Software Intensive Systems: Dealing with Complexity

Hans Vangheluwe

McGill
School of Computer Science

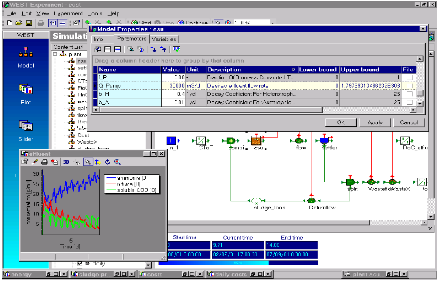
necsis

AnSyMo

Universiteit Antwerpen

Department of Mathematics
and Computer Science

MSDL
Modelling, Simulation and Design Lab



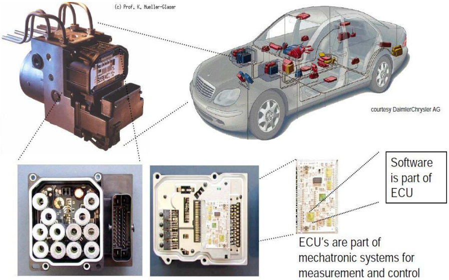
WEST Experiment - cost

Model Properties : asu

Drag a column header here to group by that column

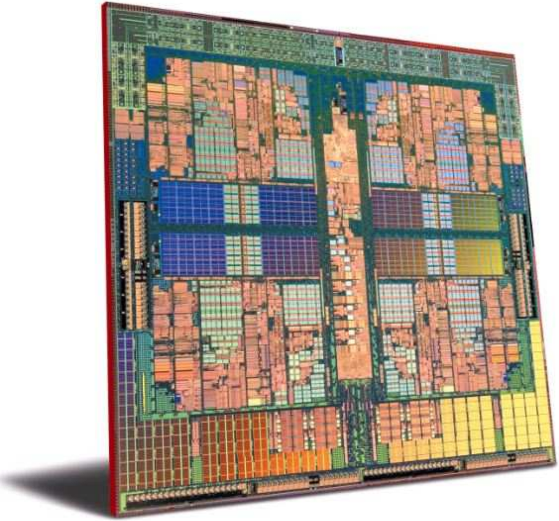
Name	Value	Unit	Description	Lower bound	Upper bound	File
f_P	0.00	-	Fraction Of Biomass Converted T...	0	1	
Q_Pump	30000	m3/d	Desired effluent flow rate	0	1.79769313486232E308	
b_H	0.4	1/d	Decay Coefficient For Heterotroph...	0	25	
b_A	0.01	1/d	Decay Coefficient For Autotrophic...	0	25	

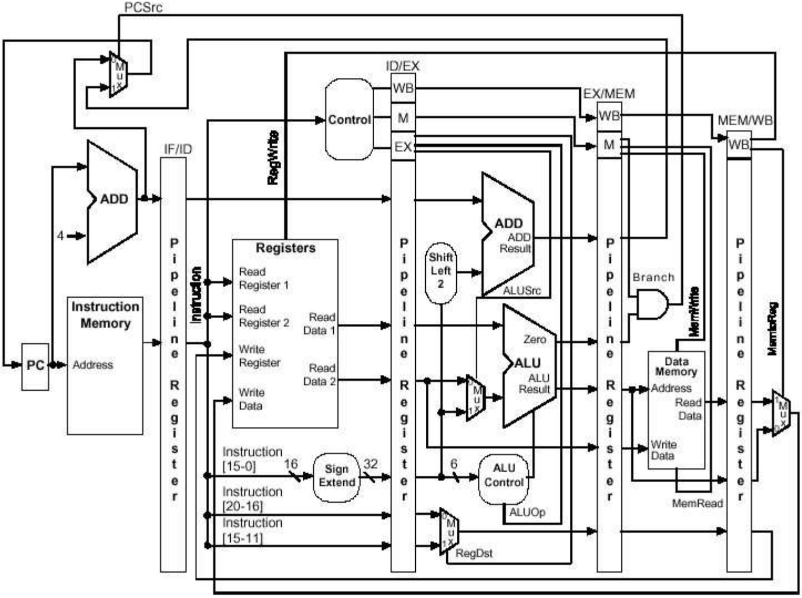
Google

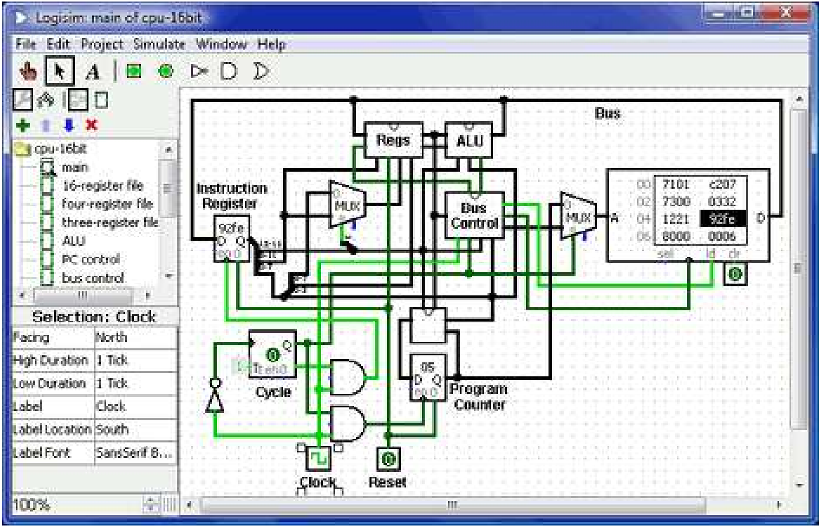


Software is part of ECU

ECU's are part of mechatronic systems for measurement and control




PCSrc
M
u
x
0
1
ID/EX
WB
M
EX
Control
EX/MEM
WB
M
MEM/WB
WB
IF/ID
RegWrite
ADD
4
Pipeline Register
Instruction
Registers
Read Register 1
Read Register 2
Write Register
Write Data
Read Data 1
Read Data 2
Instruction Memory
Address
PC
Pipeline Register
Shift Left 2
ADD
ADD Result
ALUSrc
Zero
ALU
ALU Result
Pipeline Register
Branch
MemWrite
Data Memory
Address
Read Data
Write Data
MemRead
Pipeline Register
MemtoReg
Instruction [15-0]
16
Sign Extend
32
6
ALU Control
ALUOp
Instruction [20-16]
Instruction [15-11]
RegDst



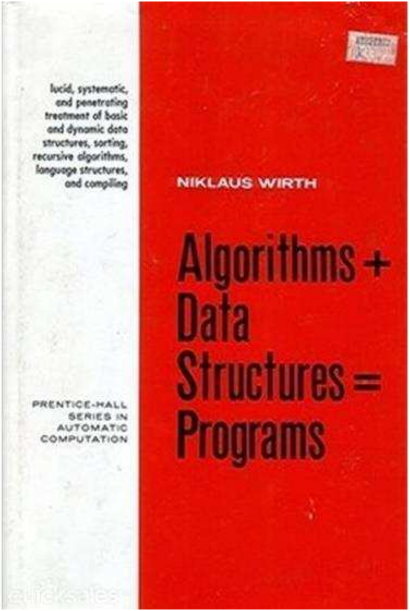
Logisim: main of cpu-16bit

Selection: Clock

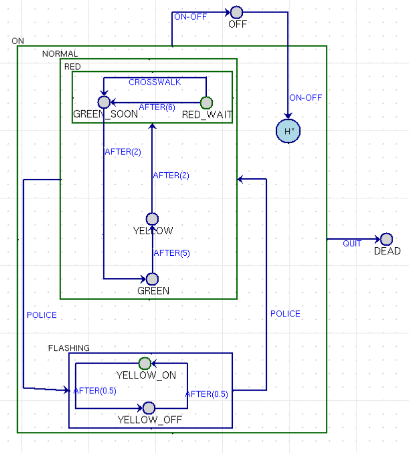
Facing	North
High Duration	1 Tick
Low Duration	1 Tick
Label	Clock
Label Location	South
Label Font	SansSerif 8...

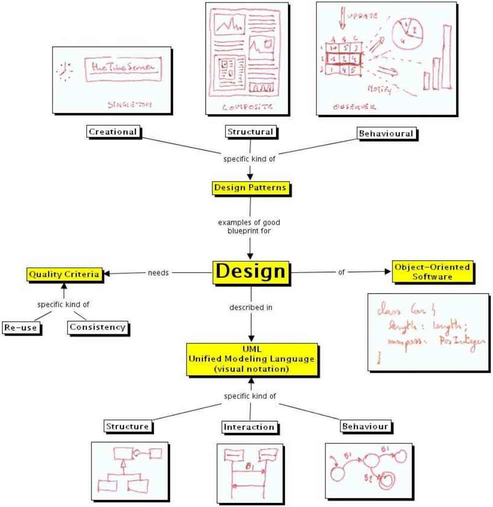
00	7101	c207
02	7300	0332
04	1221	92fe
06	8000	8006

```
         rce dataSource
   ion = dataSource
n =   getConnection();
=   connection.create
lectSQL = "SELECT * FROM
=   statement.execute
  Set.next()) {
```


lucid, systematic, and penetrating treatment of basic and dynamic data structures, sorting, recursive algorithms, language structures, and compiling
PRENTICE-HALL SERIES IN AUTOMATIC COMPUTATION
NIKLAUS WIRTH
Algorithms + Data Structures = Programs

Model, don’t code





Creational

Structural

Behavioural

specific kind of

Design Patterns

examples of good blueprint for

Design

needs

Quality Criteria

specific kind of

Re-use

Consistency

of

Object-Oriented Software

described in

UML
Unified Modeling Language
(visual notation)

specific kind of

Structure

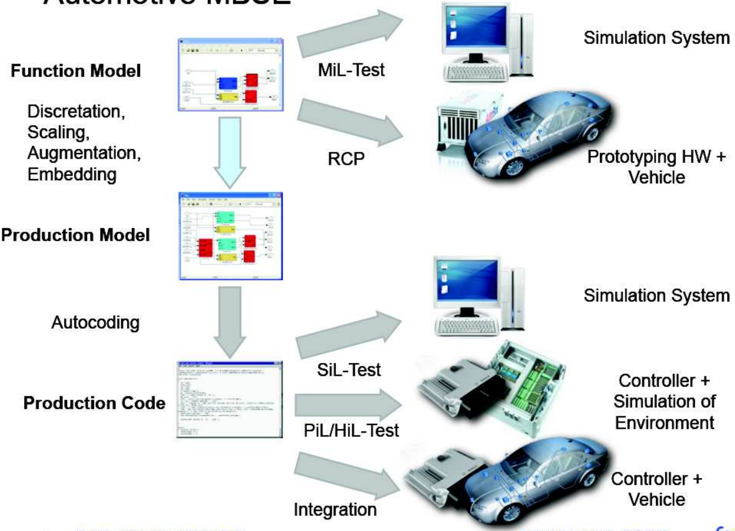
Interaction

Behaviour

BOSCH
F 01R 00D 125
Made in China
MT.9.7JL465Q
3600010A40

MOTORAUTHORITY
WWW.MOTORAUTHORITY.COM

Automotive MBSE



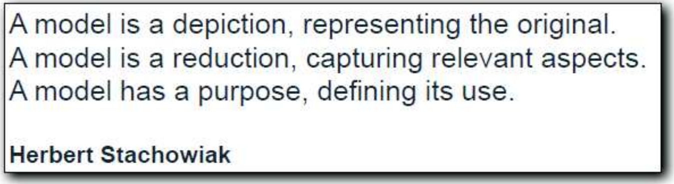
Dealing with Complexity

Dealing with Complexity

Model Everything . . . Explicitly

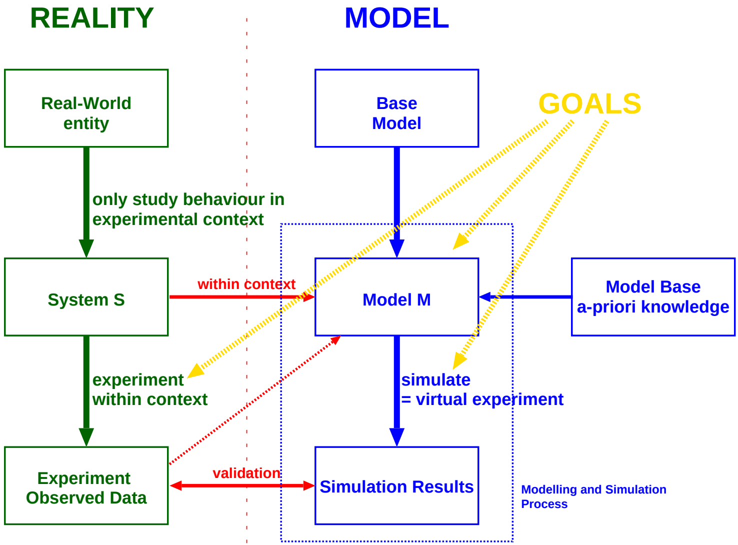
Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)



A model is a depiction, representing the original.
A model is a reduction, capturing relevant aspects.
A model has a purpose, defining its use.

Herbert Stachowiak


REALITY
MODEL
Real-World entity
only study behaviour in experimental context
System S
within context
Model M
Base Model
GOALS
Model Base a-priori knowledge
experiment within context
simulate = virtual experiment
Experiment Observed Data
validation
Simulation Results
Modelling and Simulation Process

Bernard P. Zeigler. *Multi-faceted Modelling and Discrete-Event Simulation.* Academic Press, 1984.

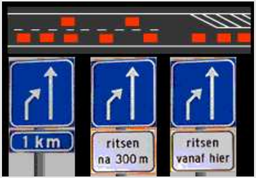
Simulation . . . when too costly/dangerous

analysis ↔ design

Simulation . . . real experiment not ethical

“physical” simulation, training

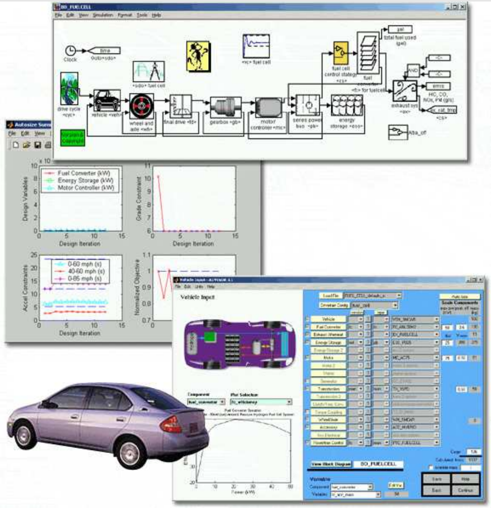
Simulation . . . evaluate alternatives



RITSEN.
'T MOET VAN TWEE KANTEN KOMEN.
UITWEG

Modelling and Simulation for . . .

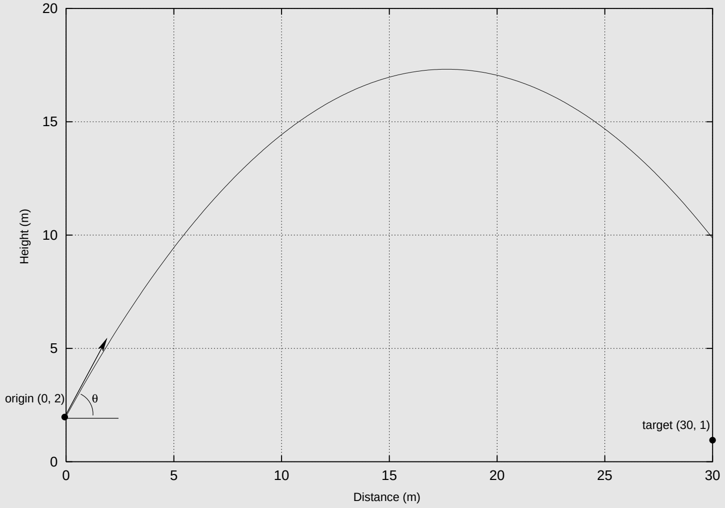
Simulation . . . “Do it Right the First Time”



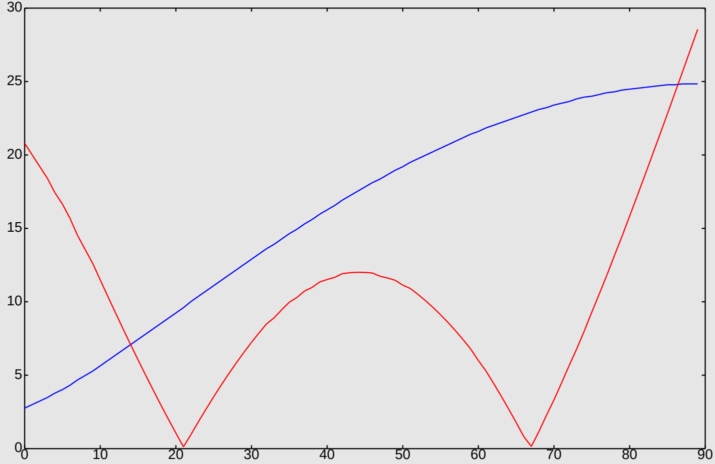
essence: “shooting” problems

© Colorpix.be

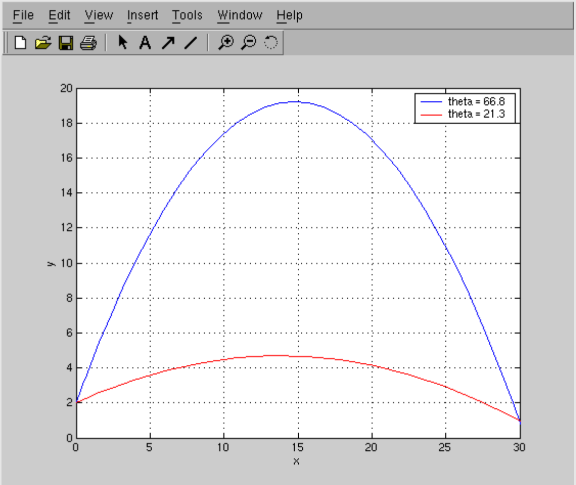
defining a “hit”



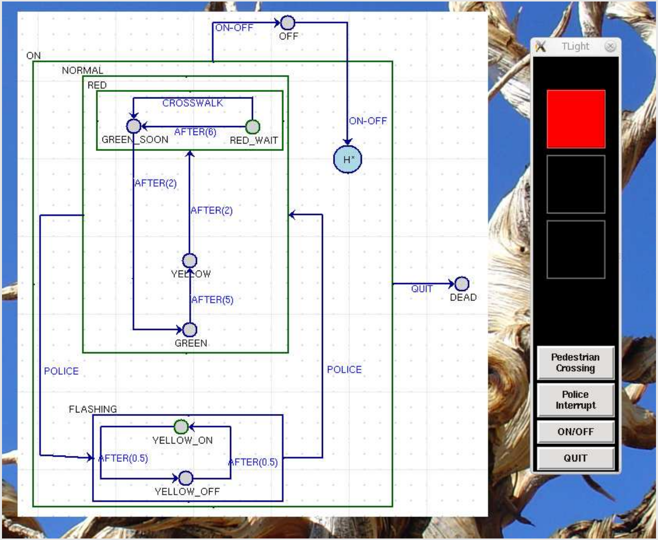
optimizing a “performance metric”



optimal solution. . . s



Modelling/Simulation . . . and code/app Synthesis



The spectrum of uses of models

- Documentation

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)
- Test Generation

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)
- Test Generation
- Simulation (one model, one behaviour)
  . . . for calibration, optimization, . . .

Modelling and Simulation for . . .

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)
- Test Generation
- Simulation (one model, one behaviour)
  . . . for calibration, optimization, . . .
- Application Synthesis (mostly for models of software)

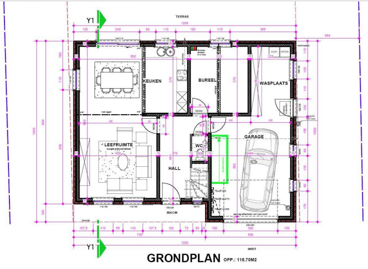
Requirements (“What?”)

- Detached or Semi-detached
- Style (classical, modern, . . .)
- Number of Floors
- Number of rooms of different types (bedrooms, bathrooms, . . .)
- Garage, Storage, . . .
- Cellar
- . . .

Requirements (“What?”)

- Detached or Semi-detached
- Style (classical, modern, . . .)
- Number of Floors
- Number of rooms of different types (bedrooms, bathrooms, . . .)
- Garage, Storage, . . .
- Cellar
- . . .

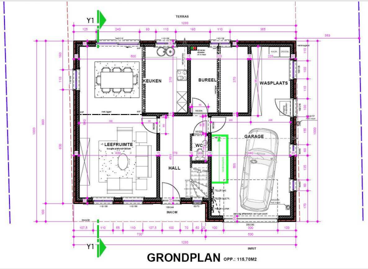
Design (“How?”)

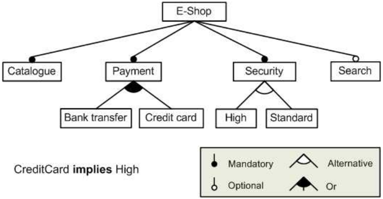


Requirements (“What?”)

- Detached or Semi-detached
- Style (classical, modern, . . .)
- Number of Floors
- Number of rooms of different types (bedrooms, bathrooms, . . .)
- Garage, Storage, . . .
- Cellar
- . . .

Design (“How?”)





E-Shop

Catalogue

Payment

Security

Search

Bank transfer

Credit card

High

Standard

Mandatory

Alternative

Optional

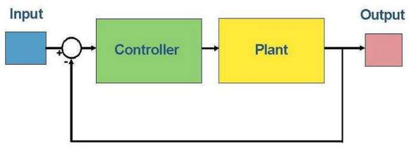
Or

CreditCard **implies** High

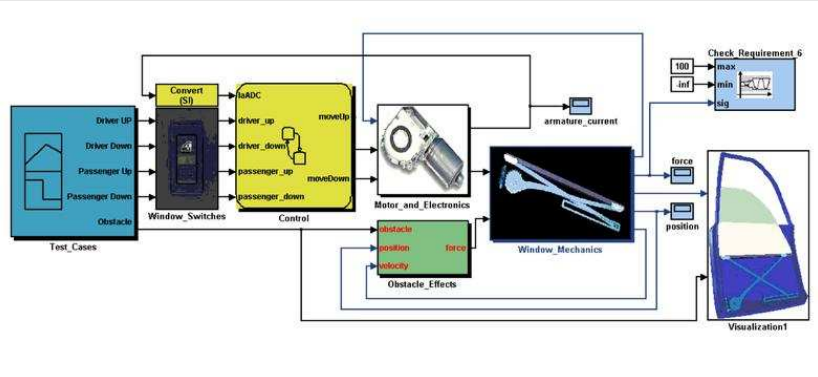
System Boundaries

- **System** to be built/studied
- **Environment** with which the system interacts

System vs. “Plant”

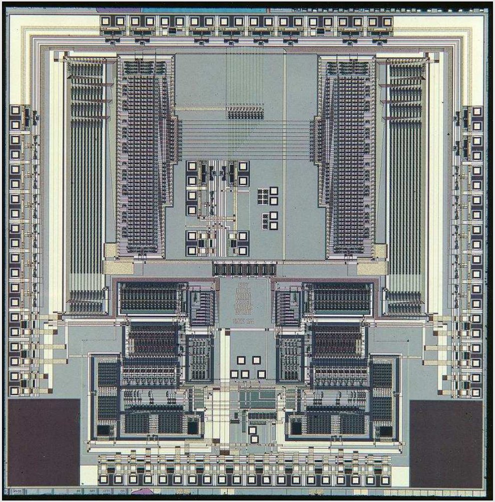


System vs. “Plant”



www.mathworks.com/products/demos/simulink/PowerWindow/html/PowerWindow1.html

Number of Components

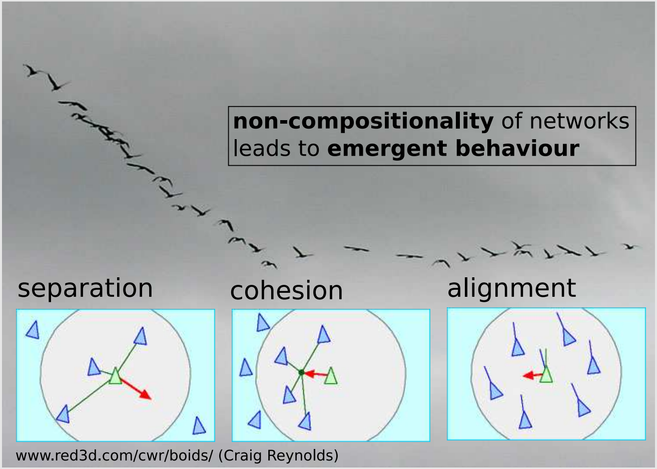


Crowds: diversity, interaction

www.3dm3.com

Diversity of Components: Power Window

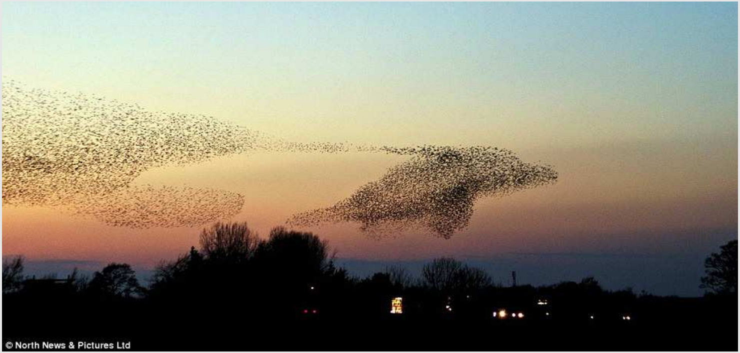
Non-compositional/Emergent Behaviour



non-compositionality of networks leads to **emergent behaviour**

www.red3d.com/cwr/boids/ (Craig Reynolds)

Emergent Behaviour



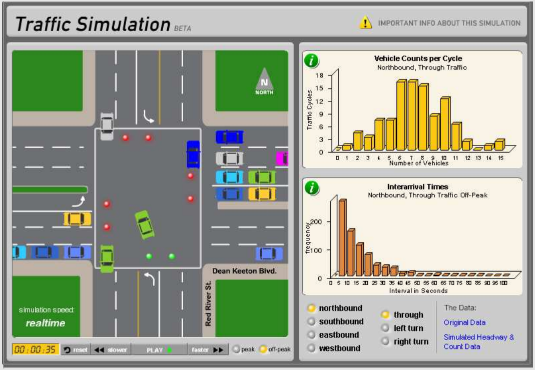
© North News & Pictures Ltd

Engineered Emergent Behaviour

Robert Bogue. *Swarm intelligence and robotics.* Industrial Robot: An International Journal. 35(6):488 - 495, 2008.

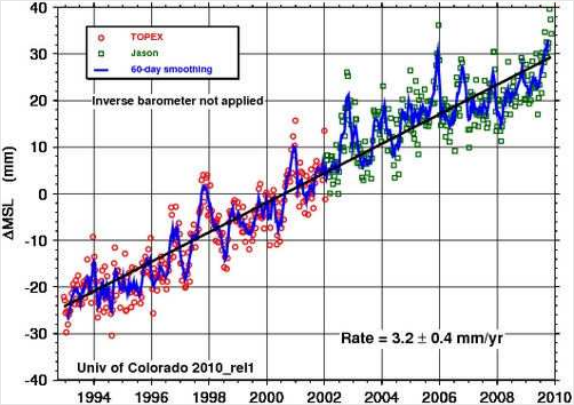
Uncertainty

Often related to level of abstraction:
for example continuous vs. discrete

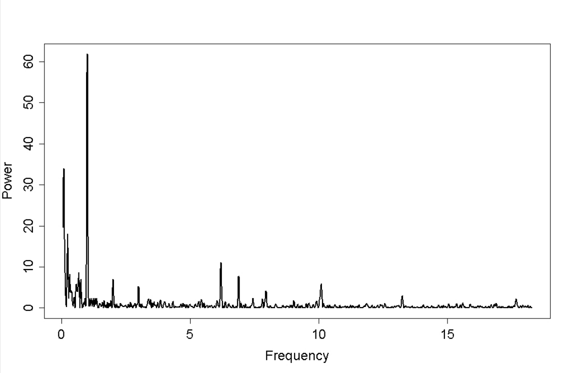


www.engr.utexas.edu/trafficSims/

Question: is the deviation from the trend periodic?



Answer: transform to make the solution obvious

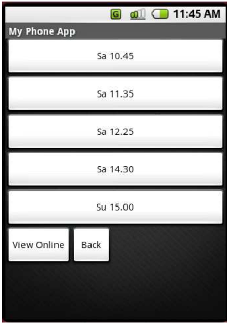


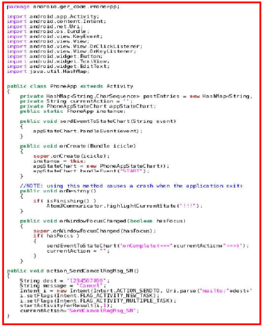
Guiding principle (~ physics: principle of minimal action)

minimize **accidental complexity**,
only **essential complexity** remains

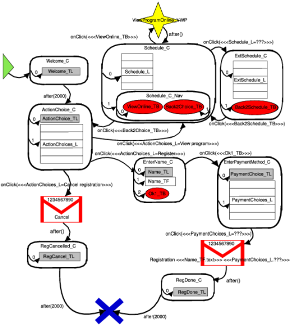
Fred P. Brooks. No Silver Bullet – Essence and Accident in Software Engineering.
Proceedings of the IFIP Tenth World Computing Conference, pp. 1069–1076, 1986.

`http://www.lips.utexas.edu/ee382c-15005/Readings/Readings1/05-Broo87.pdf`





```
package android.ger_code.PhoneApp;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnKeyListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;
import java.util.HashMap;


public class PhoneApp extends Activity
{
    private HashMap<String,CharSequence> pastEntries = new HashMap<String,
    private String currentAction = "";
    private PhoneAppStateChart appStateChart;
    public static PhoneApp instance;

    public void sendEventToStateChart(String event)
    {
        appStateChart.handleEvent(event);
    }

    public void onCreate(Bundle icicle)
    {
        super.onCreate(icicle);
        instance = this;
        appStateChart = new PhoneAppStateChart();
        appStateChart.handleEvent("START");
    }

    //NOTE: using this method causes a crash when the application exits
    public void onDestroy()
    {
        if( isFinishing() )
            Atom3Communicator.highlightCurrentState("!!!");
    }

    public void onWindowFocusChanged(boolean hasFocus)
    {
        super.onWindowFocusChanged(hasFocus);
        if( hasFocus )
        {
            sendEventToStateChart("onComplete(<<<"+currentAction+">>>)");
            currentAction = "";
        }
    }

public void action_SendCancelRegMsg_SM()
{
    String dest = "1234567890";
    String message = "Cancel";
    Intent i = new Intent(Intent.ACTION_SENDTO, Uri.parse("mailto:"+dest+"
    i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    i.setFlags(Intent.FLAG_ACTIVITY_MULTIPLE_TASK);
    startActivityForResult(i,1);
    currentAction="SendCancelRegMsg_SM";
}
```



Dealing with Complexity: some approaches

Dealing with Complexity: some approaches

- multiple abstraction levels

Dealing with Complexity: some approaches

- multiple abstraction levels
- optimal formalism

Dealing with Complexity: some approaches

- multiple abstraction levels
- optimal formalism
- multiple formalisms

Dealing with Complexity: some approaches

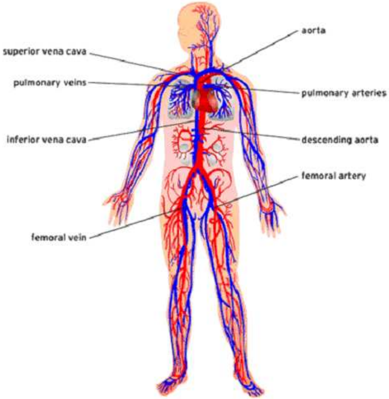
- multiple abstraction levels
- optimal formalism
- multiple formalisms
- multiple views

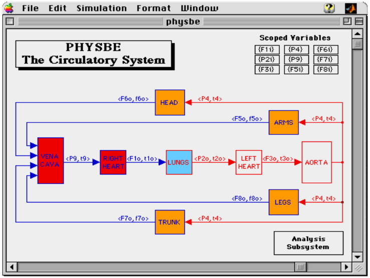
Dealing with Complexity: some approaches

- multiple abstraction levels
- optimal formalism
- multiple formalisms
- multiple views

Modularity!

Different Abstraction Levels – properties preserved





Levels of Abstraction/Views: Morphism

$$\begin{tikzcd}
\text{model} & \text{M\_t} \arrow[r, "\text{abstraction}"'] \arrow[d, "\text{simulation}"'] & \text{M\_d} \arrow[d] \\
\text{trajectory} & \text{traj\_t} \arrow[r] & \text{traj\_d} \\
 & \text{detailed (technical) level} & \text{abstract (decision) level}
\end{tikzcd}$$

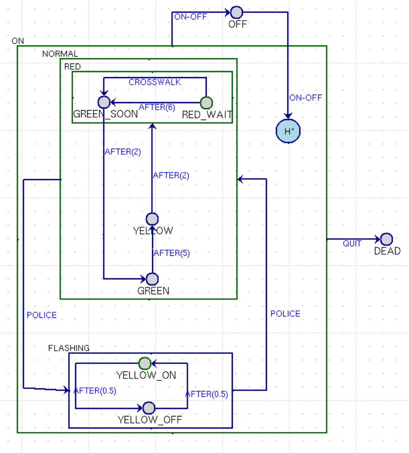
Abstraction Relationship

foundation: the *information* contained in a model M.
Different *questions* (properties) $P = I(M)$ which can be asked concerning the model.
These questions either result in true or false.

Abstraction and its opposite, *refinement* are
relative to a non-empty set of questions (properties) P.

- If M_1 is an *abstraction* of M_2 with respect to P, for all $p \in P$: $M_1 \models p \Rightarrow M_2 \models p$. This is written $M_1 \sqsupseteq_P M_2$.
- M_1 is said to be a *refinement* of M_2 iff M_2 is an *abstraction* of M_1. This is written $M_1 \sqsubseteq_P M_2$.

Most Appropriate Formalism

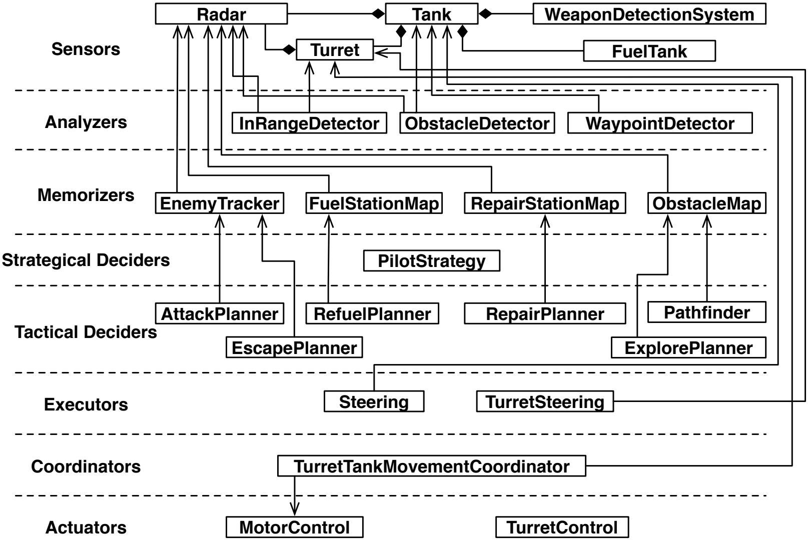


Most Appropriate Formalism (Minimizing Accidental Complexity)

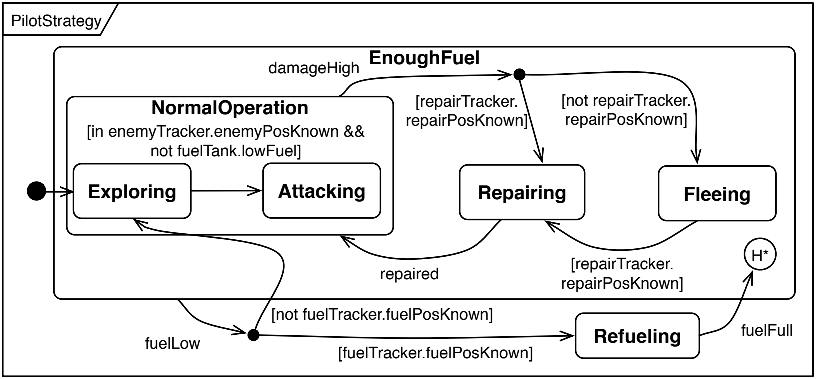
`www.planeshift.it`

Massively Multiplayer Online Role Playing games need Non-Player Characters (NPCs)

TankWars: high level

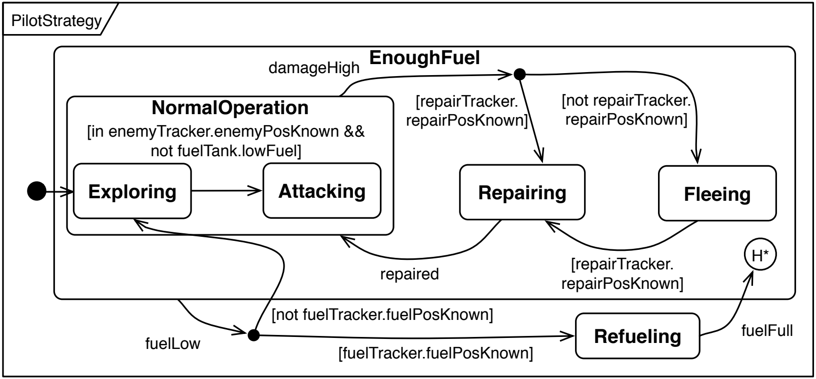


Strategic Deciders – High-level Goals



Jörg Kienzle, Alexandre Denault, Hans Vangheluwe. Model-Based Design of Computer-Controlled Game Character Behavior. MoDELS 2007: 650-665

Strategic Deciders – High-level Goals

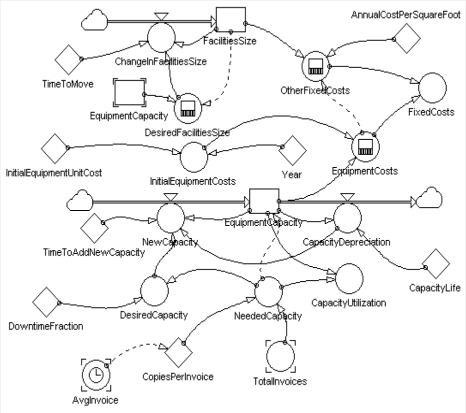


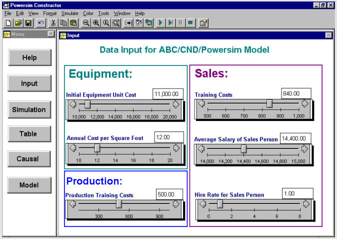
Jörg Kienzle, Alexandre Denault, Hans Vangheluwe. Model-Based Design of Computer-Controlled Game Character Behavior. MoDELS 2007: 650-665

Could have used production rules instead of Statecharts

Eugene Syriani, Hans Vangheluwe: Programmed Graph Rewriting with DEVS. AGTIVE 2007: 136-151

“Management Flight Simulator” using Forrester System Dynamics model





Data Input for ABC/CND/Powersim Model

Equipment:

Initial Equipment Unit Cost 11,000.00

Annual Cost per Square Foot 12.00

Production:

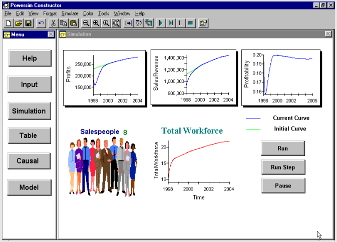
Production Training Costs 500.00

Sales:

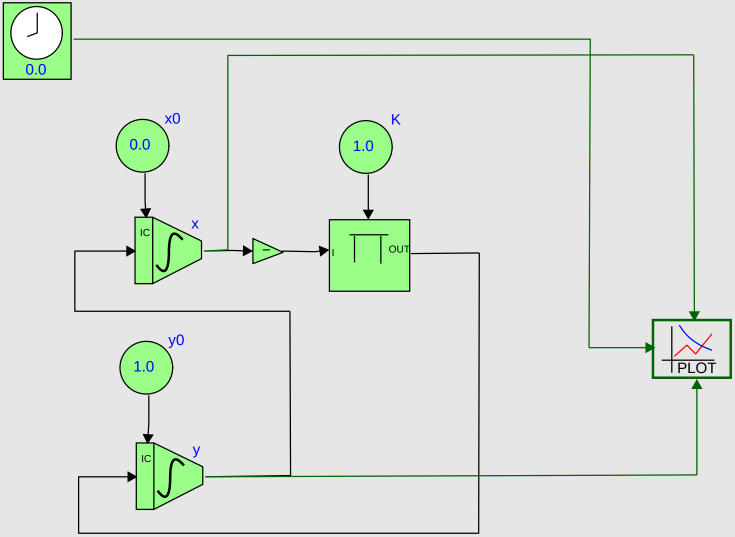
Training Costs 840.00

Average Salary of Sales Person 14,400.00

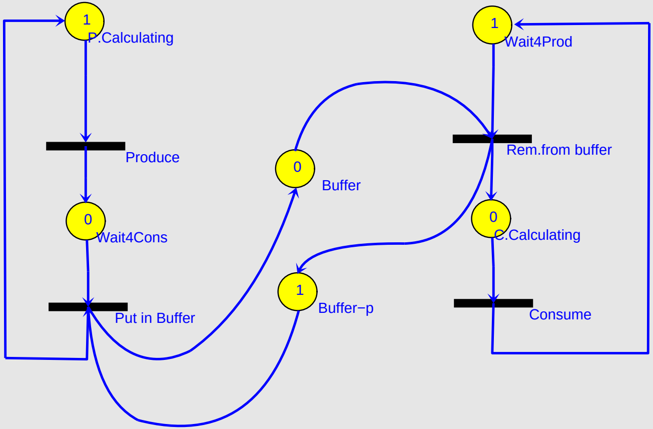
Hire Rate for Sales Person 1.00



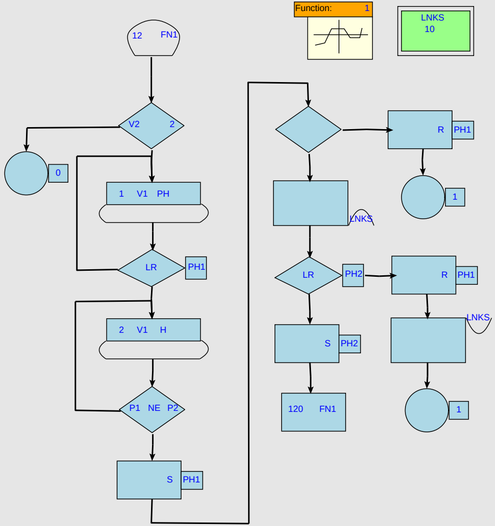
Causal Block Diagram model of Harmonic Oscillator



Petri Net model of Producer – Consumer

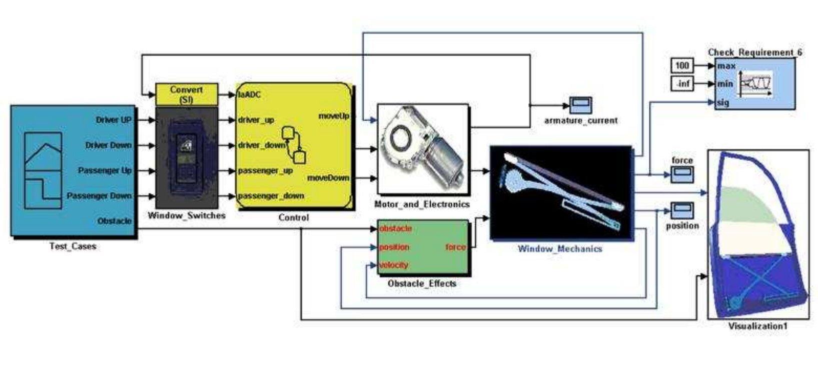


GPSS model of Telephone Exchange



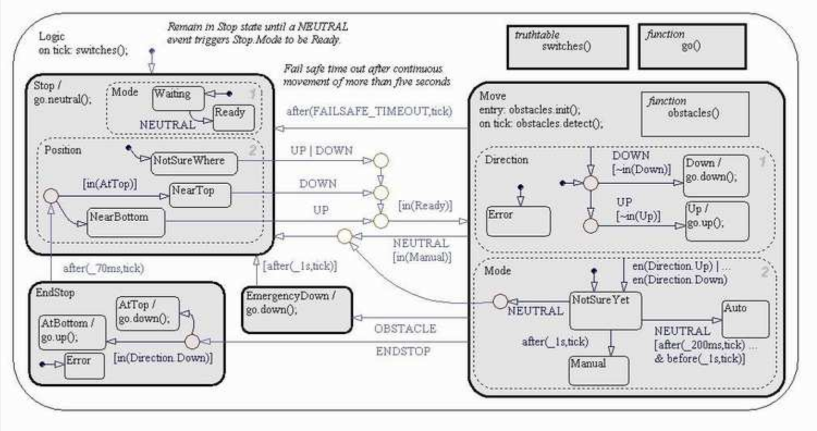
Multiple Formalisms: Power Window

Components in Different Formalisms



www.mathworks.com/products/demos/simulink/PowerWindow/html/PowerWindow1.html

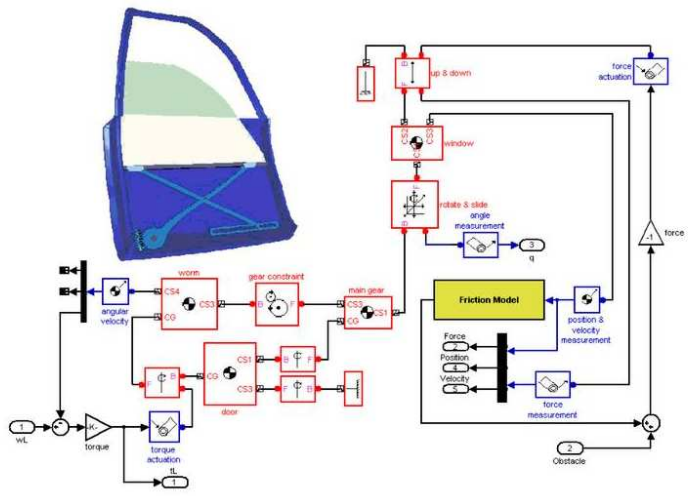
Controller, using Statechart(StateFlow) formalism



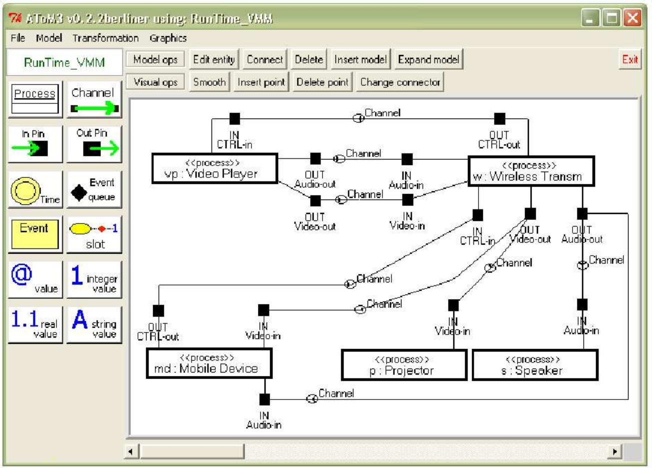
Remain in Stop state until a NEUTRAL event triggers Stop.Mode to be Ready.

Fail safe time out after continuous movement of more than five seconds

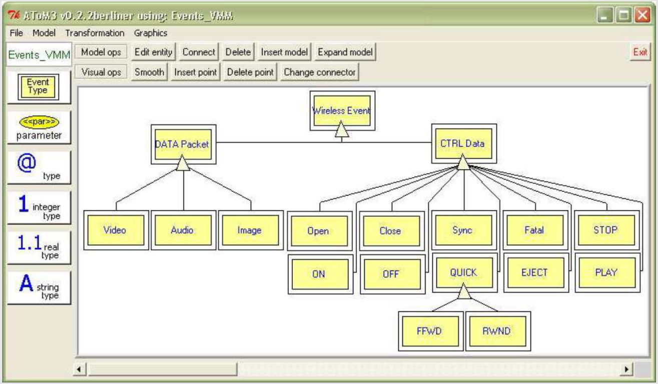
Mechanics subsystem



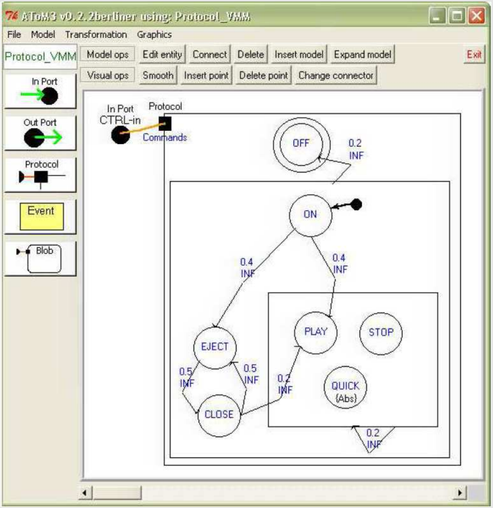
Multiple (consistent !) Views (in ≠ Formalisms)



View: Events Diagram



View: Protocol Statechart



No Free Lunch!

Solutions often introduce
their **own accidental complexity**

No Free Lunch!

Solutions often introduce
their **own accidental complexity**

- multiple abstraction levels (need **morphism**)

No Free Lunch!

Solutions often introduce
their **own accidental complexity**

- multiple abstraction levels (need **morphism**)
- optimal formalism (need **precise meaning**)

No Free Lunch!

Solutions often introduce
their **own accidental complexity**

- multiple abstraction levels (need **morphism**)
- optimal formalism (need **precise meaning**)
- multiple formalisms (need **relationship**)

No Free Lunch!

Solutions often introduce
their **own accidental complexity**

- multiple abstraction levels (need **morphism**)
- optimal formalism (need **precise meaning**)
- multiple formalisms (need **relationship**)
- multiple views (need **consistency**)

No Free Lunch!

Solutions often introduce
their **own accidental complexity**

- multiple abstraction levels (need **morphism**)
- optimal formalism (need **precise meaning**)
- multiple formalisms (need **relationship**)
- multiple views (need **consistency**)

Multi-Paradigm Modelling
(*model* everything, minimize *accidental complexity*)

- at the most appropriate **level of abstraction**
- using the most appropriate **formalism(s)**
  Class Diagrams, Differential Algebraic Equations, Petri Nets, Bond Graphs, Statecharts, CSP, Queueing Networks, Sequence Diagrams, Lustre/Esterel, . . .
- with **transformations** as first-class models

Pieter J. Mosterman and Hans Vangheluwe.

Computer Automated Multi-Paradigm Modeling: An Introduction. Simulation 80(9):433–450, September 2004.

Special Issue: Grand Challenges for Modeling and Simulation.