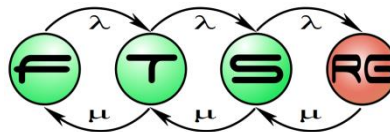


System Modelling

Course leader: Dr. PATARICZA András

Lecturer: HUSZERL Gábor

**Budapest University of Technology and Economics
Fault Tolerant Systems Research Group**



The Course

- Course (VIMIAA00)
 - 3 in 1 (in Hungarian/English/German)
 - students of B.Sc. in computer engineering (2nd semester)
- Dr. PATARICZA András
 - professor in charge
- GÖNCZY László
 - operative lead, organising
- HUSZERL Gábor (huszerl@mit.bme.hu)
 - English speaking lectures, ...

The English Speaking Course

- Lectures and lead
 - HUSZERL Gábor
- Hand-on Trainings
 - VÁRADY Balázs (Fridays), MONDOK Milán (Thursdays)
- No video or audio recording!
 - But take your own notes
- Web site
 - <https://inf.mit.bme.hu/en/edu/courses/remo-en>
 - Infos, news, slides, ...

The Course

- 14 lectures (Thursdays 10-12 AM, Q.BF10)
- 6 hand-on trainings
 - on odd weeks (beware of the spring holidays!)
 - Thursdays 2-4 PM in I.E320 (GA2)
 - Fridays 2-4 PM in I.B147 (GA1)
 - first time on the 21st/22nd February
 - tests (not an entry test!), presence mandatory
- 1 home assignment
- 2 mid-term exams (with entry tests)

Grading

- 2 mid-term exams (with entry tests): 35% + 35%
 - Fridays 8-10 AM: 29th March, 17th May (conflicts?)
- 1 home assignment: 30% of the mark
- All three of them with at least 40% of the points
 - mid-terms with passed entry tests
 - mid-term exams and home assignment can be repeated
- Optional additional points:
 - hand-on training tests (min. 4 tests out of 6): +5%
 - optional assignments for additional points

Home Assignment

- Mandatory submission
- Personal Assignment (different chess clocks)
- Deadlines
 - preparatory assignments: 3-5. semester week
 - release of assignments: 6. semester week
 - final submission: 12. semester week
- Electronic submission
 - hf.mit.bme.hu
- NO home assignments from earlier years accepted

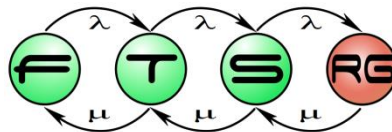
Course Topics

- Structure modelling
- Behaviour modelling
 - State based, process based
- Development of models
- Checking of models
- Performance modelling
- Visual data analysis
- Simulation
- Benchmarking, Code generation

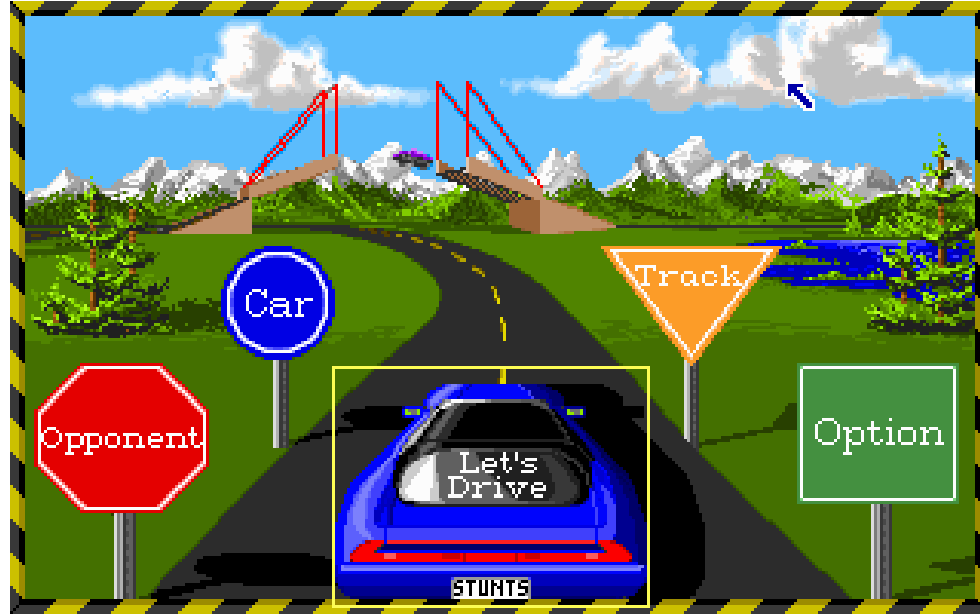
Foundations of Modelling

Dr. PATARICZA András, BERGMANN Gábor
HUSZERL Gábor

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



“Motivation”



- Stunts racing game, 1990.
 - Distinctive Games/Brøderbund Software
 - [https://en.wikipedia.org/wiki/Stunts_\(video_game\)](https://en.wikipedia.org/wiki/Stunts_(video_game))
- Racing + track creation (!)
 - „domain specific model”

“Motivation”: why to model?

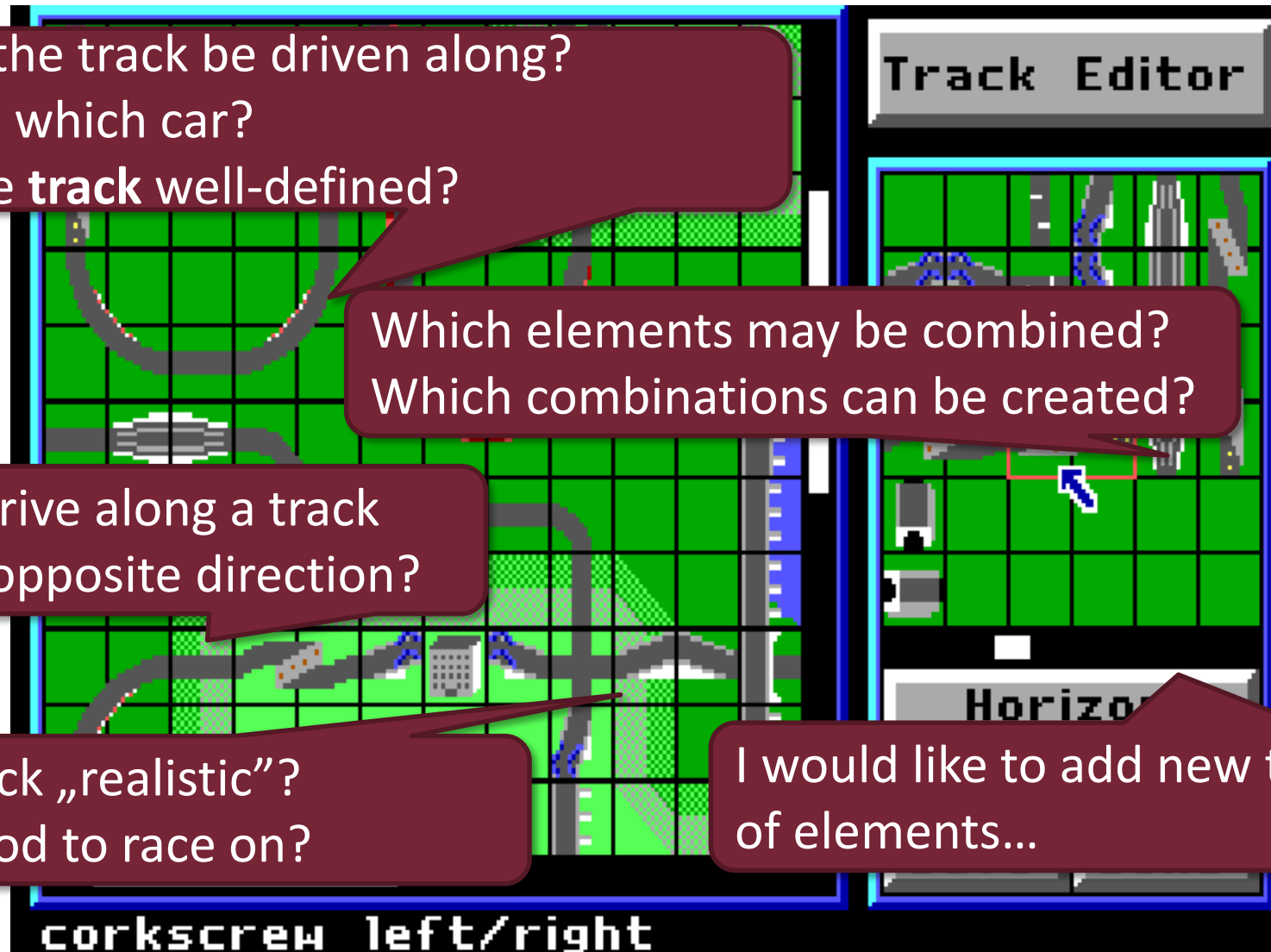
Can the track be driven along?
With which car?
Is the **track** well-defined?

Which elements may be combined?
Which combinations can be created?

Can I drive along a track
in the opposite direction?

Is a track „realistic“?
Is it good to race on?

I would like to add new type
of elements...



Pic: <http://www.abandonia.com/games/73/Stunts.htm>

Table of Contents

Models and
modelling

Purpose of the
models

Basic Concepts

Examples

Models and
modelling

Purpose of the
models?

Basic Concepts

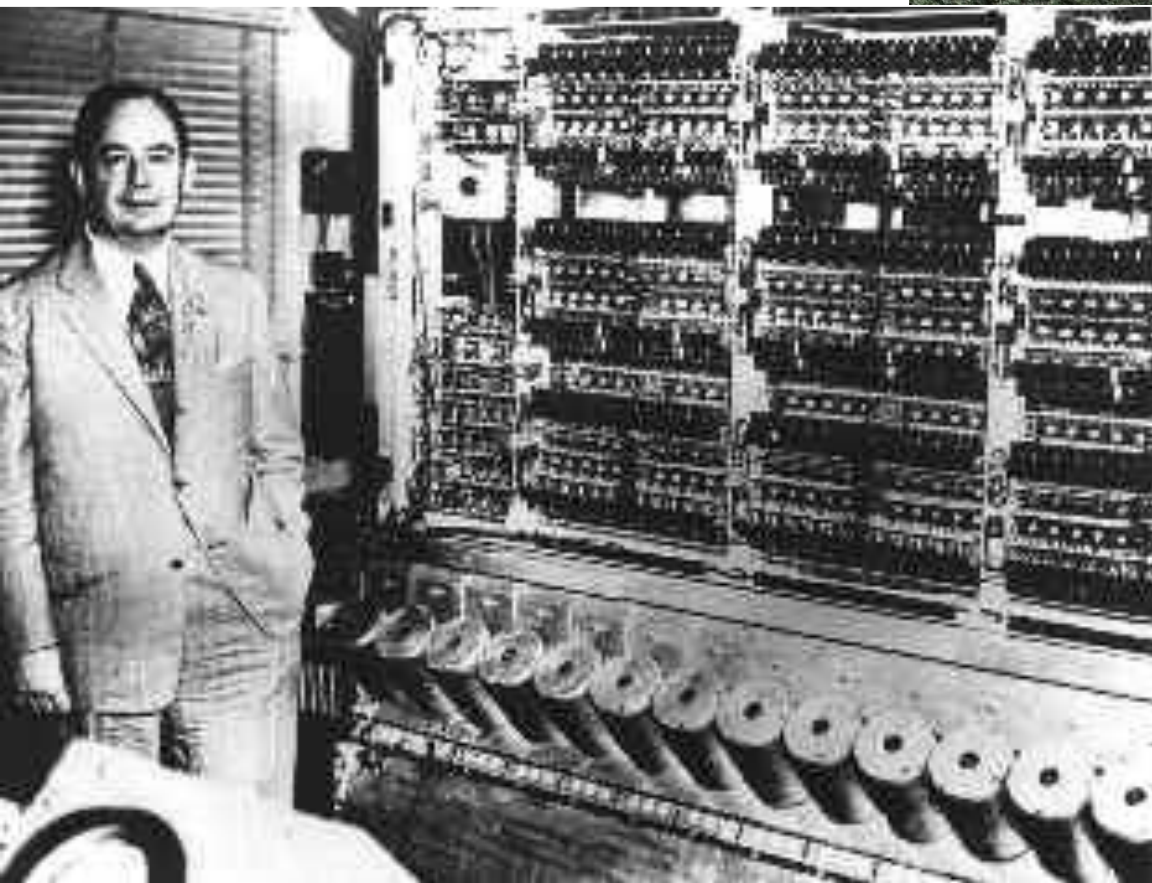
Examples

MODELS AND MODELLING

What is a model?

- "The sciences do not try to explain, they hardly even try to interpret, they mainly make models.
- By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena.
- The justification of such a mathematical construct is solely and precisely that it is expected to work – that is correctly to describe phenomena from a reasonably wide area.
- Furthermore, it must satisfy certain esthetic criteria – that is, in relation to how much it describes, it must be rather simple.,,

John von Neumann



E HÁZBAN SZÜLETETT
ÉS ÉLT 18 ÉVES KORÁIG
NEUMANN JÁNOS
1903 — 1957

A XX. SZÁZAD EGYIK LEGKIVÁLÓBB
MATEMATIKUSA.

AKI 1951 — 1952 — BEN
AZ AMERIKAI MATEMATIKAI
TÁRSULAT ELNÖKE VOLT.

AZ EMLÉKTÁBLÁT SZÜLETÉSÉNEK
100. ÉVFORDULÓJÁRA

A BOLYAI JÁNOS MATEMATIKAI
TÁRSULAT ÉS

AZ AMERIKAI MATEMATIKAI
TÁRSULAT KÖZÖSEN ÁLLÍTOTTA.



IN THIS HOUSE WAS BORN
AND LIVED UNTIL HE WAS 18

JOHN VON NEUMANN
1903 — 1957

ONE OF THE MOST OUTSTANDING
MATHEMATICIANS OF THE 20TH
CENTURY. PRESIDENT OF THE
AMERICAN MATHEMATICAL
SOCIETY IN 1951 — 1952.

THIS MEMORIAL PLAQUE WAS
ERECTED JOINTLY BY THE
JÁNOS BOLYAI MATHEMATICAL
SOCIETY AND THE AMERICAN
MATHEMATICAL SOCIETY ON THE
100TH ANNIVERSARY OF HIS BIRTH.

What is a model?

- Model is a **partial and simplified depiction of the real or a hypothetical world (the „system“)**, of which can be **substituted** for specific viewpoints
- Decisions:
 - Which part of the world?
 - What is simplified?
 - How it can be mapped to the world?
- Benefits
 - smaller (finite)
 - has manageable size

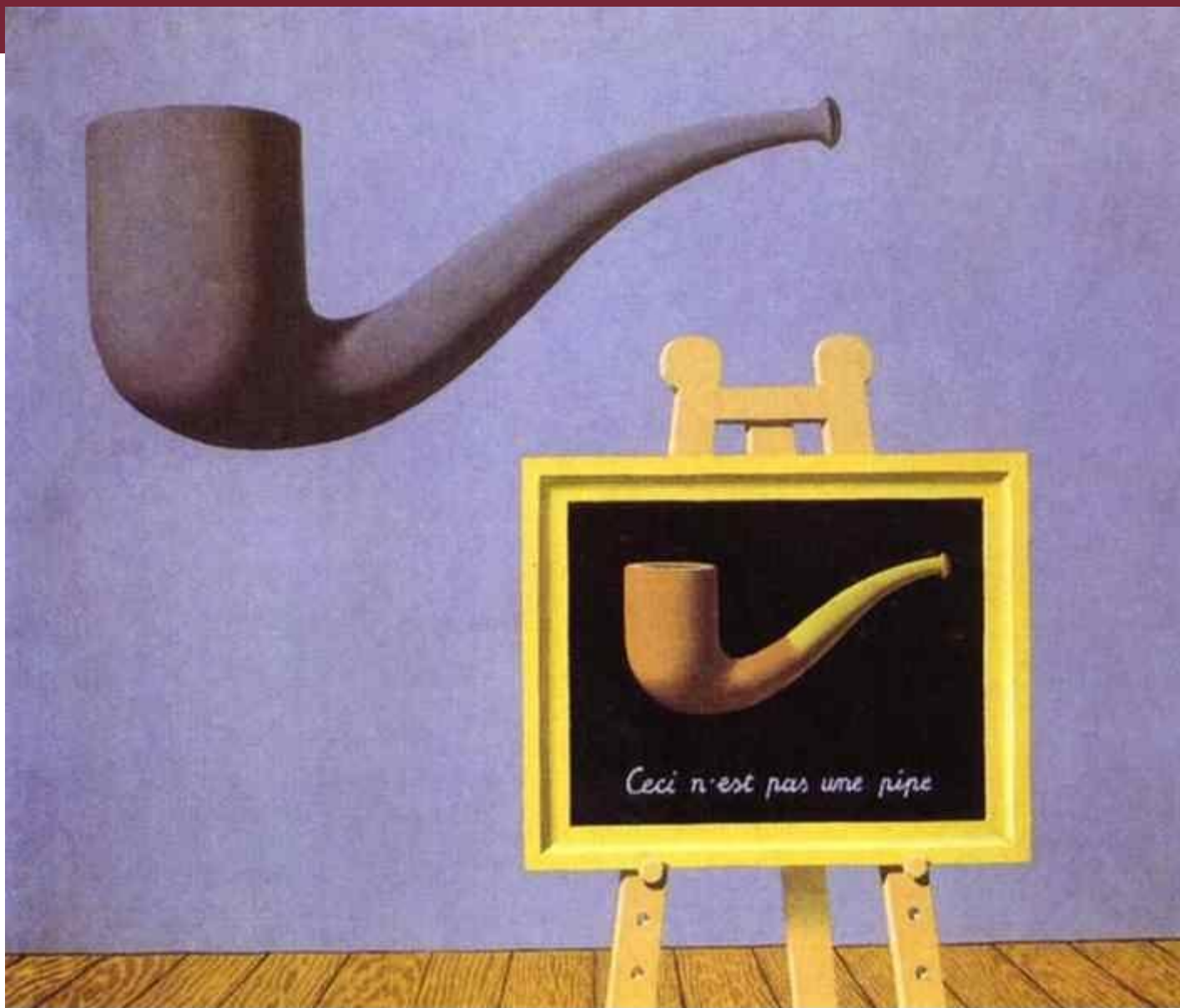
When is it possible or recommended to use?

What is NOT a model?

- The model is NOT the reality!



- The model is not a diagram.
 - That is only a view ...



abcgallery.com - Internet's biggest art collection

Model vs. Real World



Mathematical Models vs. Real World

- All of the models are:
closed world
 - Effects, factors
 - Parameters
 - Validity
- The models have no defined behavior outside its scope
- Not all aspects can be expressed in advance
 - *Human decisions*
 - *Generated models*
- *Validation of the solutions*
- *Constructed to answer questions...*

- Normal operations
 - Scope:
 - Enough materials are available
 - **All** of the orders can be completed in time
 - Target function:
 - Minimal cost
- Corner case
 - Scope:
 - Not enough materials are available
 - Target function :
 1. **As many as possible** orders completed in time
 2. Minimal cost

Example: safety critical SW

- Braking of an airplane: wheel brake + thrust



1993 Warsaw: Lufthansa 2904

- (SW) protection:

(load on both wheels)

OR

(one wheel is rotating fast)

→ *(plane on the ground)*

→ **(PILOT CAN BRAKE)**

Wheel sliding

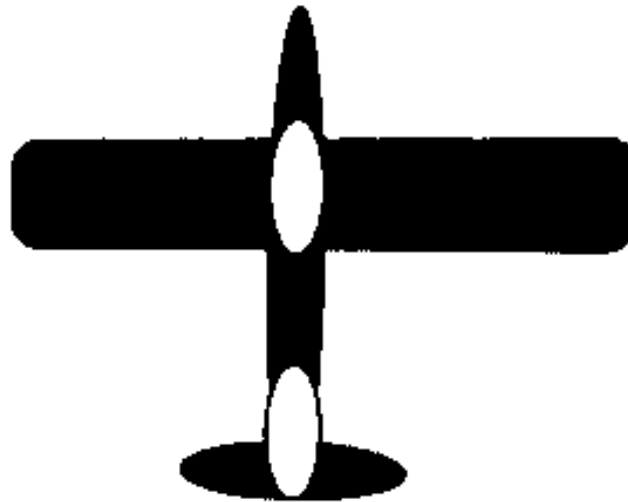
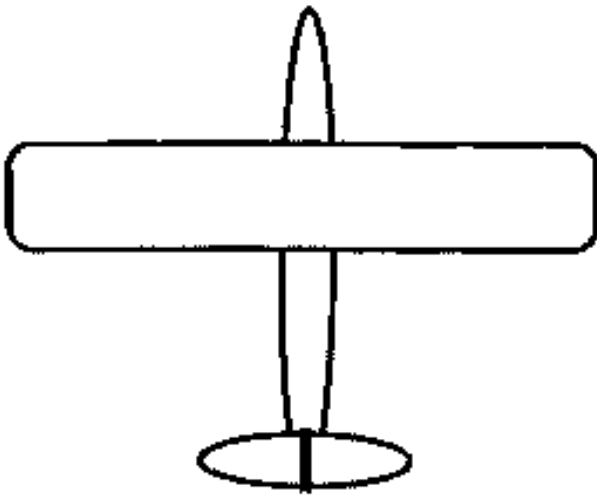
Wheel in the air



Model quality

- Reality: **open world** $\leftrightarrow$
Model: closed world
- “Faithfulness” of models:
 - Probable + critical cases
- Implementation of a bad model can be deadly...

Armor? → Abraham Wald



Armor? → Abraham Wald



**Model is valid only with
environment ...**



Models and
modelling

Purpose of the
models

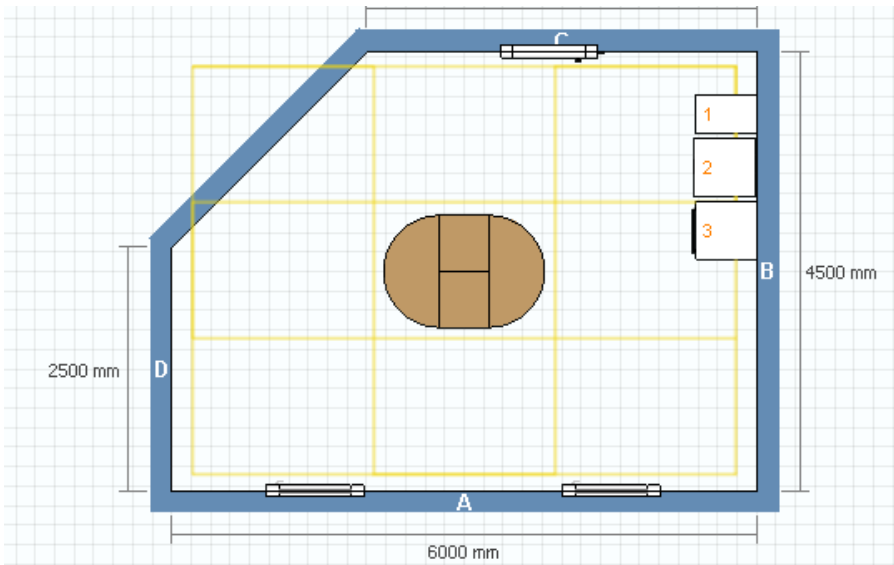
Basic Concepts

Examples

PURPOSE OF THE MODELS

Modelling in real life?

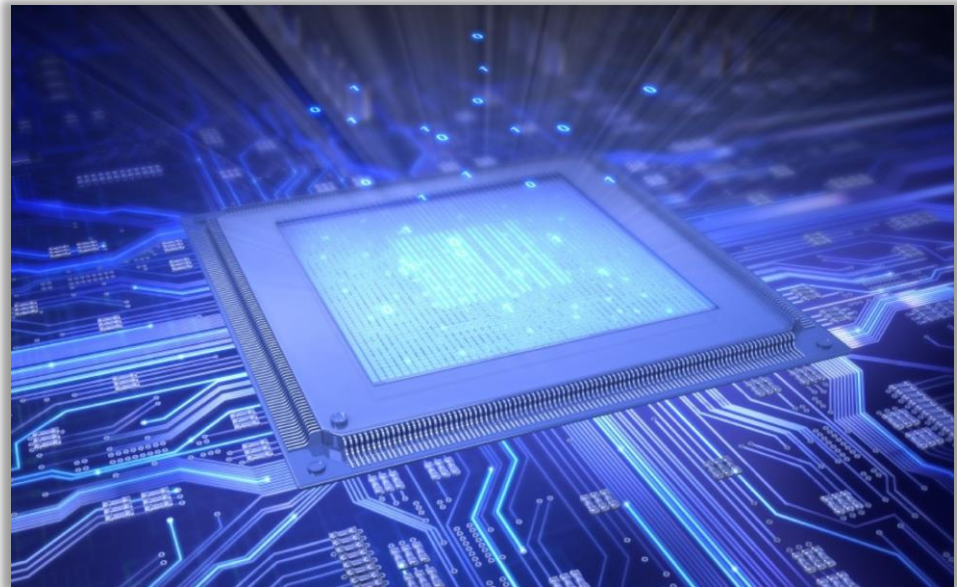
E.g.: web-based kitchen designer
[of a Scandinavian company]



This is also a modelling language!

- VHDL, Verilog – domain specific hardware description

```
1
2 ...
3 ARCHITECTURE Struct OF MyLogic IS
4
5     COMPONENT And2 IS
6         PORT (x, y: IN std_logic;
7             f: OUT std_logic);
8     END COMPONENT;
9
10    COMPONENT CustomHW IS
11        PORT (x: IN std_logic;
12            f: OUT std_logic);
13    END COMPONENT;
14
15    SIGNAL n1, n2: std_logic;
16
17 BEGIN
18     And2_1: And2 PORT MAP ( , , );
19     And2_2: And2 PORT MAP ( , , );
20     CustHW: CustomHW PORT MAP ( , );
21 END Struct;
22
```



What is the point of modelling?

- I am developing software. Should I do modelling?
 - Yes! In fact, you already do!
 - (The source code is a kind of model...)
 - Most importantly: **mental models**
 - When should be these models documented?
 - Role of the model: **communication**
 - Human → Human
 - Human → Machine
 - Machine → Machine
 - Human → Human (themselves in the future)
 - E.g. Why we implemented this algorithm this way some years ago...

Modelling language

- The purpose of the model is to help the communication
 - Understanding of the model is necessary
 - **Modelling languages** (when do we need them?)
- Syntax **How do I describe the model? „e.g. in SystemC”**
 - „Mathematical structure”: abstract syntax
 - Notation: concrete syntax
 - graphical symbols / textual format
- Semantics **What does the model mean? „i++”?**
- Boundary conditions, constraints
 - Syntactical correctness, well-formedness
 - Design conventions (can vary by teams)

Basic Concepts - Metamodelling

- Modelling language: what are the types of the objects?
 - ... what can be the connection between the objects?
 - ... how the types can be related to each other?
- **Metamodell** = model of a modelling language
- Well known examples
 - Database table
 - XML document
 - Relational database schema
 - UML object diagram
 - ...

Basic Concepts - Metamodelling

■

M

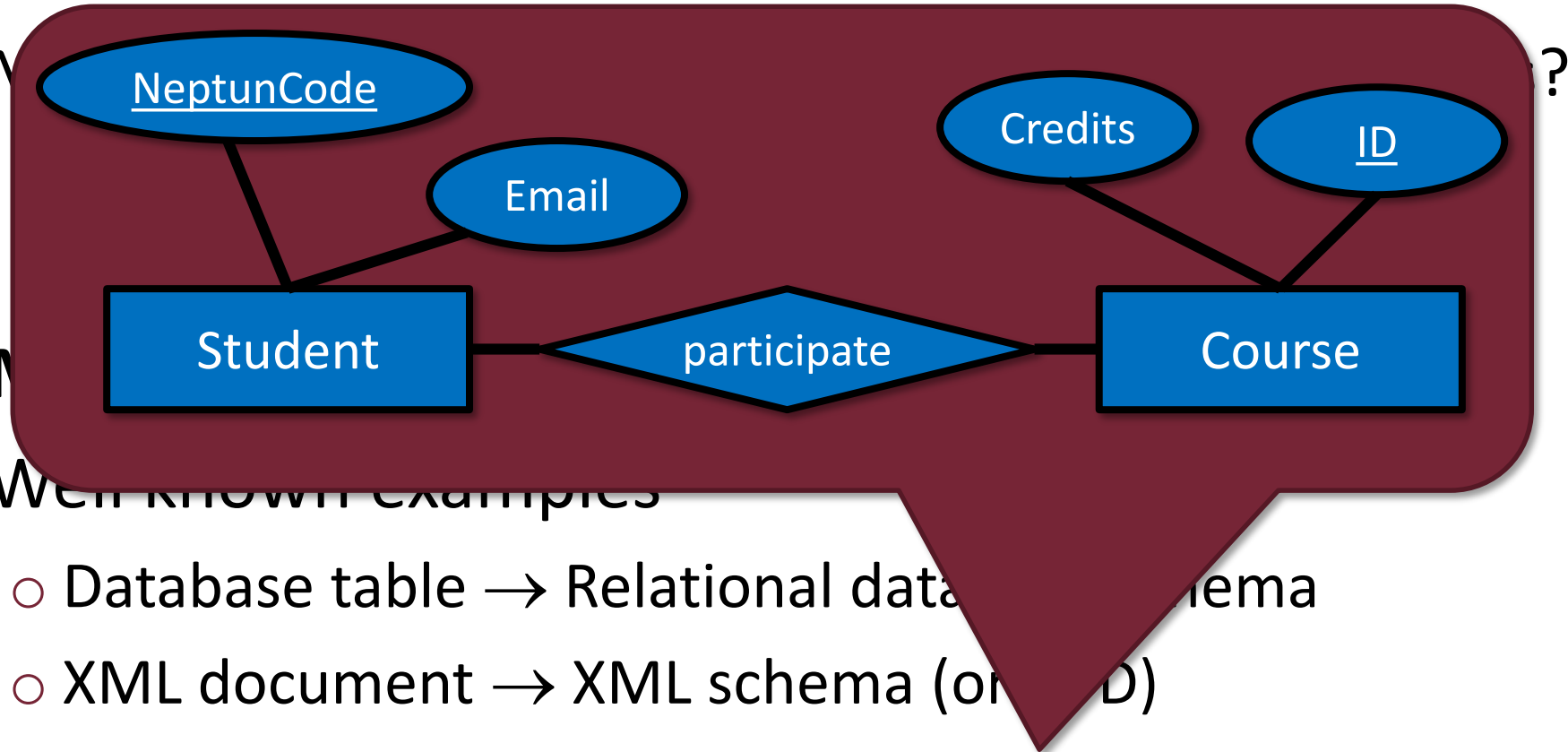
■

M

■

Well known examples

- Database table → Relational data model schema
- XML document → XML schema (or DTD)
- Relational database schema → ER model
- UML object diagram → class diagram
- ...



Models in Engineering Jobs

■ Engineering Jobs

(Architects, Mechanical Eng., Civil Eng., Electro Eng., ..., Landscape Architects, ...)

- Concept
 - Description
 - Refinement
 - Dimensioning
 - Checking
 - Implementation
- Drawing models
 - Model refinement
 - Model refinement
 - Model analysis & simulation
 - Implementation, derivation

It is the widely used engineering approach: **Planning**

System Design Process

Management

- purchasing, supplies
- Planning, operations, assessment

ANSI/EIA 632
standard

System
Design

- Requirement definitions
- Solution definitions

Engineering tasks

Product
Development

- Implementation
- Usability

These steps
are usually
supported by
models

Quality
Check

- System Analysis
- Requirements validation
- System Verification
- End-product Verification

System Design Process - Analogy

Management

System
Design

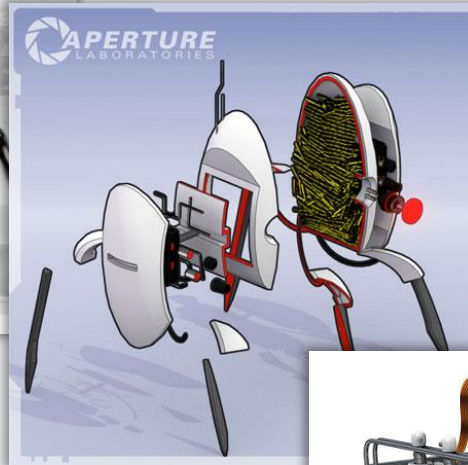
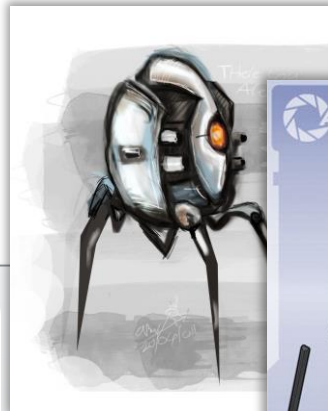
Product
Development

Quality
Check

Validation: Are we building the
right product?
Verification: Are we building the
product right?

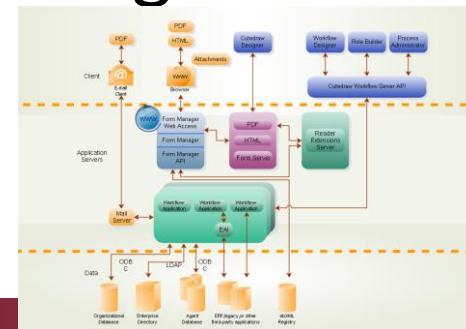
PORTAL 2

Engineering Tasks



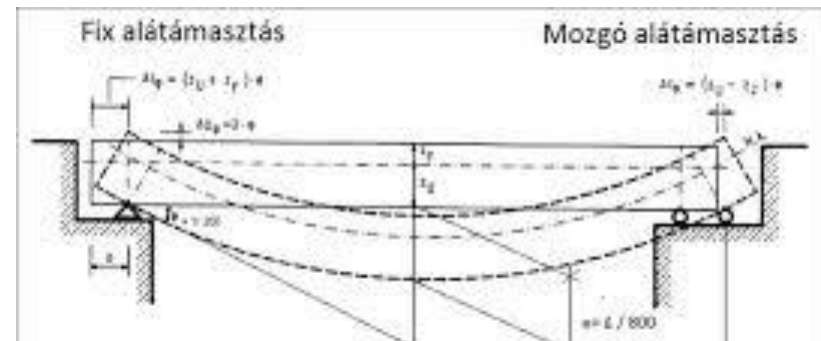
Application – Documentation

- The model is simpler to handle
 - Easier to communicate than the whole reality
 - Can be refined step-wise (explained later)
- Communication, Visualization
 - Demonstrations (explained later)
 - Easy to understand textual language
 - Expressive diagram
- Supporting conceptualization, design
 - Similar viewpoints
 - „thinking tool“



Application - Analysis

- Human contribution or (semi-)automated
- Method
 - Review, static analysis
 - Dynamic state space exploration – model checking
 - Proving theorems about the model
- Goal
 - Detecting errors (best effort)
 - Proving safety properties (stronger requirement!)
 - Calculating properties of the model (e.g. timing)



Usage of models - Simulation

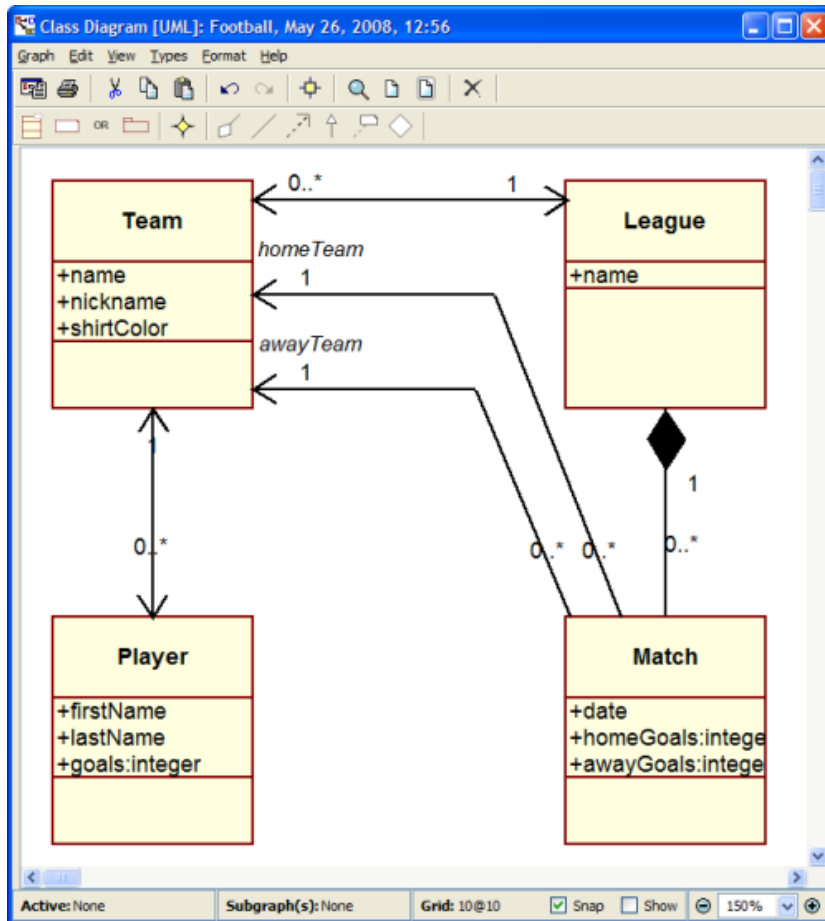
- Demonstration
 - A tool for communication
- Validation
 - „I built it correctly... but did I build the right thing?“
- Experiment
 - Proof for desired system characteristics
 - Measurement of quantitative properties
 - Substitute expensive real-life experiments
 - Properties that cannot be predicted using formulae



Application – Deriving Artifacts

- Human contribution or (semi-)automated
- Output
 - Generating program code, analysable language, etc.
 - Another model
 - Refinement, next design (abstraction) phase
 - Partial view
 - Model integration
 - May preserve model properties

Web Application Development



League [Gears] - Mozilla Firefox

File Edit View History Bookmarks Tools Help

League [Gears]

Enter a League to store in the database:

name

OK

Teams name

Matches date

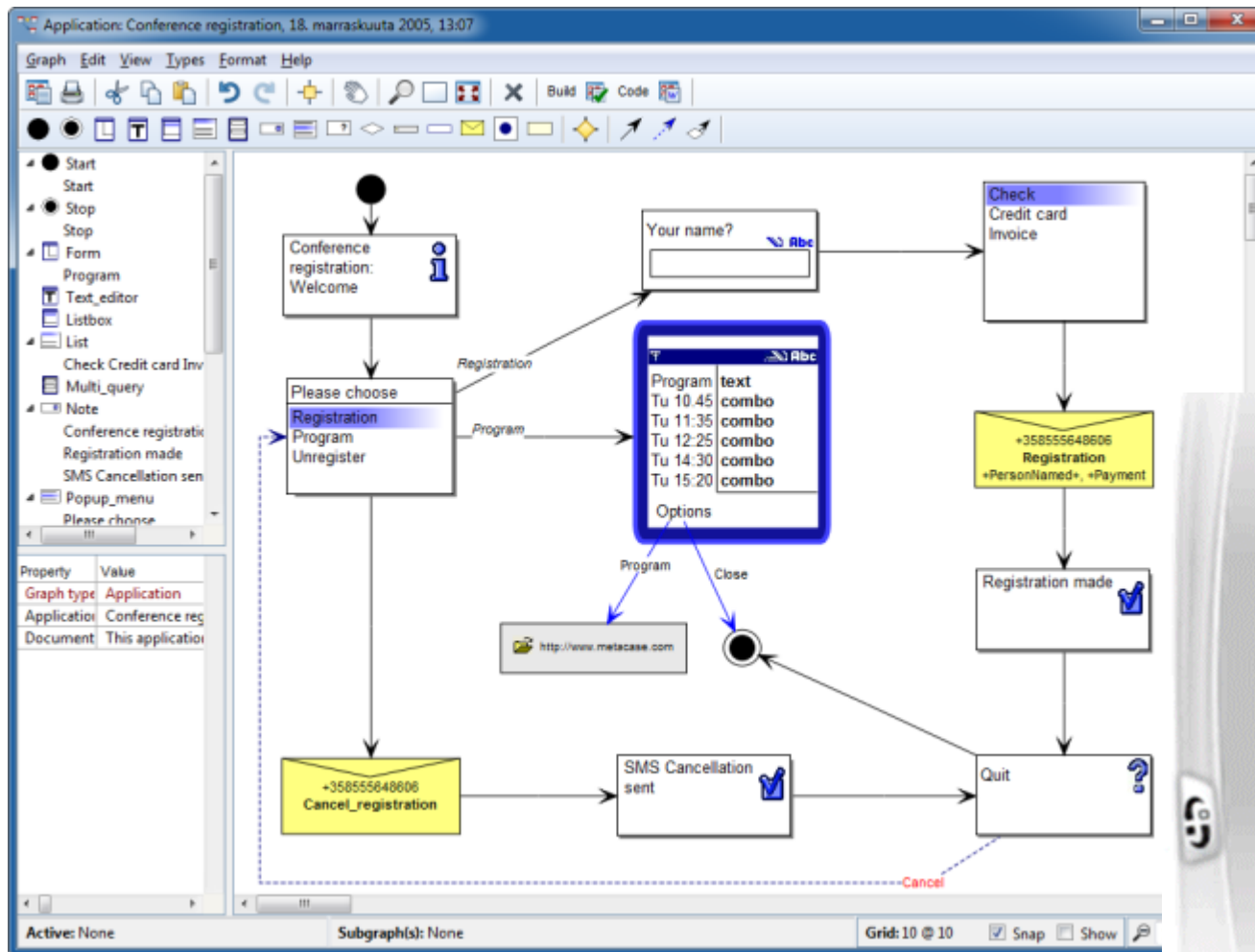
4 most recently edited League entries:

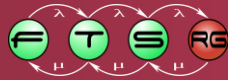
name	
Edit	Premiership Delete

[Back to top page of Football application.](#)

This page uses Gears to record your entries on the local disk. If you navigate away and revisit this page, all your data will still be here. Try it!

Smartphone App Development





Reverse Engineering

League [Gears] - Mozilla Firefox

File Edit View History Bookmarks Tools Help

League [Gears]

Enter a League to store in the database:

name

OK

Teams name

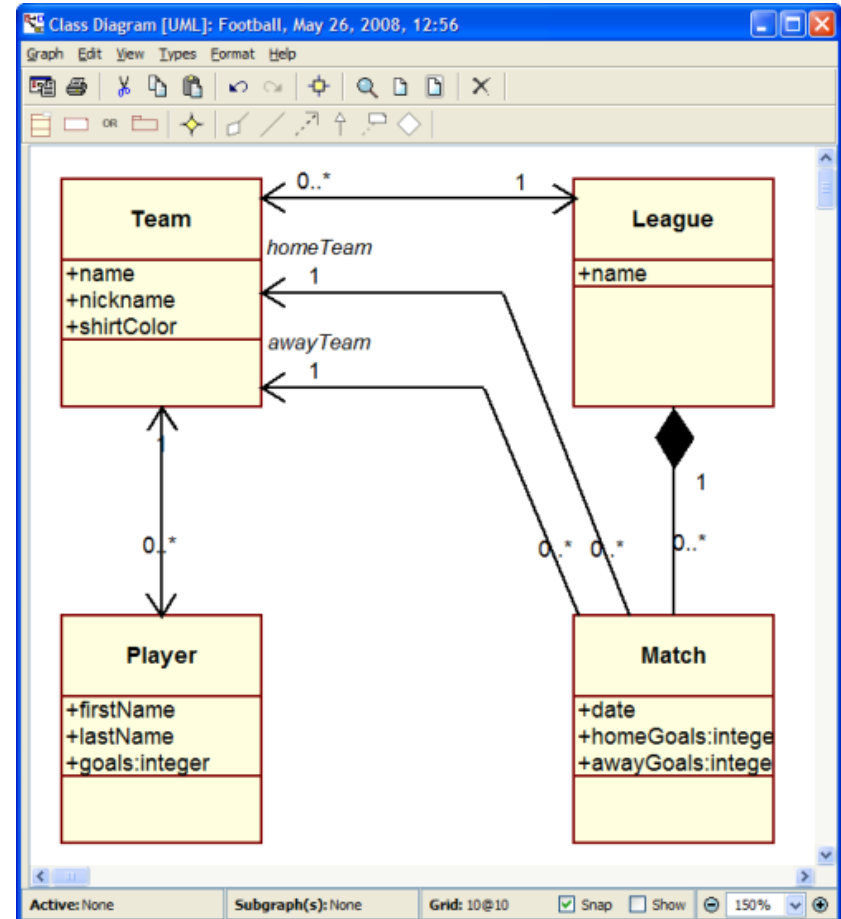
Matches date

4 most recently edited League entries:

name
Edit Premiership Delete
New View All Delete All

[Back to top page of Football application.](#)

This page uses Gears to record your entries on the local disk. If you navigate away and revisit this page, all your data will still be here. Try it!



Model and
modelling

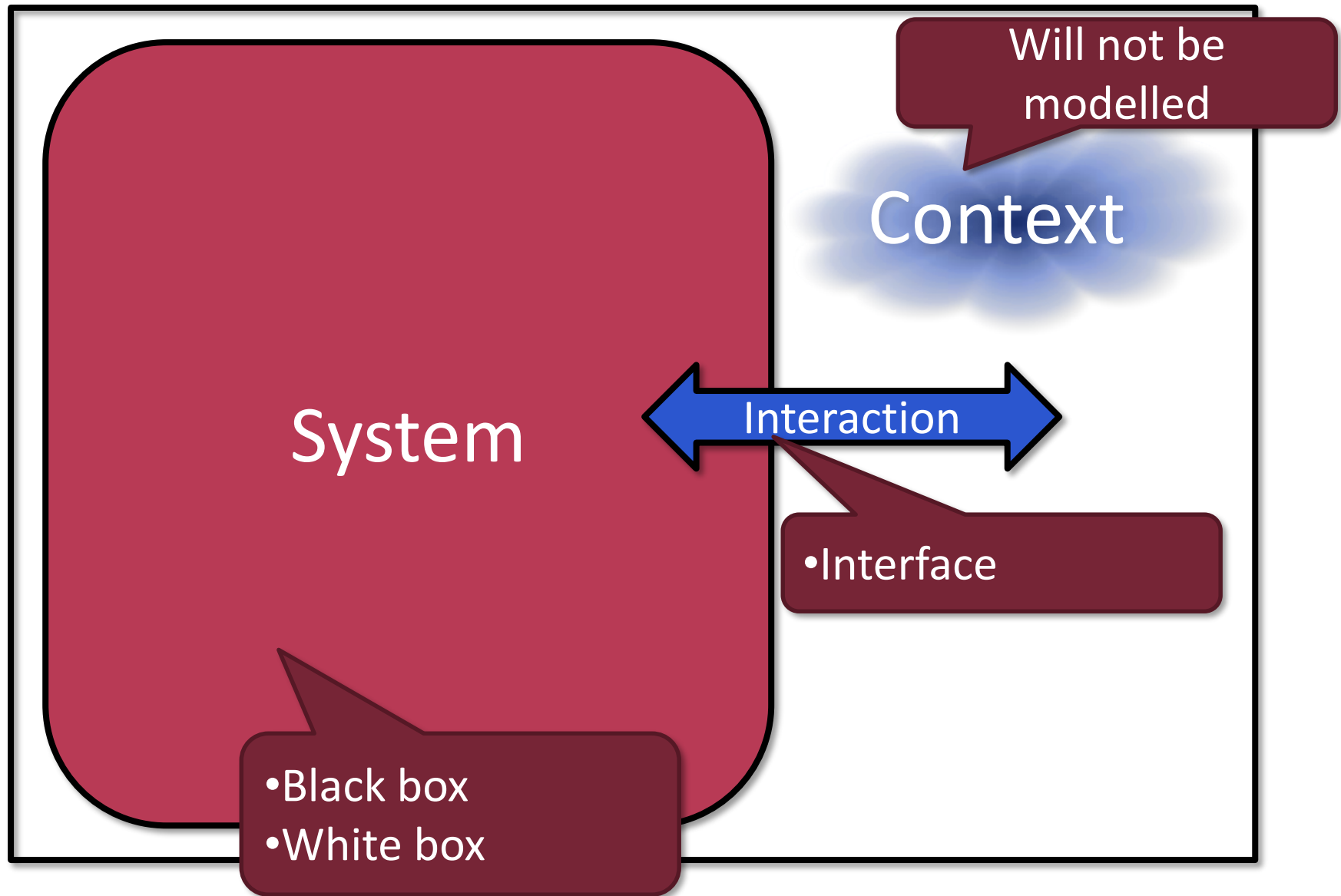
Purpose of the
models?

Basic Concepts

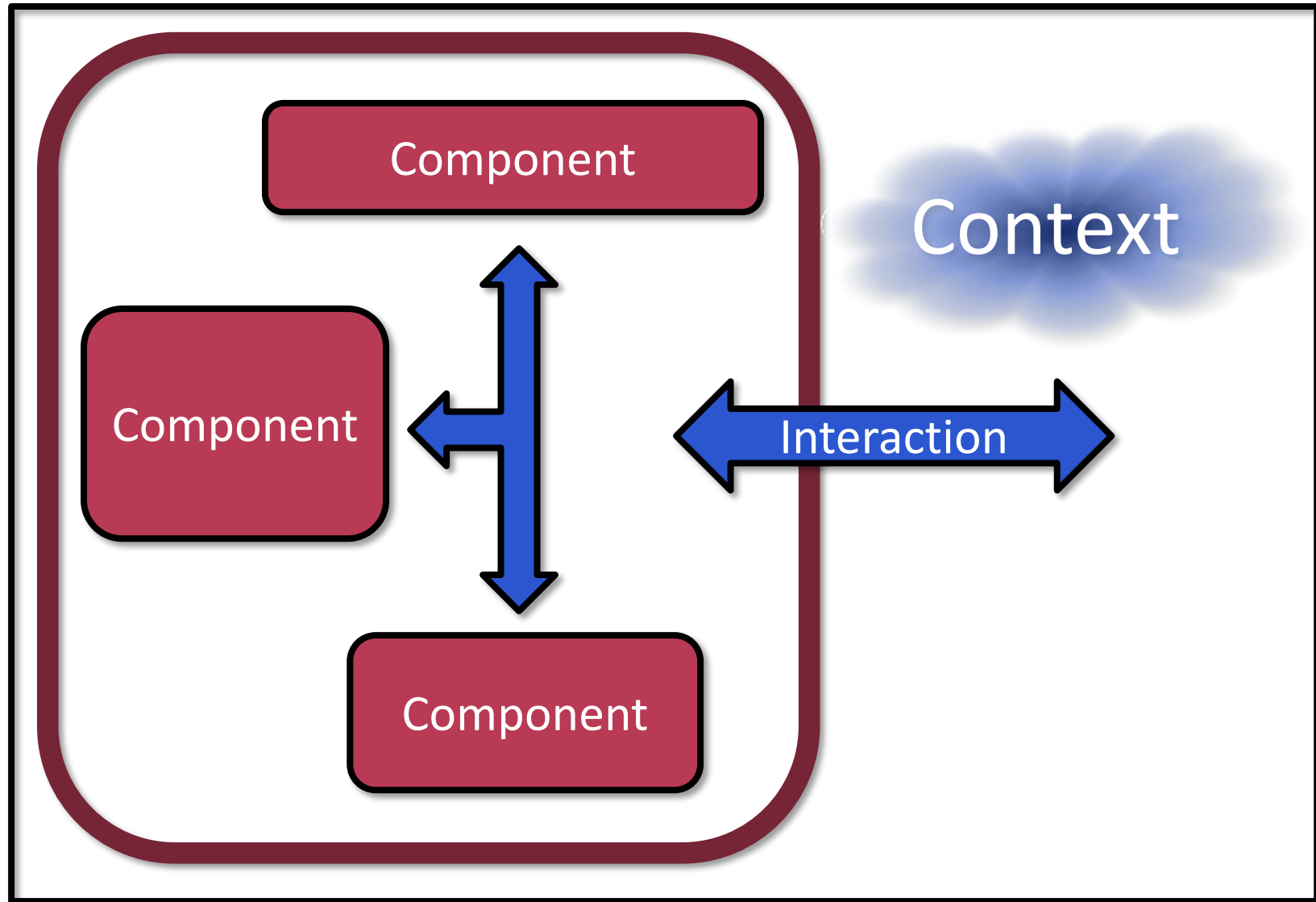
Examples

BASIC CONCEPTS

Basic concepts – System and Context



Basic concepts – System and Context



Basic Concepts – Refinement/Abstraction

- *Refinement*: adding further details to the model...
- ...keep the abstraction of the input model
- Inverse: *(vertical) abstraction*
- There was a *hierarchical refinement* on the previous slide
 - „refinement of the boxes”
- Other aspects can be refined as well...
 - E.g. partitioning of a set (variable domain refinement)
 - **Good** / **bad** can be refined with:
 - **Fast** / **average** / **slow** / **incomplete** / **dangerous**

Basic Concepts - Refinement

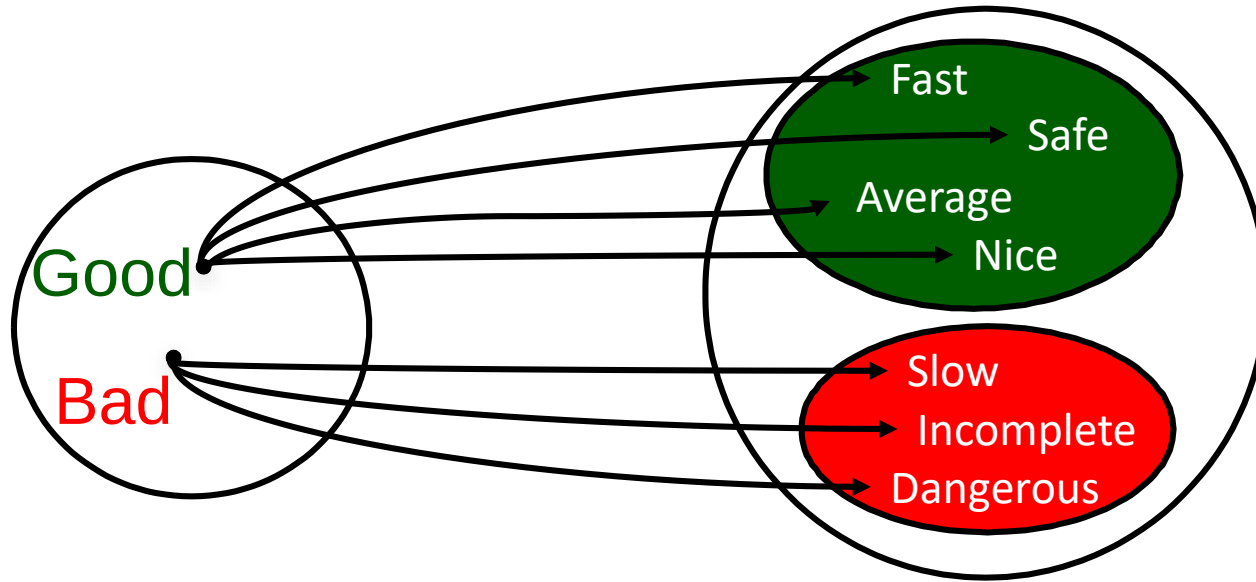


Basic Concepts - Refinement



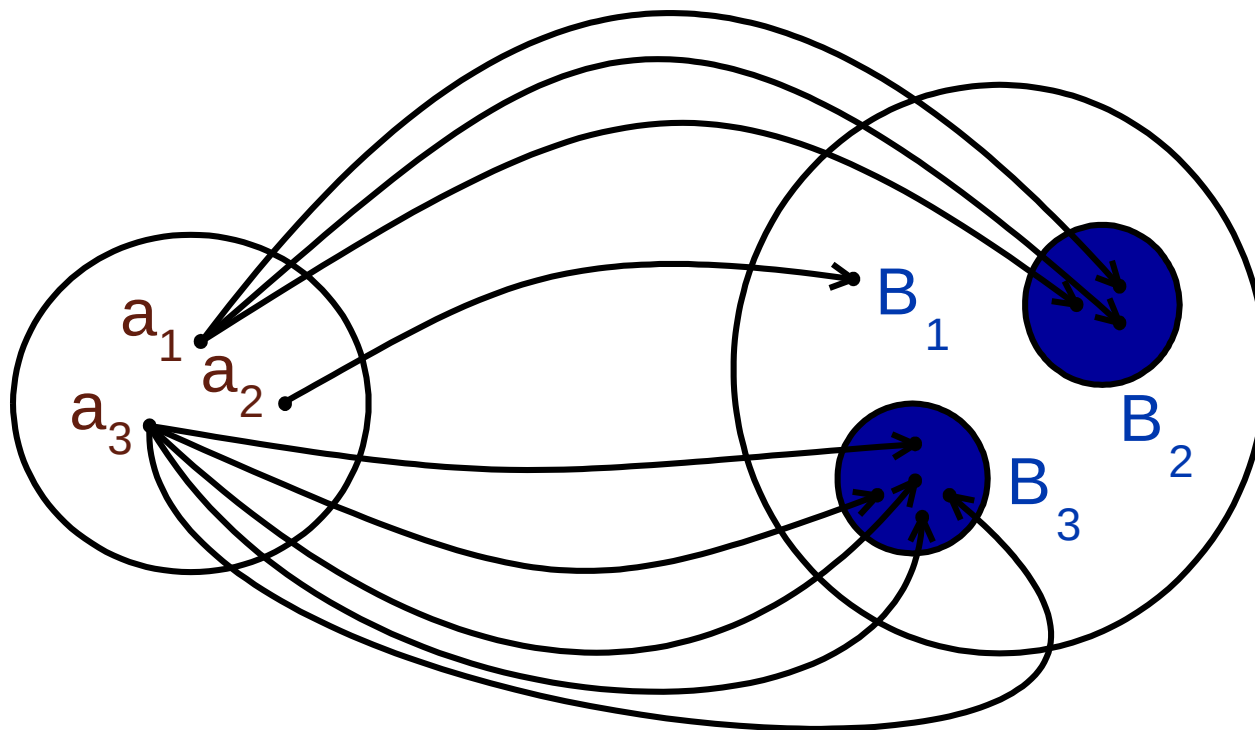
Partitions of Set Elements

Mapping of disjunct subsets to elements of a set



Partitions of Set Elements

Mapping of disjunct subsets to elements of a set



$$\forall a_i, a_j \in A, R(a_i) \subset B \mid \forall a_i, a_j R(a_i) \cap R(a_j) = \emptyset$$

Abstraction

- The abstraction is **correct**, if to each element of the original system an element of the abstract system can be assigned.
- The abstraction is **complete** in point of a given property, if each property of the system is kept that is relevant for the given property.

(Several definitions are possible.)

Example: Refactoring

- Ordering numbers
 - Ascending: 18,23,131,247,1925
 - Descending: 1925,247,131,23,18
 - Alphabetically: 131,18,1925,23,247
- Code (selecting the minimum):

```
void ordering_ascending(int* array, int size){  
    int i, j;  
    for(i = 0; i<size; i++){  
        int minimumindex = i;  
        for(j= i + 1; j<size; j++){  
            if(array[j] < array[minimumindex])  
                minimumindex = j;  
        }  
        int tmp = *i;  
        *i = *minimumindex;  
        *minimumindex = tmp;  
    }  
}
```

What is the difference in the code, if another ordering is required?

Example: Refactoring

- Ordering numbers
 - Ascending: 18,23,131,247,1925
 - Descending: 1925,247,131,23,18
 - Alphabetically: 131,18,1925,23,247
- Code (selecting the minimum):

```
void ordering_ascending(int* array, int size){  
    int i, j;  
    for(i = 0; i<size; i++){  
        int minimumindex = i;  
        for(j= i + 1; j<size; j++){  
            if(array[j] < array[minimumindex])  
                minimumindex = j;  
        }  
        int tmp = *i;  
        *i = *minimumindex;  
        *minimumindex = tmp;  
    }  
}
```

What is the difference in the code, if another ordering is required?

Example: Refactoring

- Idea: Put the comparison into a separate function

```
void ordering(int* array, int size, bool (*compare)(int,int)) {  
    int i, j;  
    for(i = 0; i<size; i++){  
        int minimumindex = i;  
        for(j= i + 1; j<size; j++){  
            if( (*compare)(array[j],array[minimumindex]) )  
                minimumindex = j;  
        }  
        int tmp = *i;  
        *i = *minimumindex;  
        *minimumindex = tmp;  
    }  
}
```

What more part of the code can be reused?

Example: Refactoring

- Idea: Put the comparison into a separate function

```
void ordering(int* array, int size, bool (*compare)(int,int)) {  
    int i, j;  
    for(i = 0; i<size; i++){  
        int minimumindex = i;  
        for(j= i + 1; j<size; j++){  
            if( (*compare)(array[j],array[minimumindex]) )  
                minimumindex = j;  
        }  
        int tmp = *i;  
        *i = *minimumindex;  
        *minimumindex = tmp;  
    }  
}
```

What more part of the code can be reused?

→ Avoiding the duplication of code

The Role of Abstraction in Coding

■ Note:

- Algorithm of „ordering by selecting”: an abstraction
- Different comparison functions ($\rightarrow$ orderings): instantiations

■ Clean code:

- Algorithms with parameters
- Different goals $\rightarrow$ different instances
 - e.g. the same algorithm can be used for ordering words

Model and
modelling

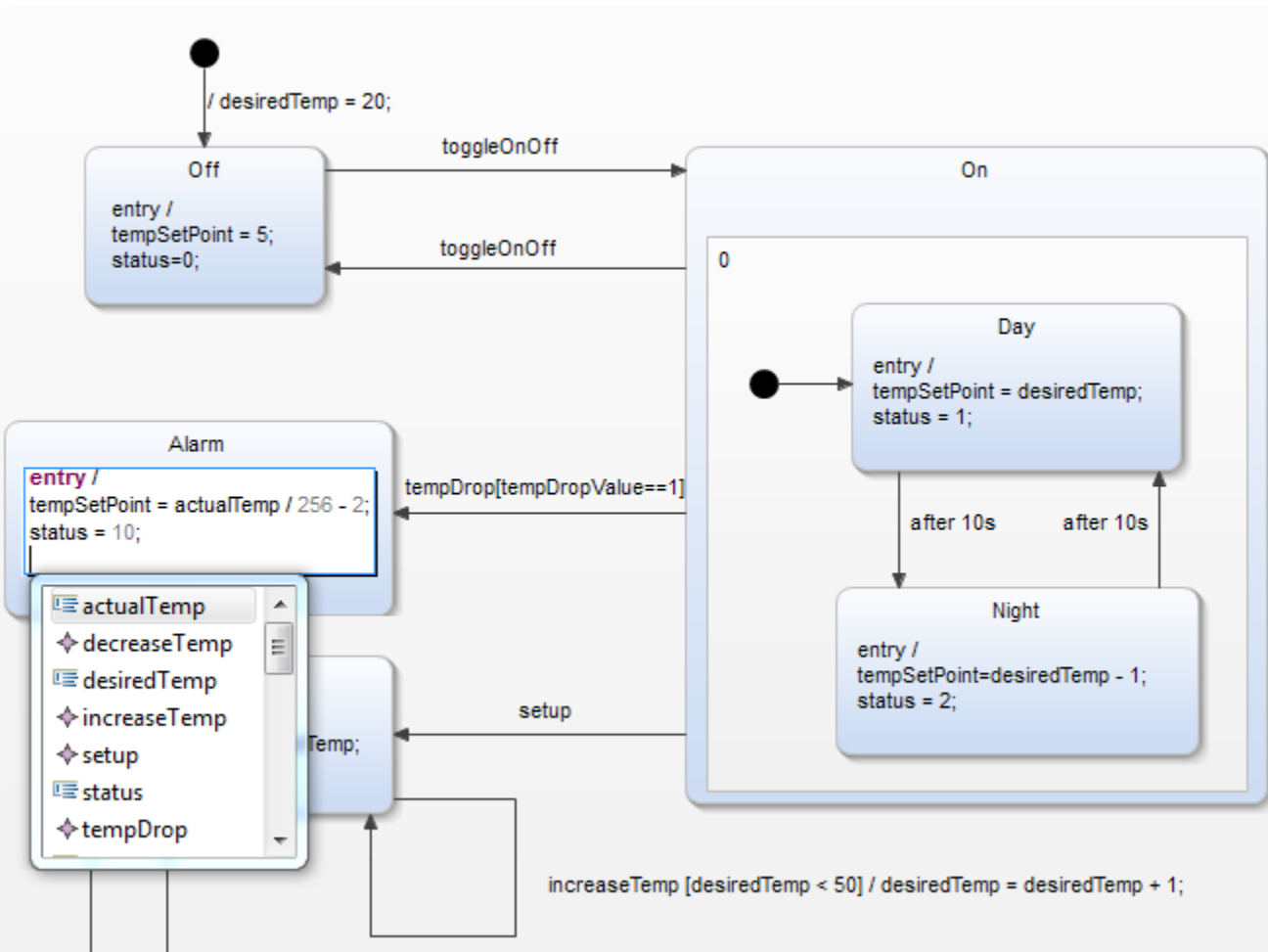
Purpose of the
models

Basic Concepts

Examples

EXAMPLES

Yakindu – State Charts

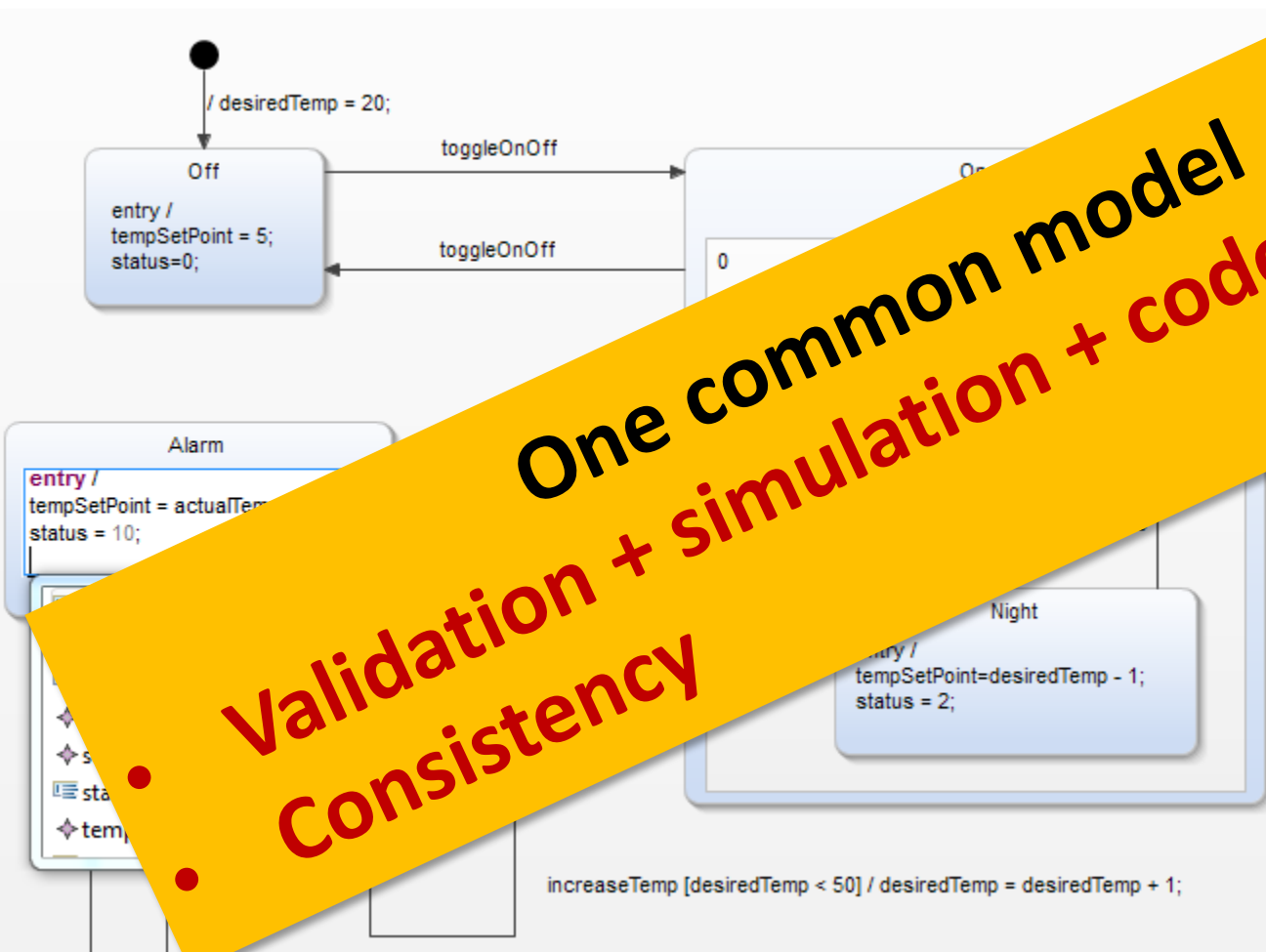


```
import java.io.*;
import java.net.*;
import java.security.*;
import protection;

public class Client {
    public void sendAuthentication(String username, String password) throws IOException {
        OutputStream outStream = new DataOutputStream(new DataOutputStream(new Socket(host, port)));
        long t1 = Math.random();
        double q1 = Math.random();
        byte[] protected1 = ProtectionUtil.protect(q1, t1);
        long t2 = Math.random();
        double q2 = Math.random();
        byte[] protected2 = ProtectionUtil.protect(q2, t2);
        out.writeUTF(username);
        out.writeInt(protected1.length);
        out.write(protected2);
        out.flush();
    }
}

public static void main(String[] args) {
    String host = args[0];
    int port = 7999;
    String user = "John";
    String password = "shh";
    Socket s = new Socket(host, port);
    Client client = new Client();
    client.sendAuthentication(user, password);
}
```

Yakindu – State Charts



```
public static void main(String[] args) {
    String host = args[0];
    int port = 7999;
    String user = "John";
    String password = "shh";
    Socket s = new Socket(host, port);

    Client client = new Client(s);
    client.sendAuthenticat
```