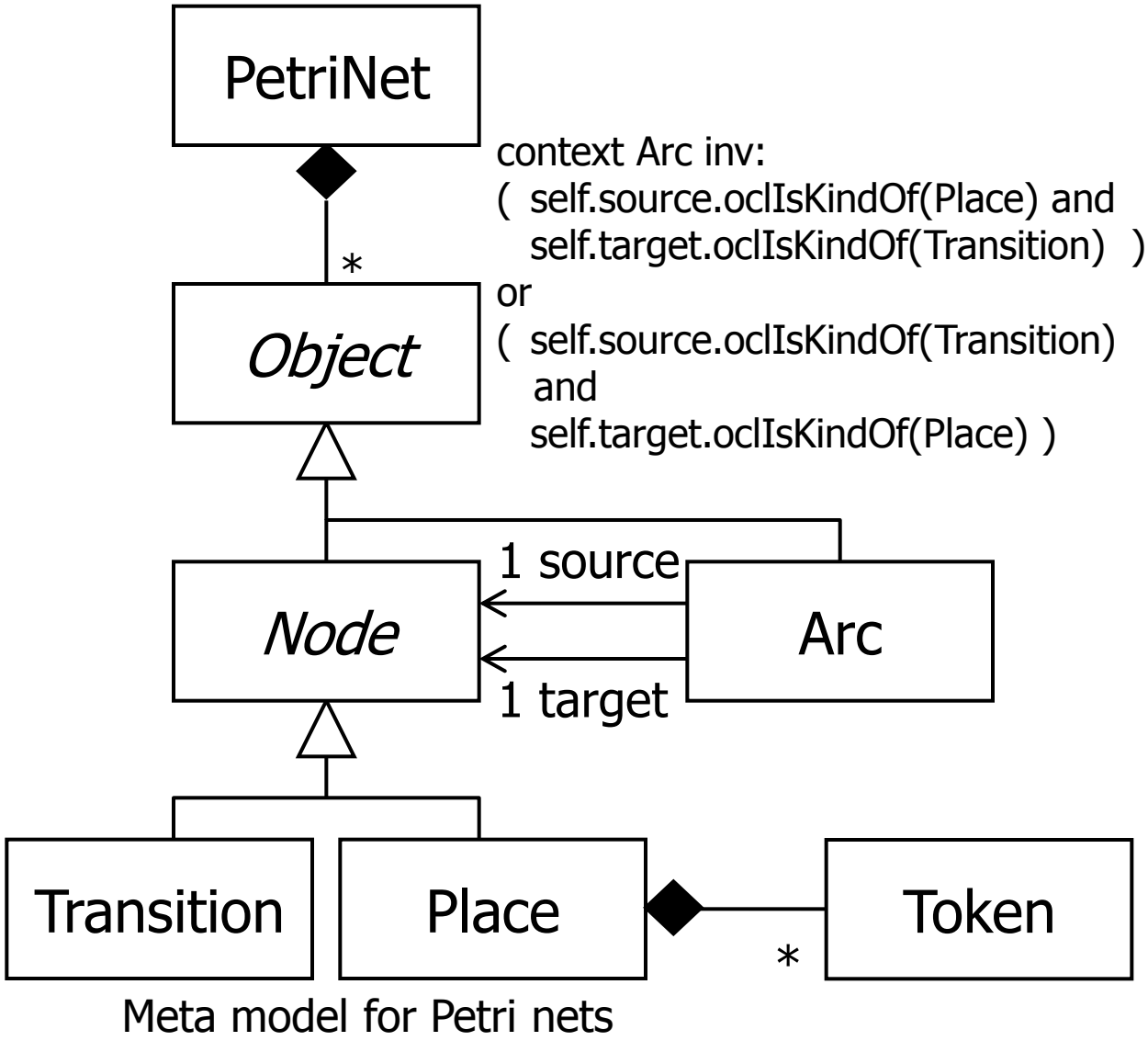
- Examples
- Taxonomy (done on blackboard)
- Glossary
- Model (see next slide)

Rule: Never ever start making a UML model without haven seen some examples first and naming the main concepts (taxonomy)!

Models and Meta Models

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



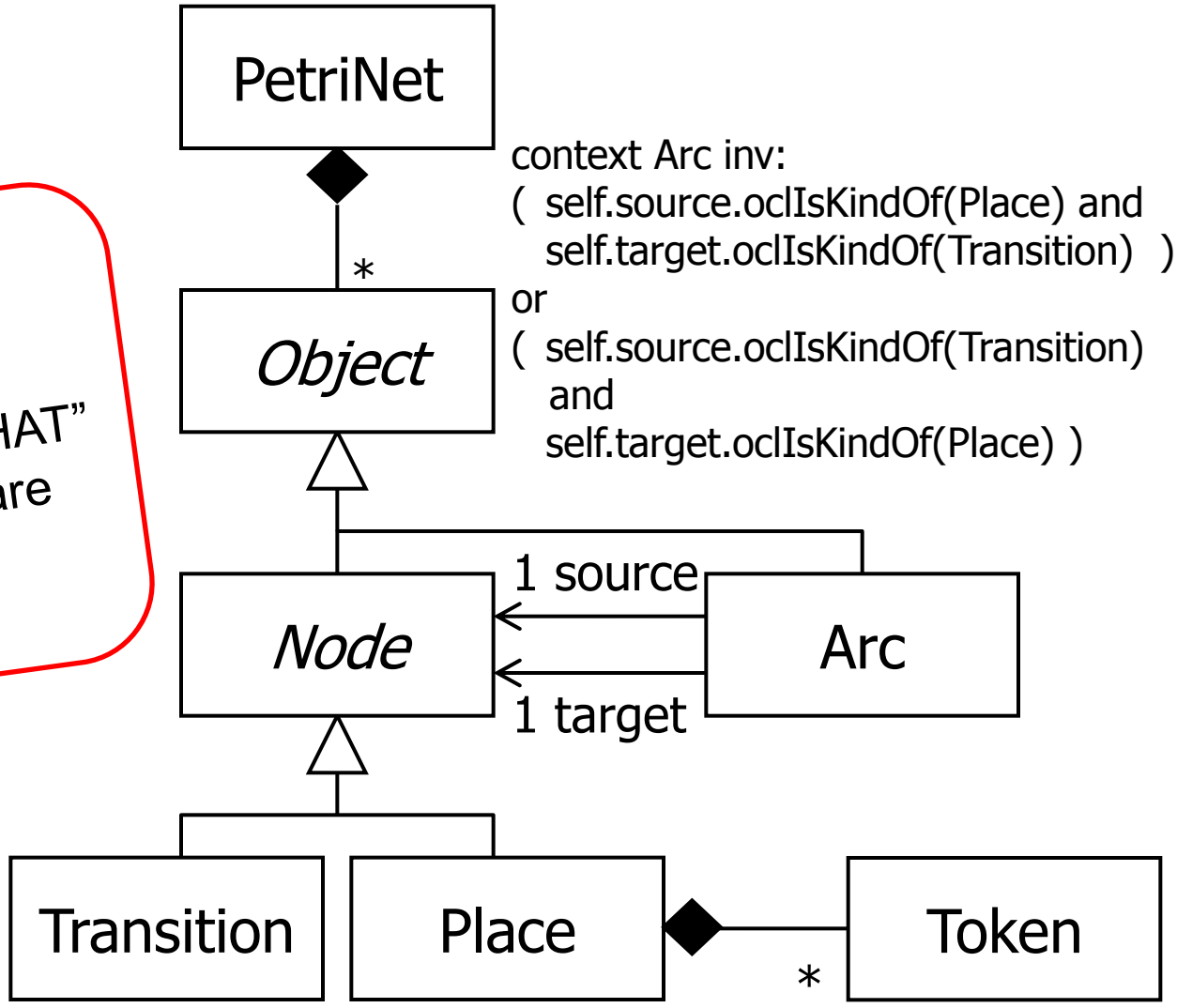


Meta model for Petri nets

Don’t think models as Java

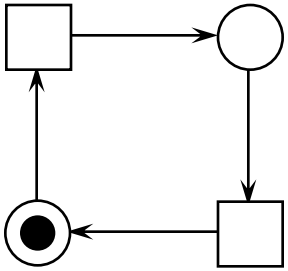
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

Rule: Don’t think of ”HOW” for now! These models are ”WHAT” only: which concepts are there in Petri nets?



Syntax (abstract and concrete)

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



graphical /
concrete syntax

Overview

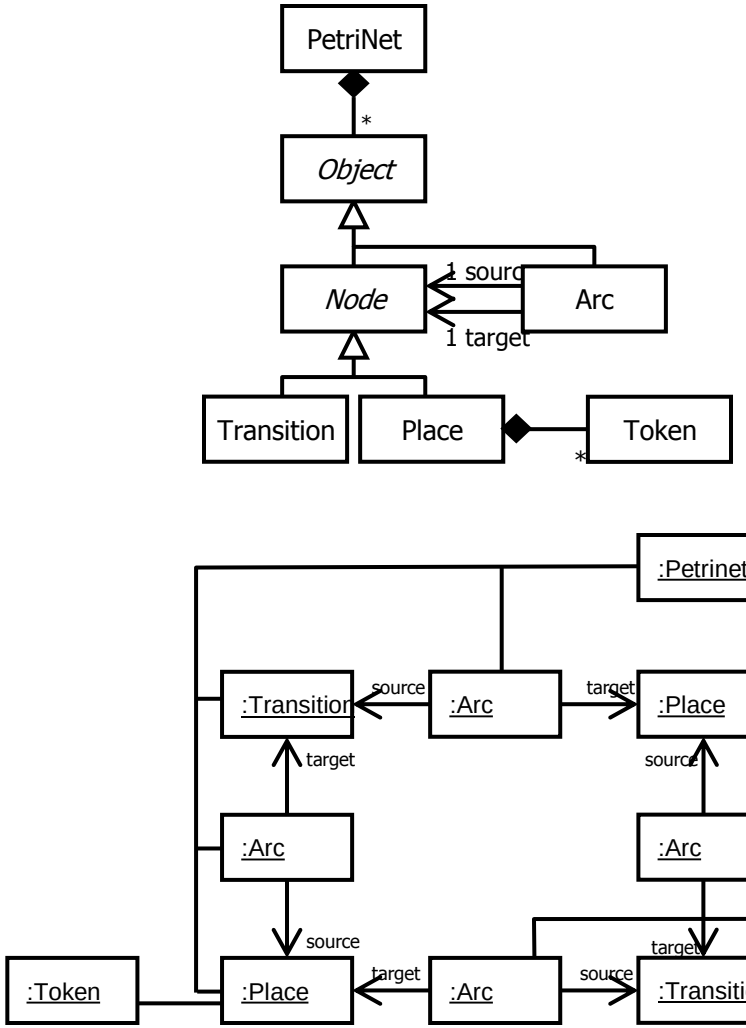
meta model

build-time

is an
instance of

model

runtime

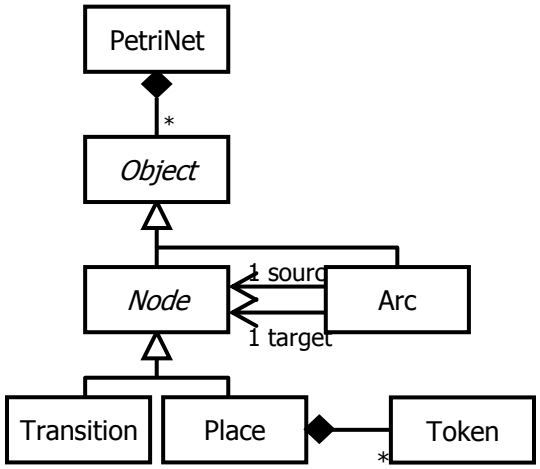


Benefits of Modelling

- Better understanding
- Mapping of instances to XML syntax (XMI)
- Automatic code generation
  - API for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)

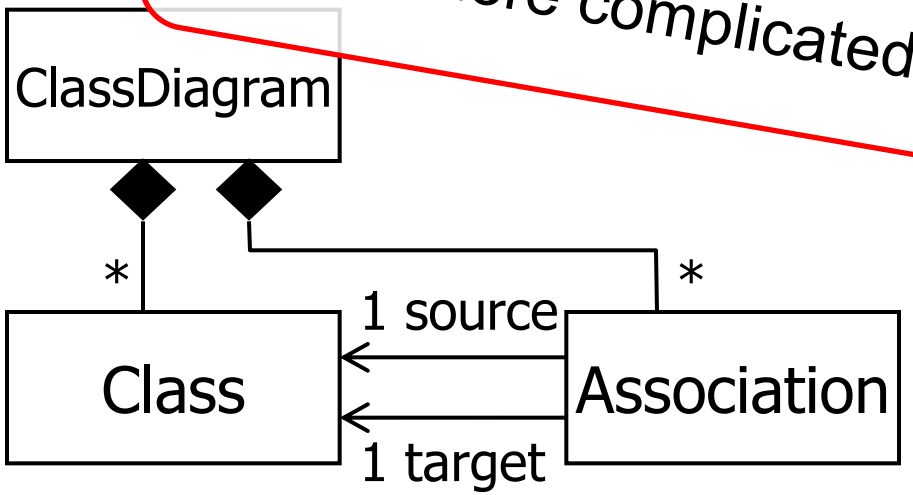
Class Diagrams are Models too

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler



UML model

Note: The real model of UML is much more complicated.

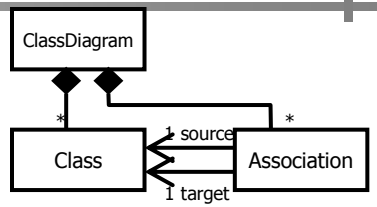


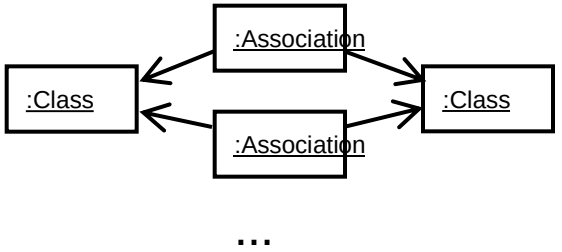
Meta model for UML (class diagrams)

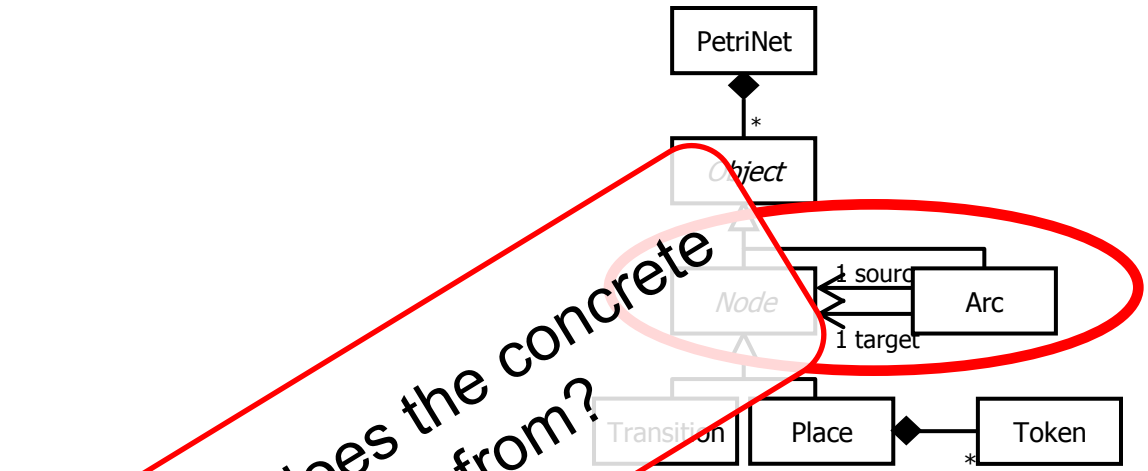
The term “meta” model makes more sense now!

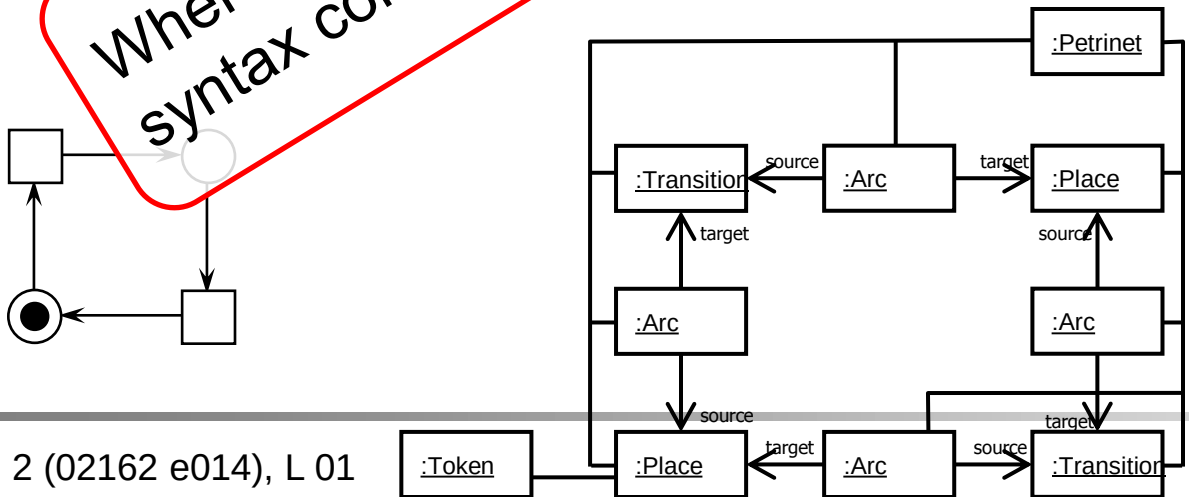
Different Meta-levels: MOF

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler









Answers:

- Program an editor

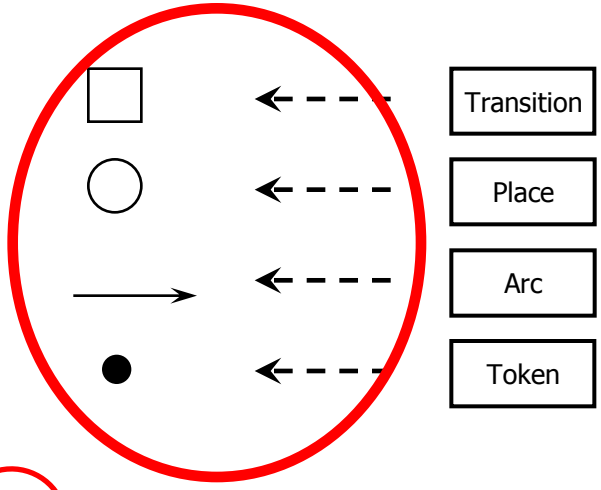
Not a good answer here!

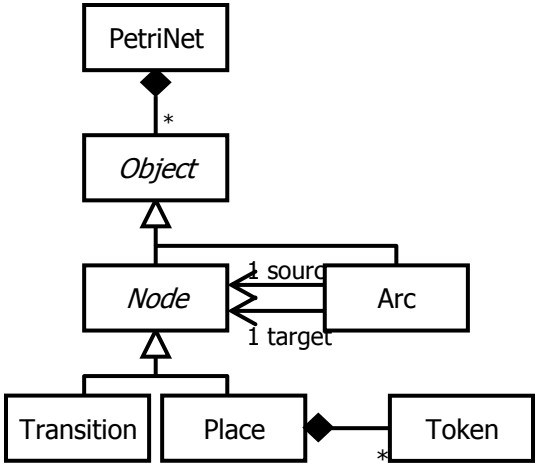
- Standard technology for mapping abstract to concrete syntax: EMF / GMF / EMFT

EMF/GMF Technology

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler

meta model

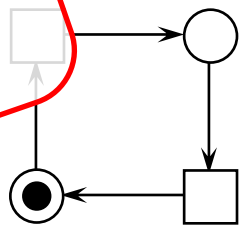


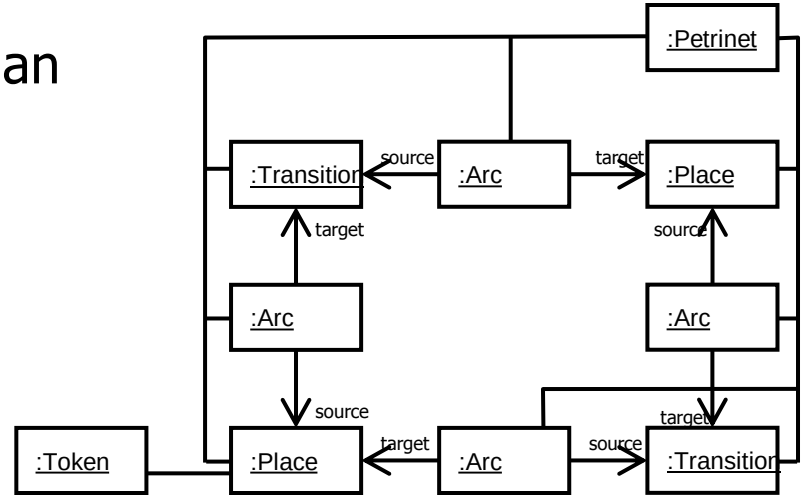


is instance of

generate an editor

This will not be covered in this year's course in detail.



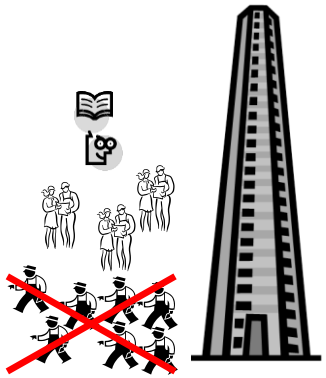


Benefits of Modelling (cntd.)

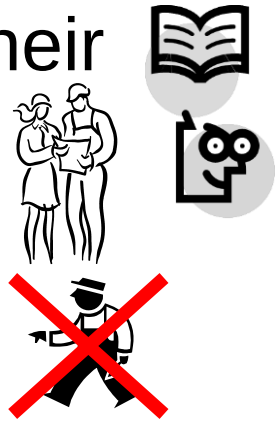
- Better Understanding
- Mapping of instances to XML syntax (XMI)
- Automatic Code Generation
  - API for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)
  - Editors and GUIs

Theses

- We will always have programming and programmers!



- We should always teach programming!
- But, software engineers should be trained in their engineering and modelling skills!
- And this is where they should be at their best!
- Most of the rest can be automated!



- Eventually, programming will be for software engineers as assembler is today for programmers.