Software Intensive Systems: Dealing with Complexity

Hans Vangheluwe

Dealing with Complexity

Dealing with Complexity

Model Everything . . . Explicitly

Dealing with Complexity

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

Dealing with Complexity

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

The spectrum of uses of models

- Documentation

Dealing with Complexity

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

The spectrum of uses of models

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

Dealing with Complexity

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

The spectrum of uses of models

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

Dealing with Complexity

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

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

Dealing with Complexity

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

The spectrum of uses of models

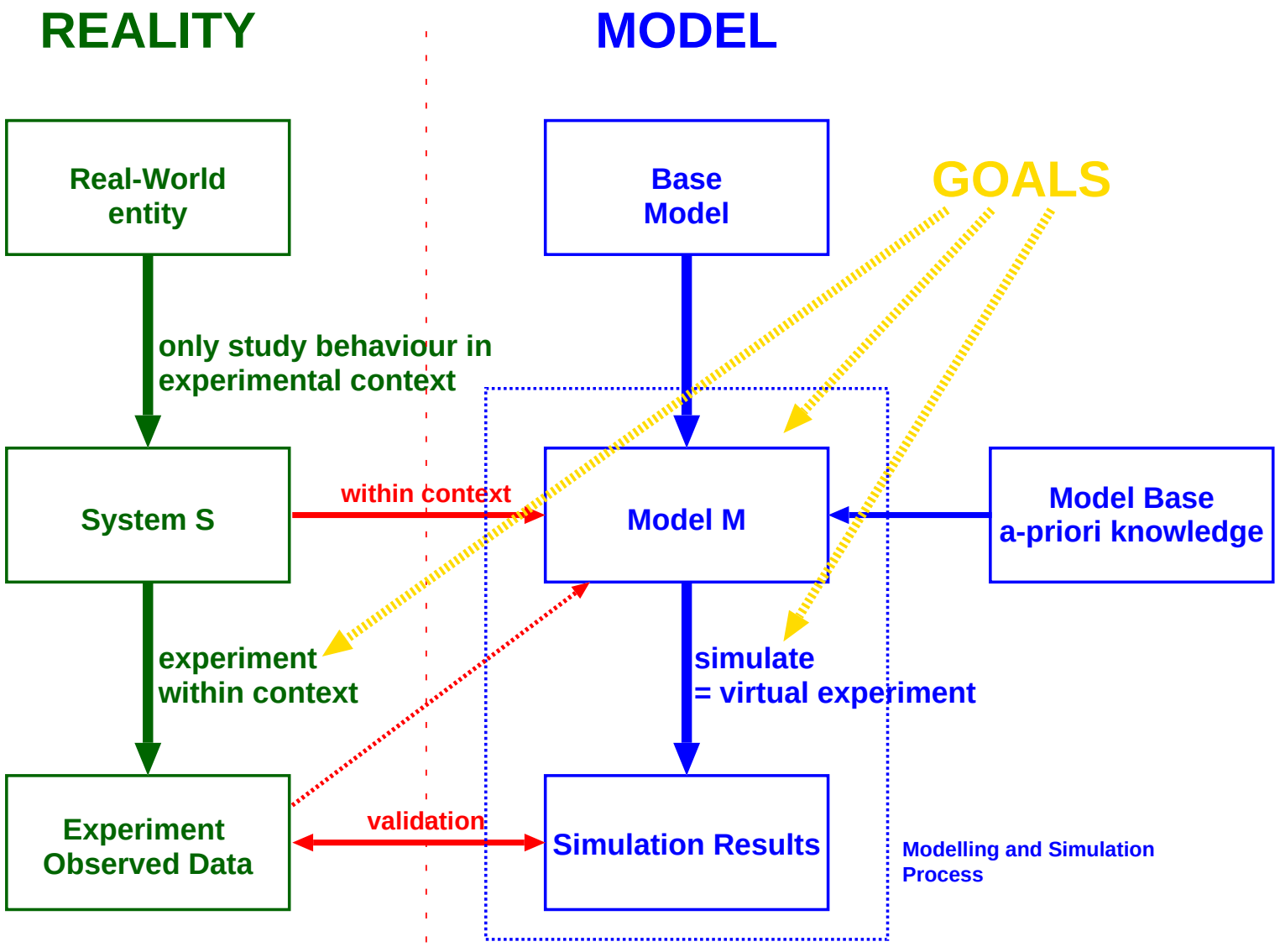
- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)
- Test Generation
- Simulation (one model, one behaviour)
  . . . for calibration, optimization, . . .

Dealing with Complexity

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

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)



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

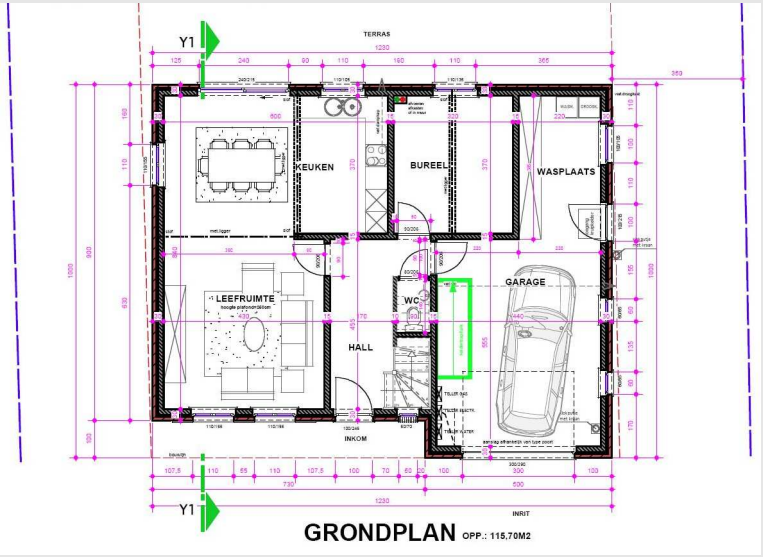
Requirements (“What?”)

- 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
- . . .

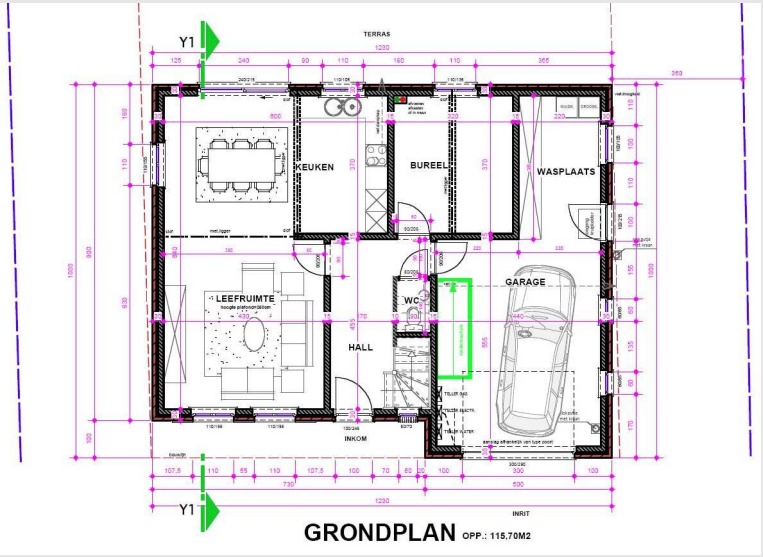
Design (“How?”)

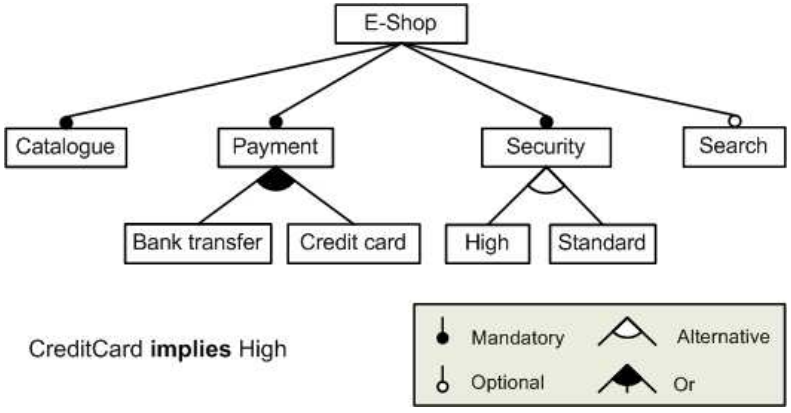


Requirements (“What?”)

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

Design (“How?”)

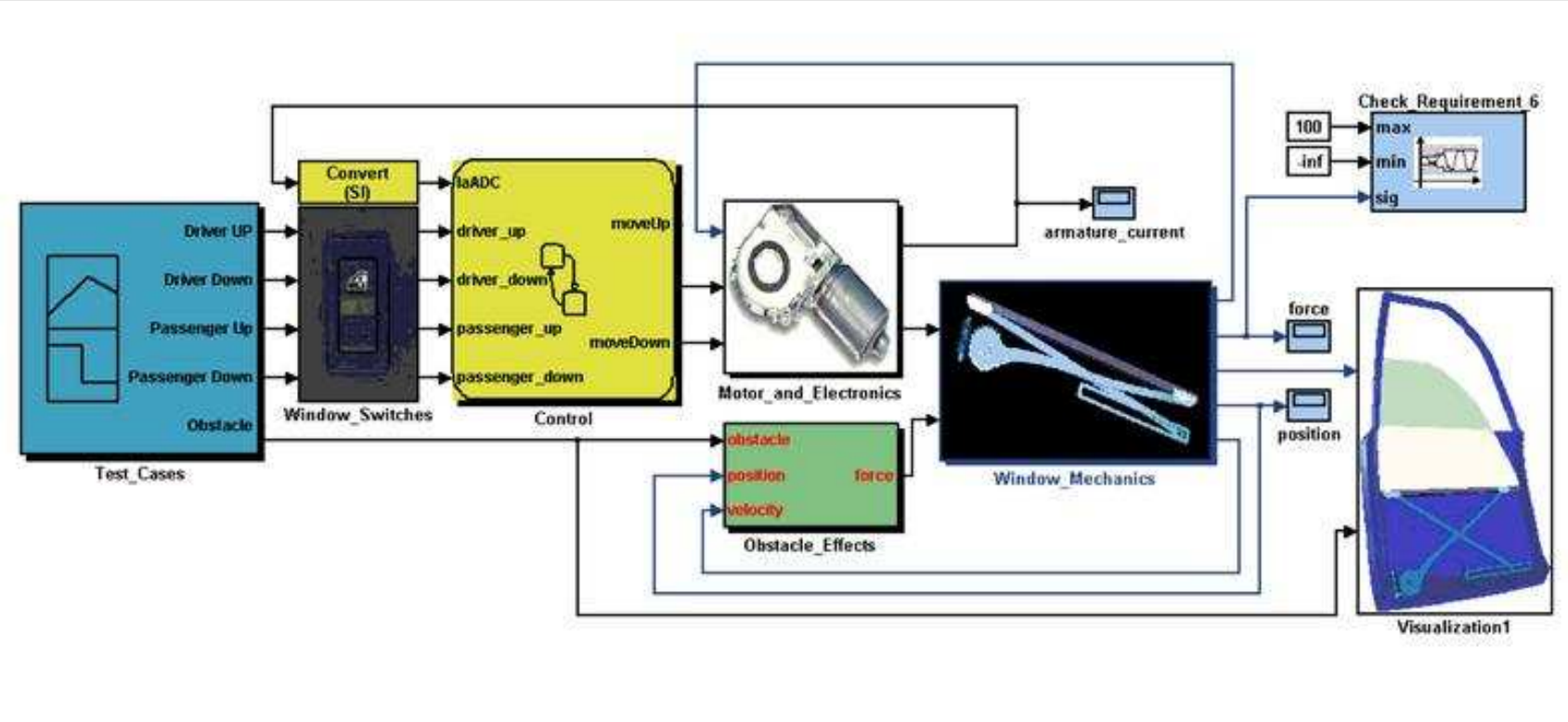




System Boundaries

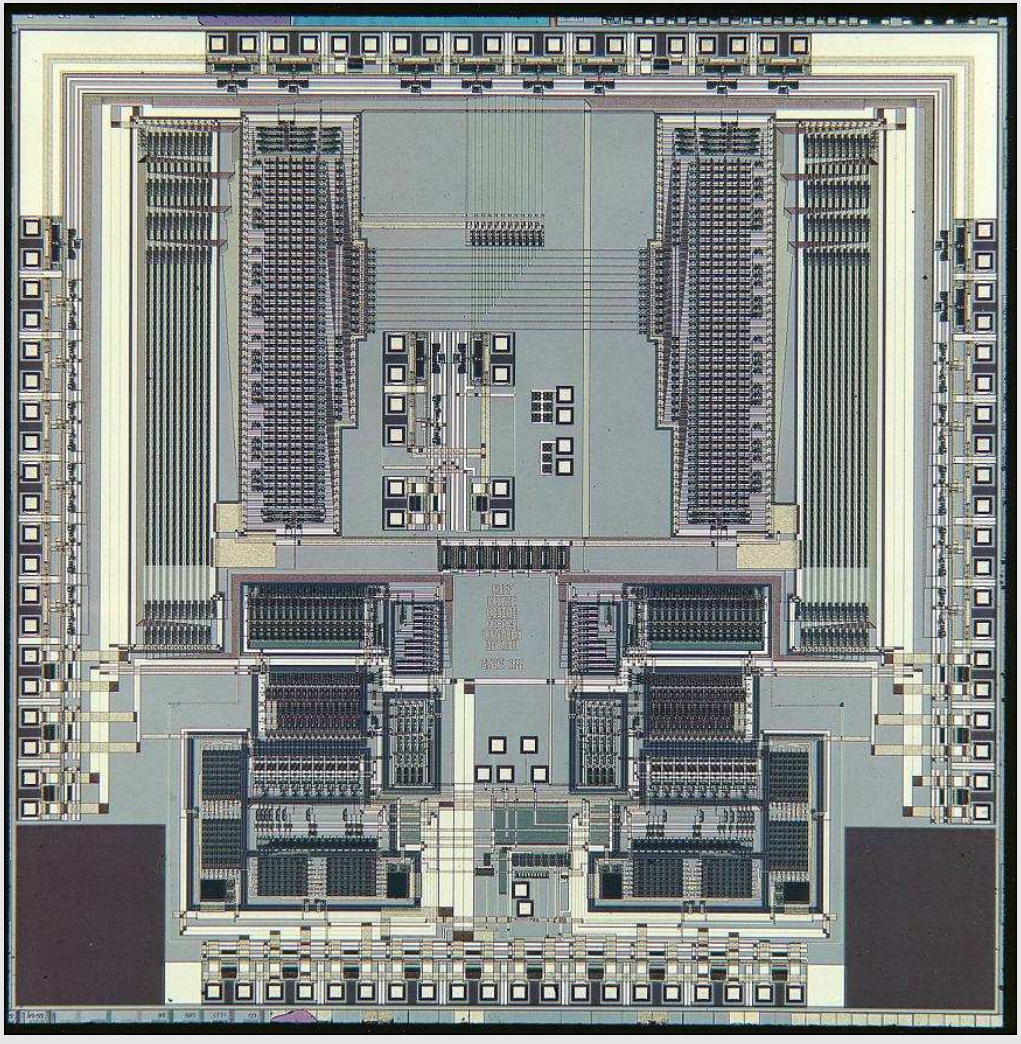
- **System** to be built/studied
- **Environment** with which the system interacts

System vs. “Plant”



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

Number of Components – hierarchical (de-)composition

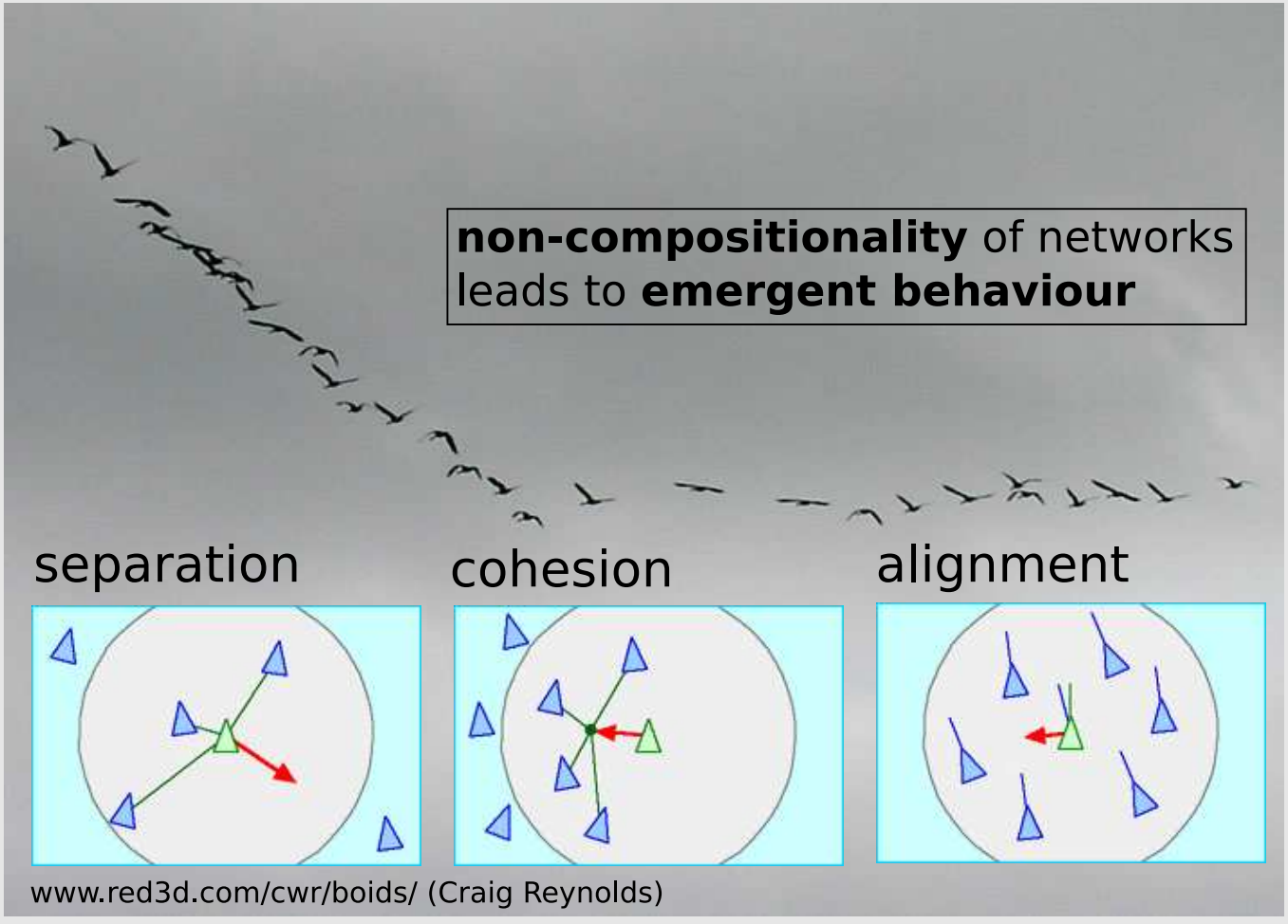


Crowds: diversity, interaction

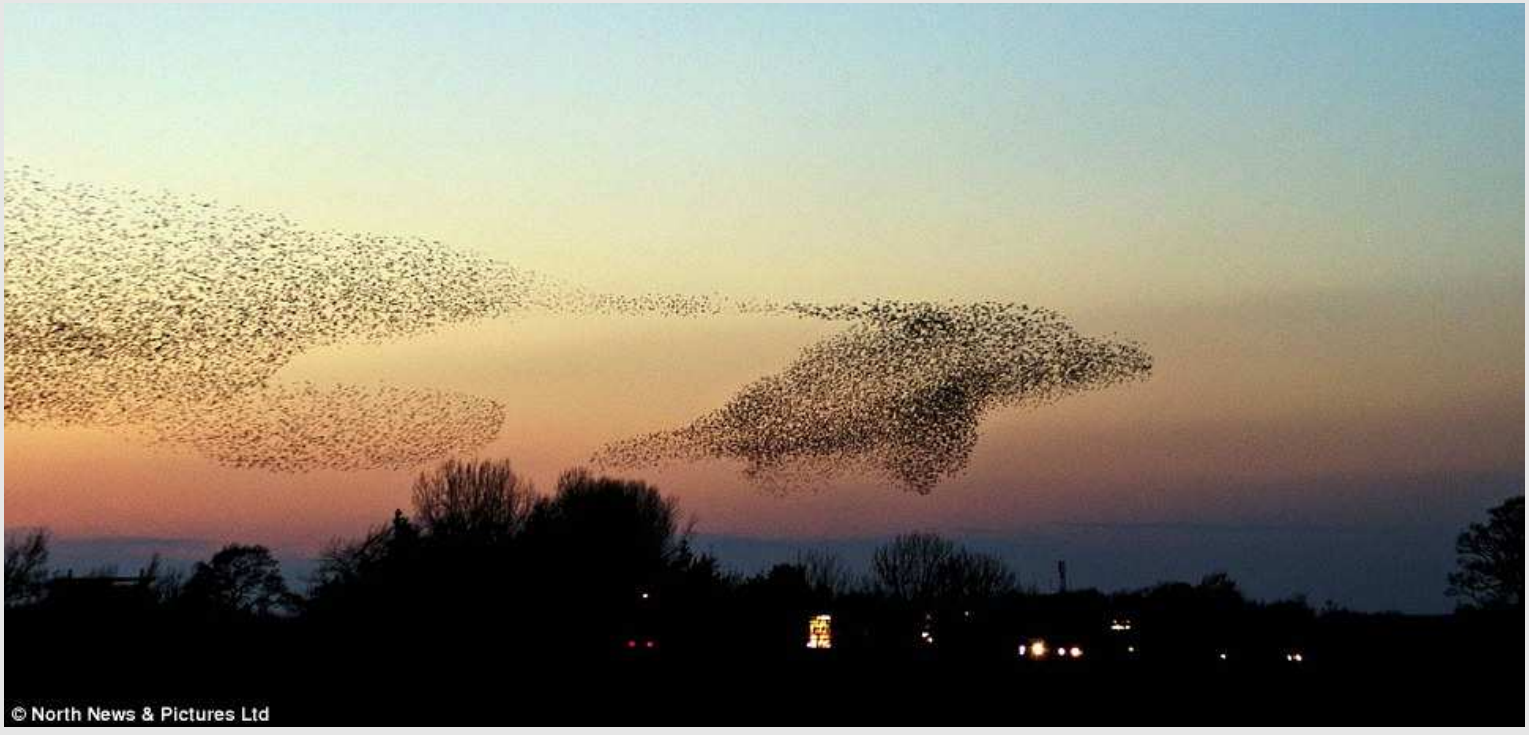
www.3dm3.com

Diversity of Components: Power Window

Non-compositional/Emergent Behaviour



Emergent Behaviour

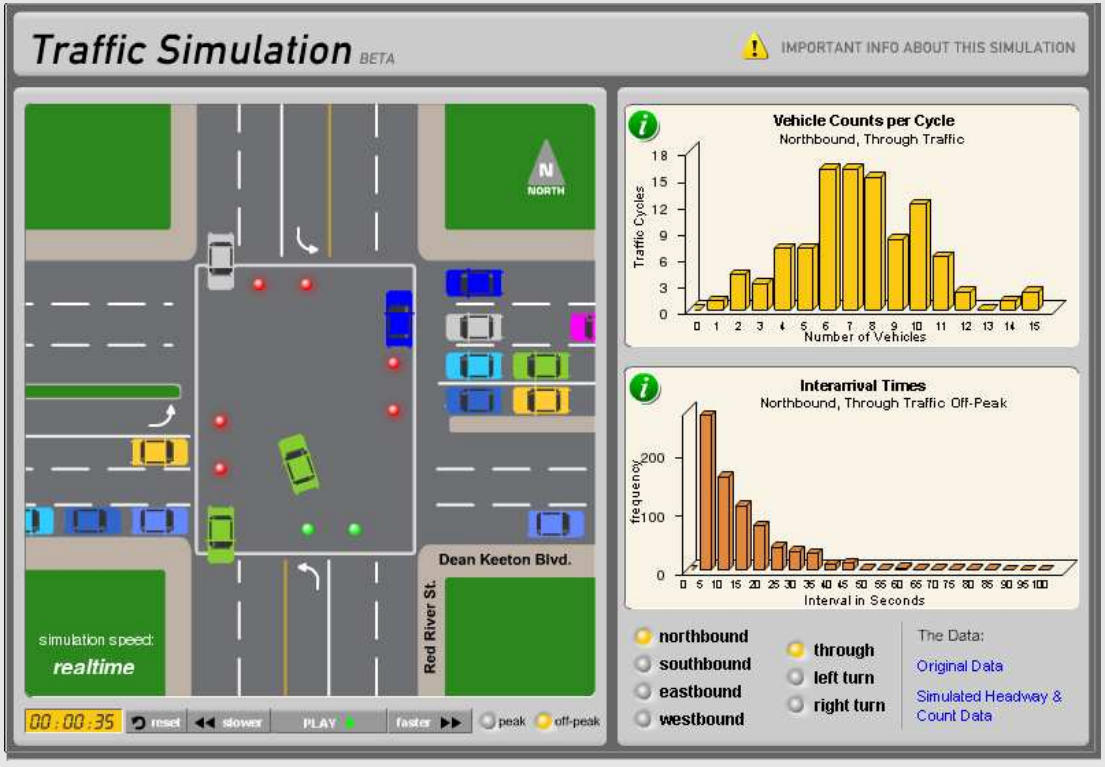


Engineered Emergent Behaviour

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

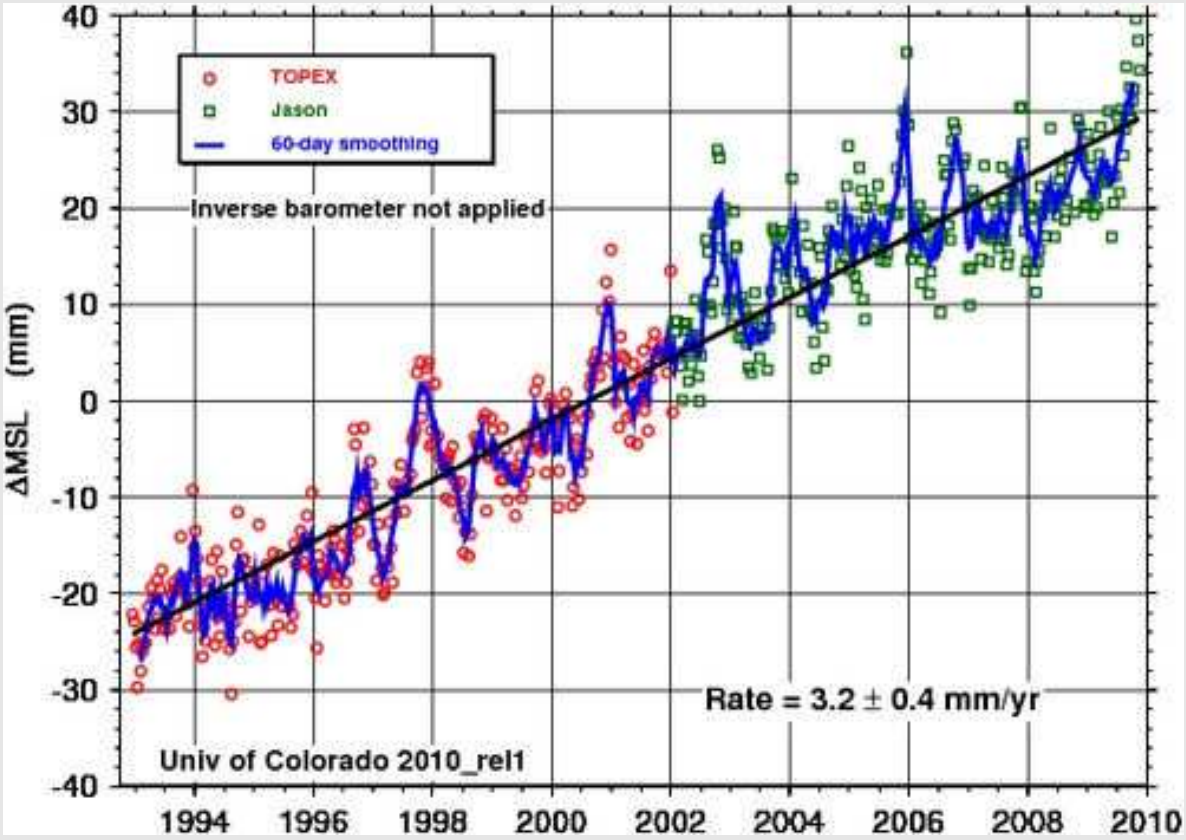
Uncertainty

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

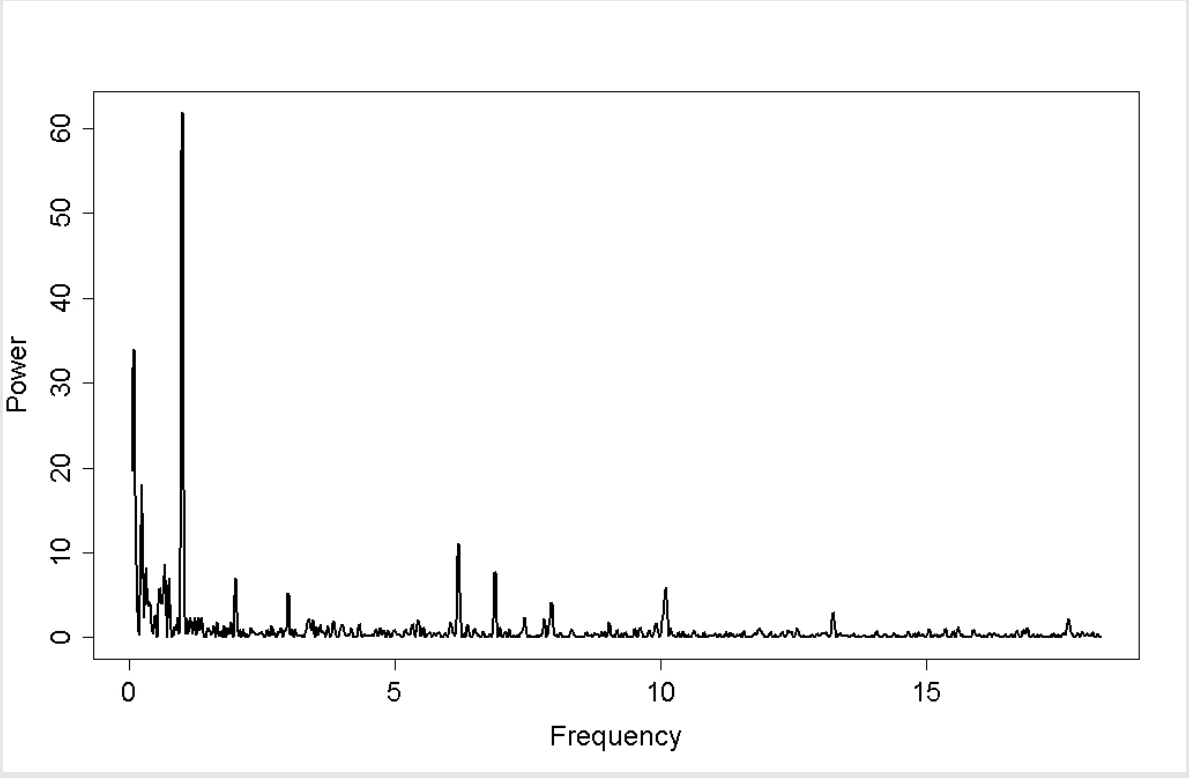


www.engr.utexas.edu/trafficSims/

Question: is the deviation from the trend periodic?



Answer: transform to make the solution obvious

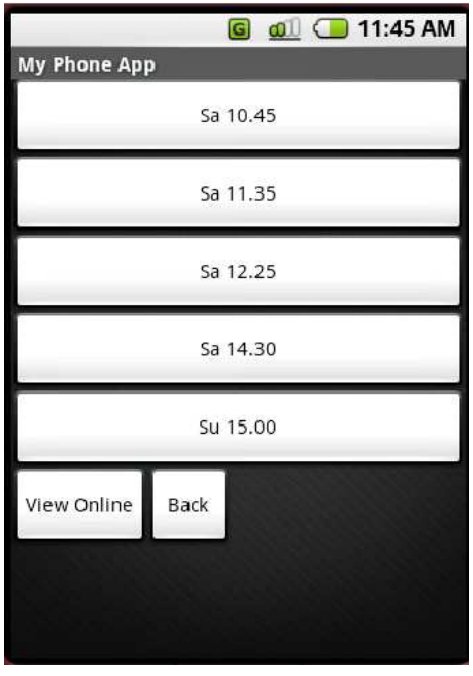


Guiding principle (∼ physics: principle of minimal action)

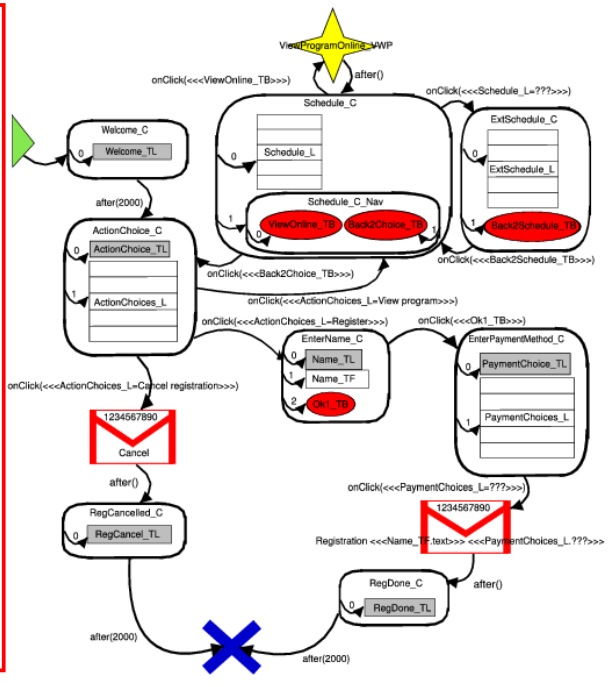
minimize **accidental complexity**,
only **essential complexity** remains

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

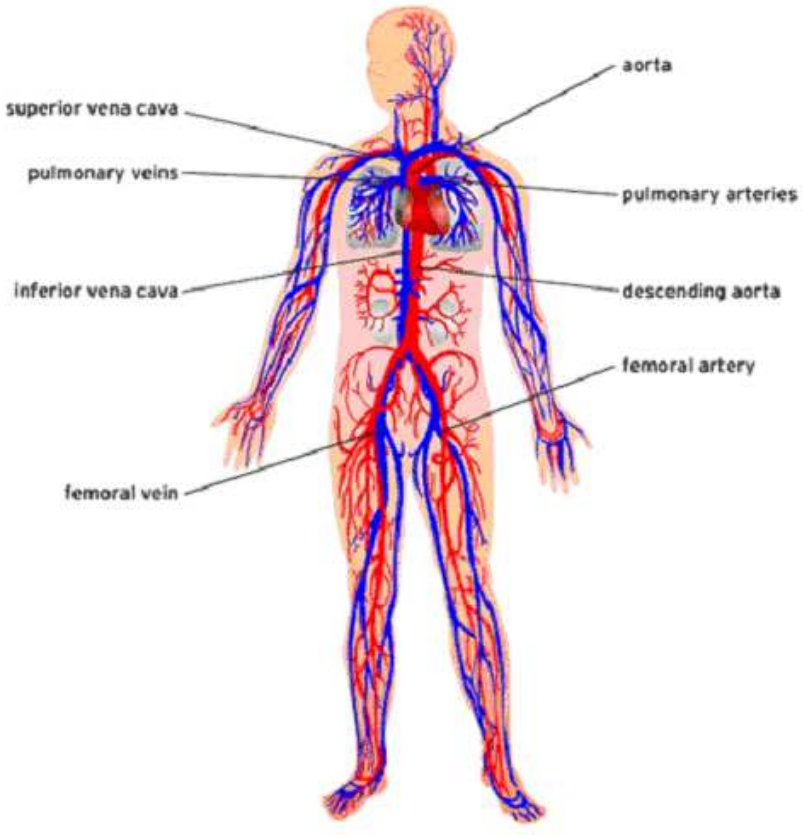
- multiple abstraction levels
- optimal formalism
- multiple formalisms
- multiple views

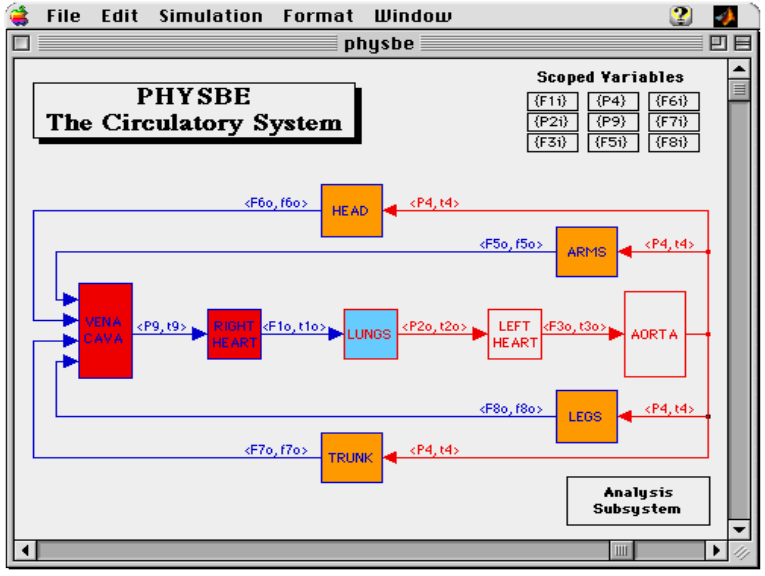
Dealing with Complexity: some approaches

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

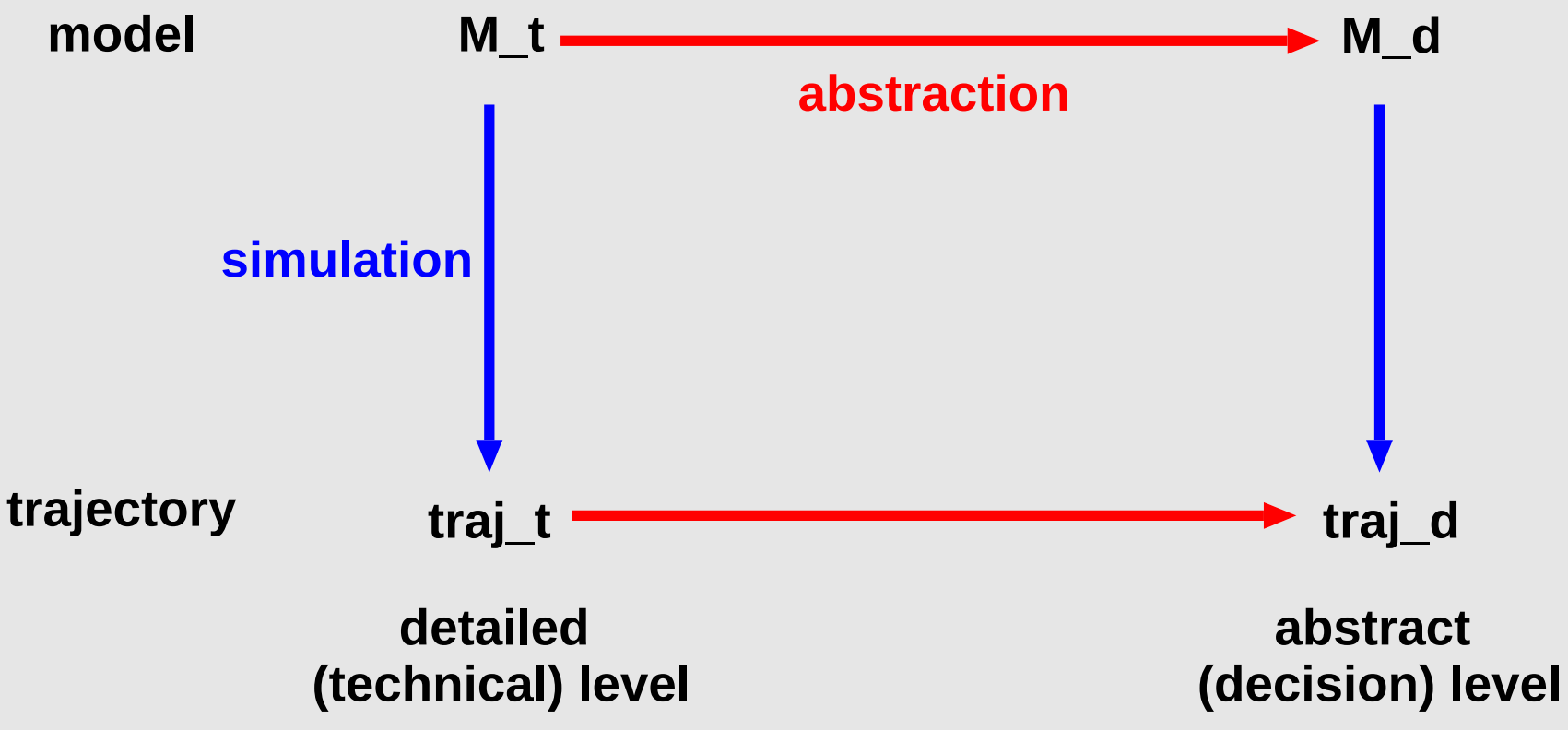
Modularity!

Different Abstraction Levels – properties preserved





Levels of Abstraction/Views: Morphism



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.

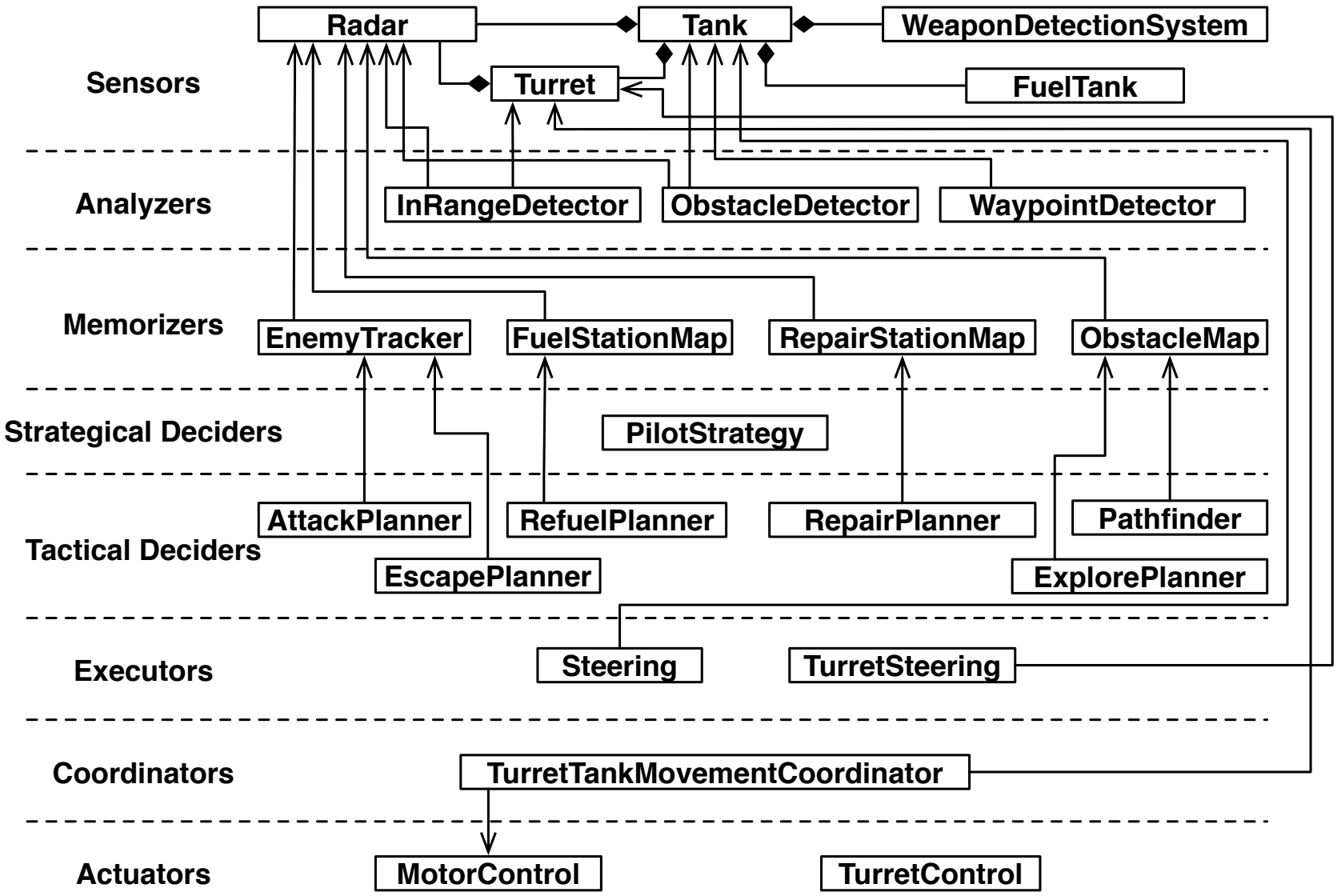
- If M_1 is an *abstraction* of M_2 with respect to P, for all $p \in P$: $M_1 \models p \Rightarrow M_2 \models p$. This is written $M_1 \sqsupseteq_P M_2$.
- M_1 is said to be a *refinement* of M_2 iff M_1 is an *abstraction* of M_2. This is written $M_1 \sqsubseteq_P M_2$.

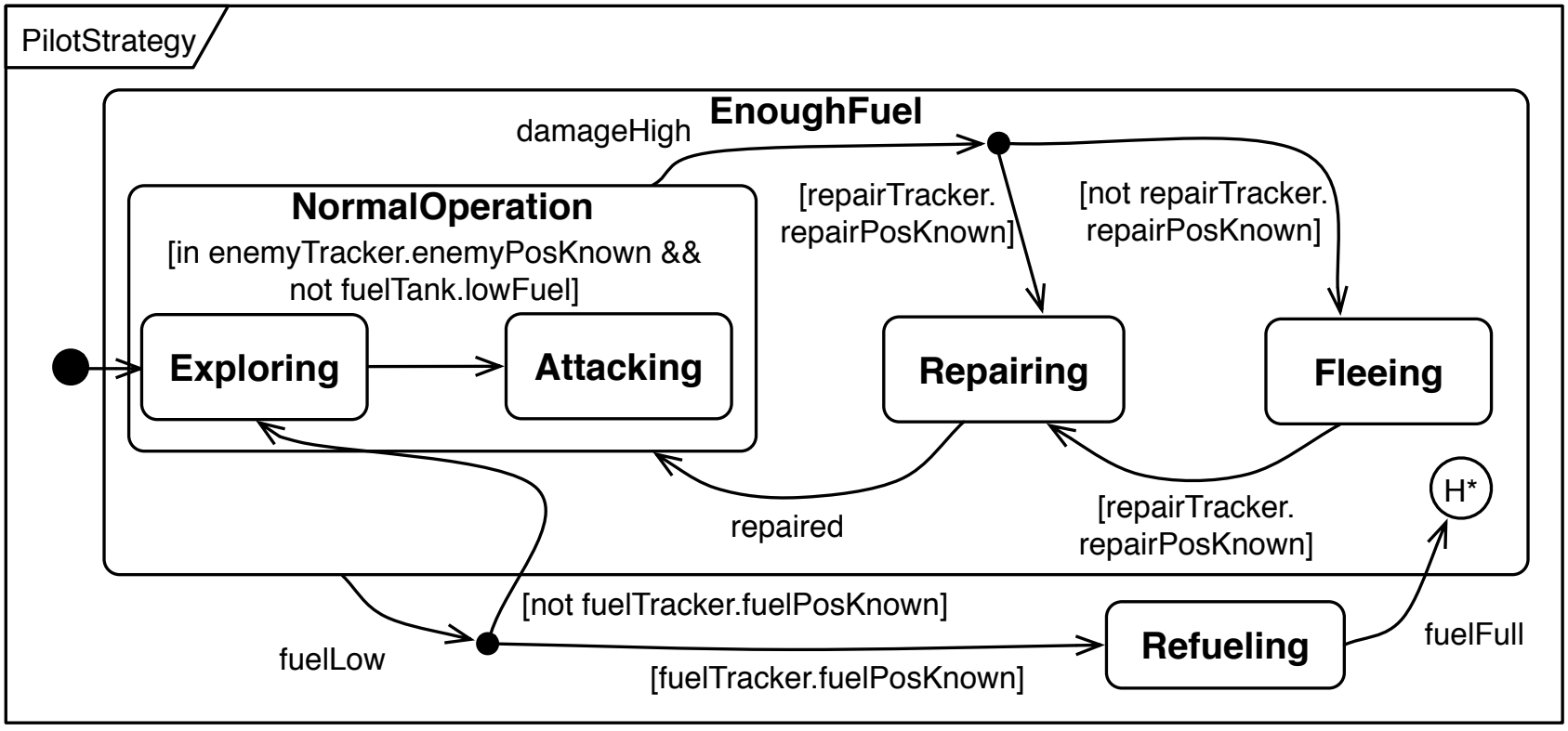
Most Appropriate Formalism (Minimizing Accidental Complexity)

`www.planeshift.it`

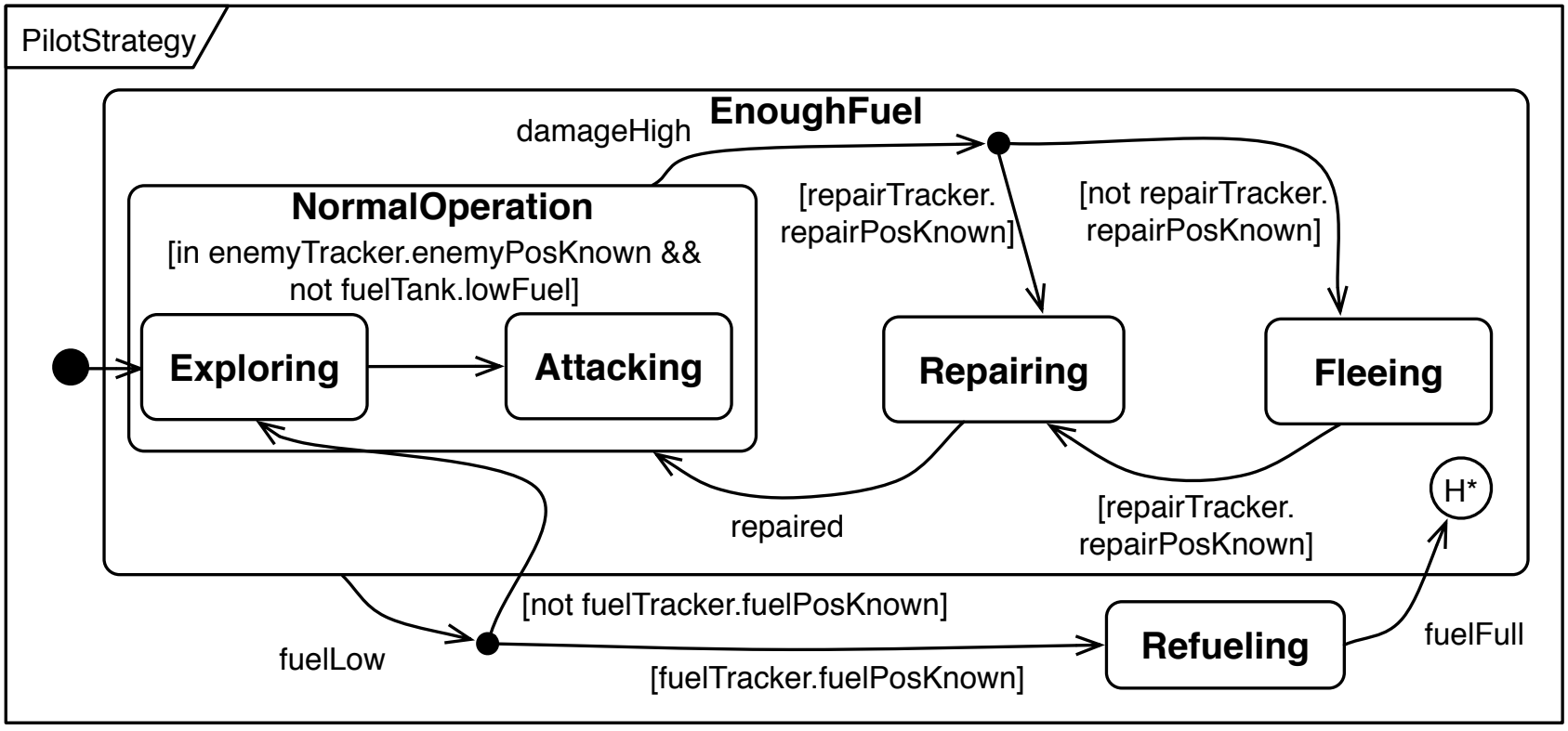
Massively Multiplayer Online Role Playing games need Non-Player Characters (NPCs)

TankWars: high level





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

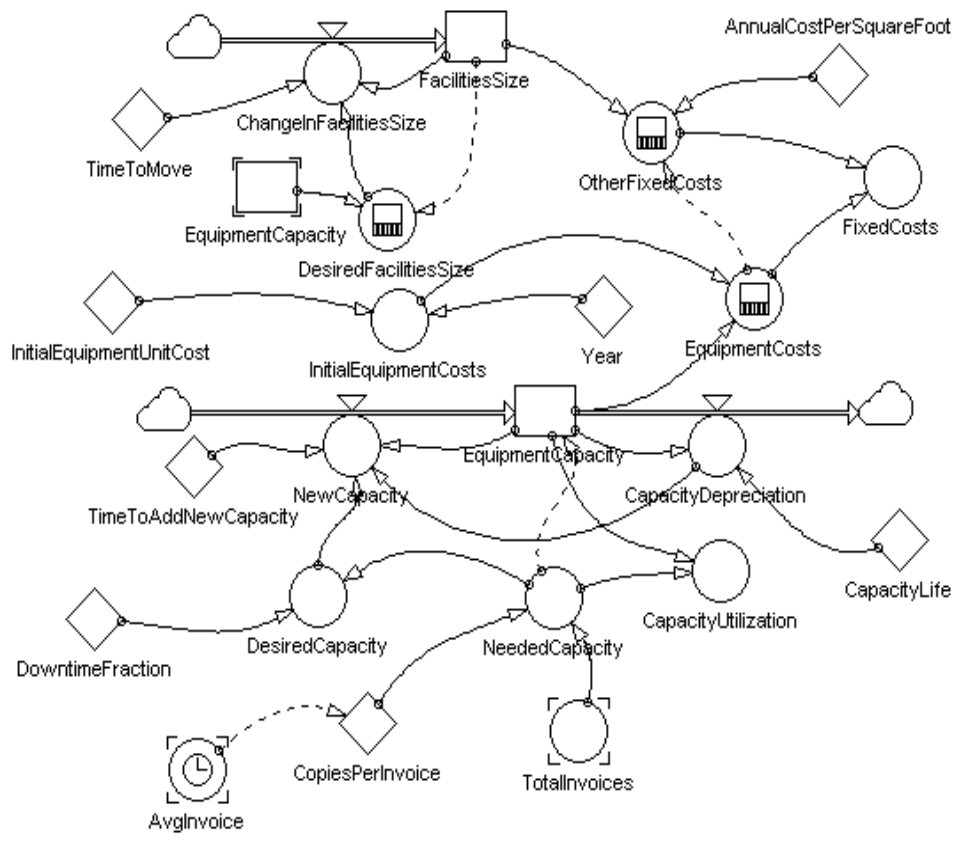


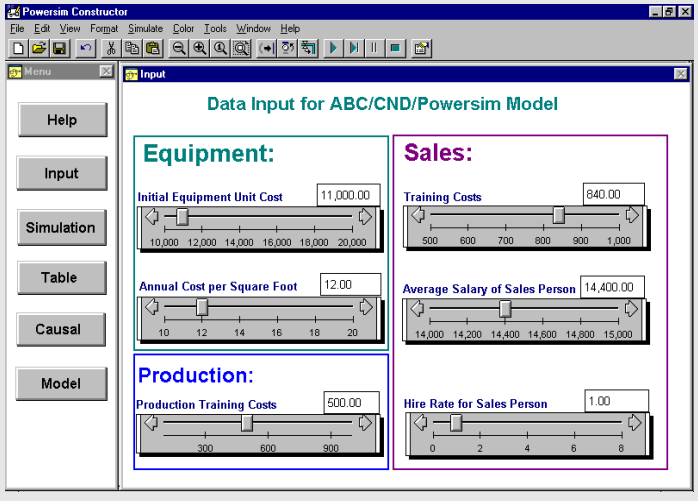
Jörg Kienzle, Alexandre Denault, Hans Vangheluwe. Model-Based Design of Computer-Controlled Game Character Behavior. MoDELS 2007: 650-665

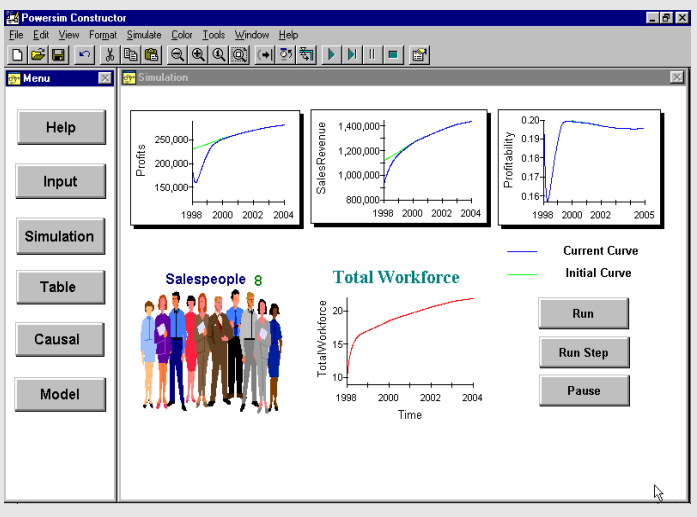
Could have used production rules instead of Statecharts

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

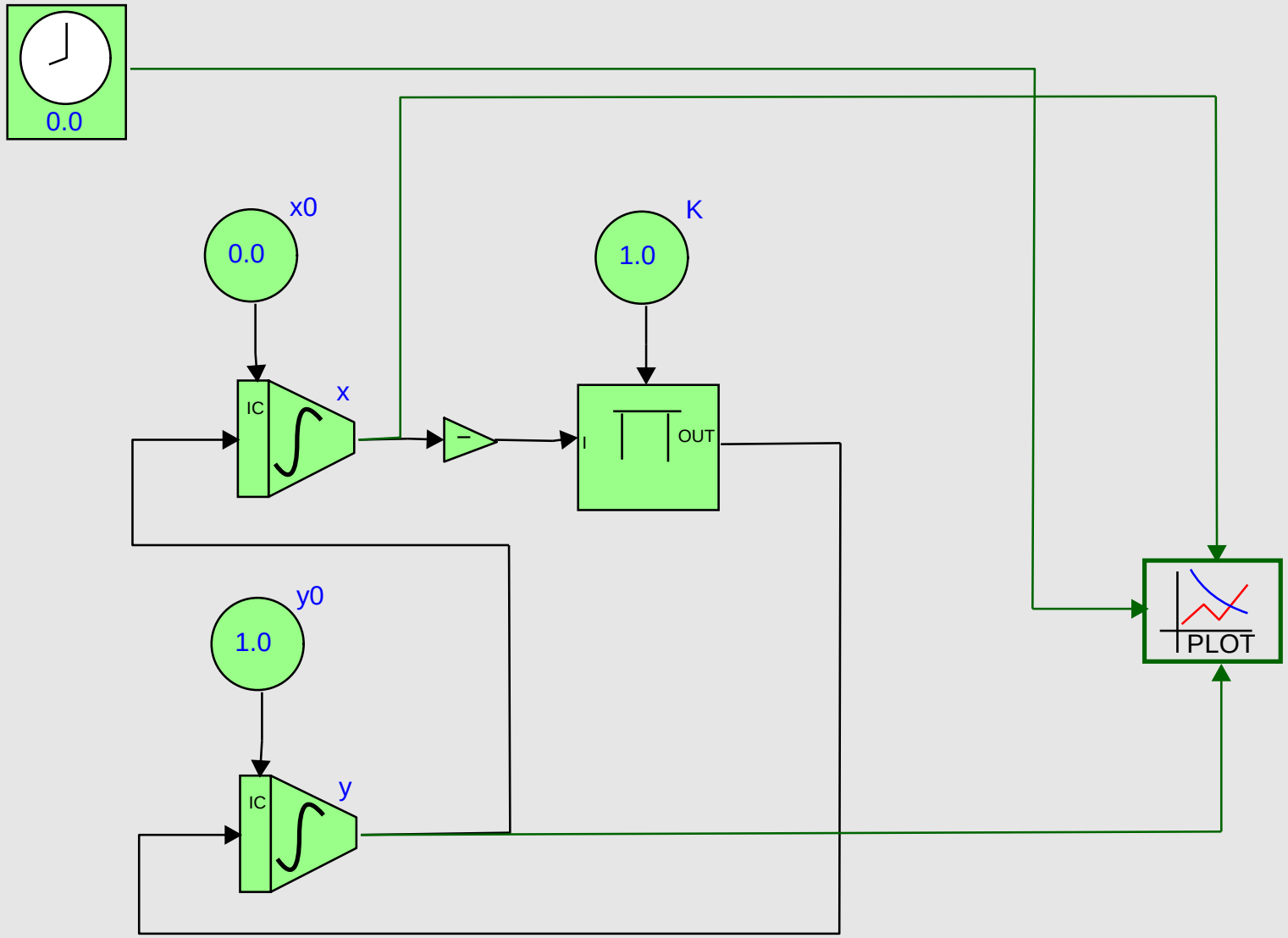
“Management Flight Simulator” using Forrester System Dynamics model



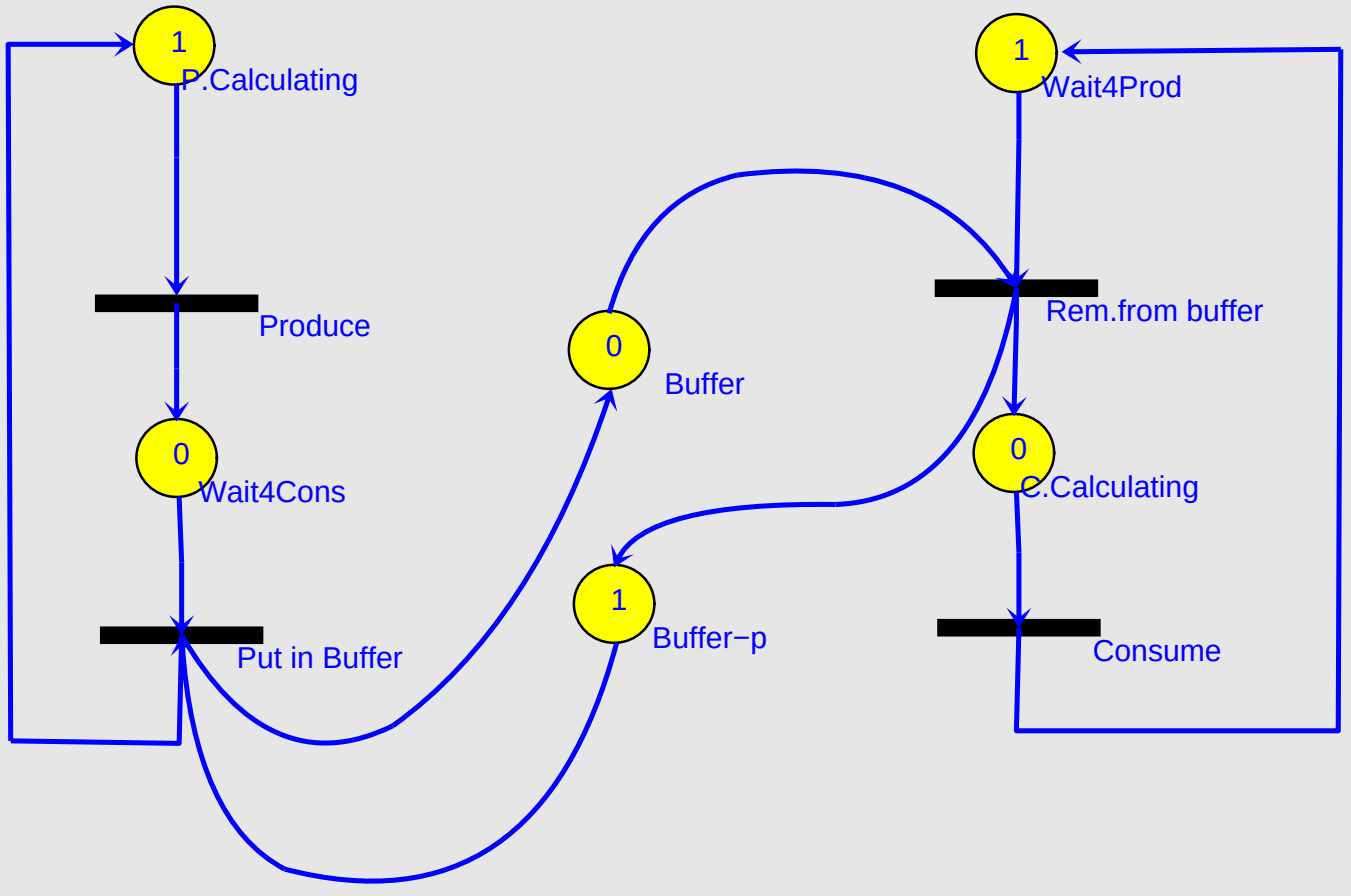




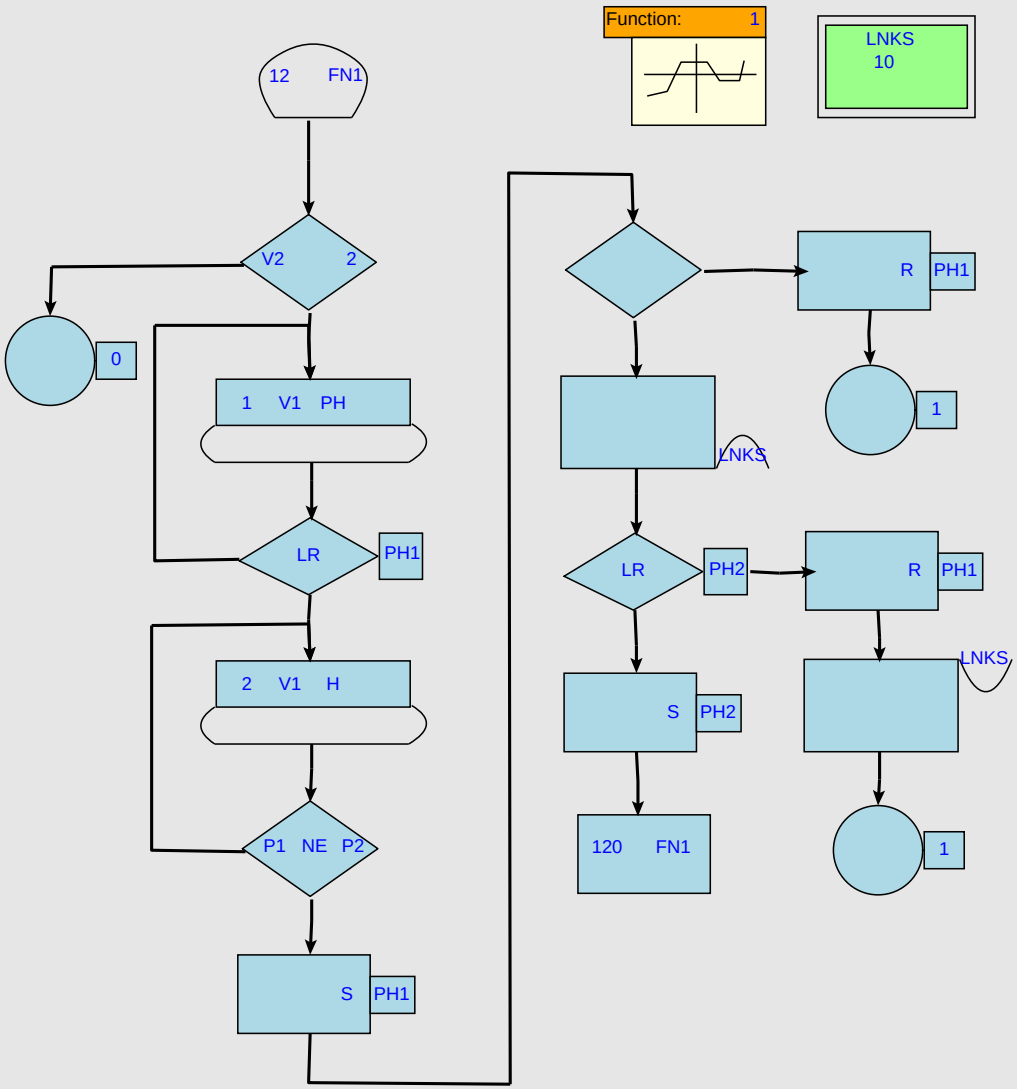
Causal Block Diagram model of Harmonic Oscillator



Petri Net model of Producer – Consumer

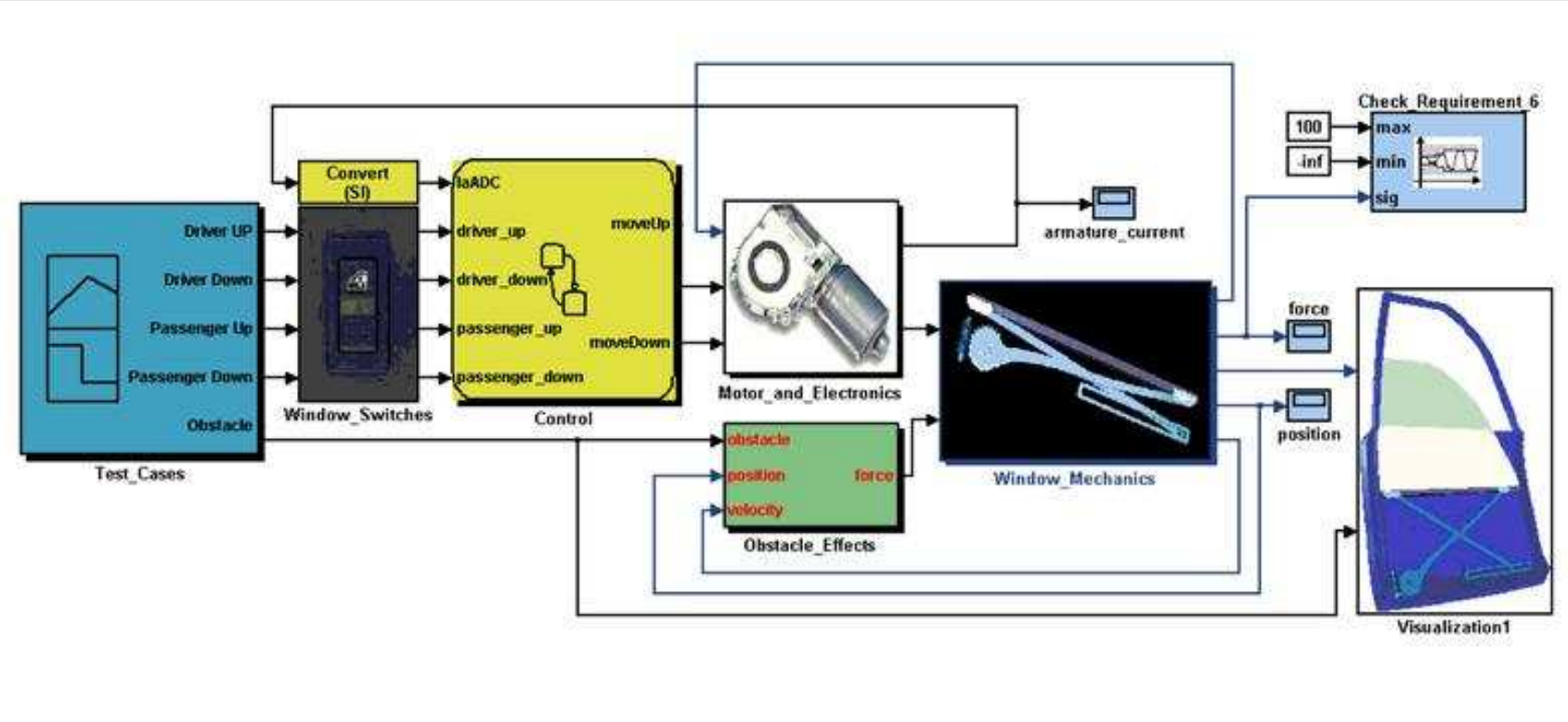


GPSS model of Telephone Exchange



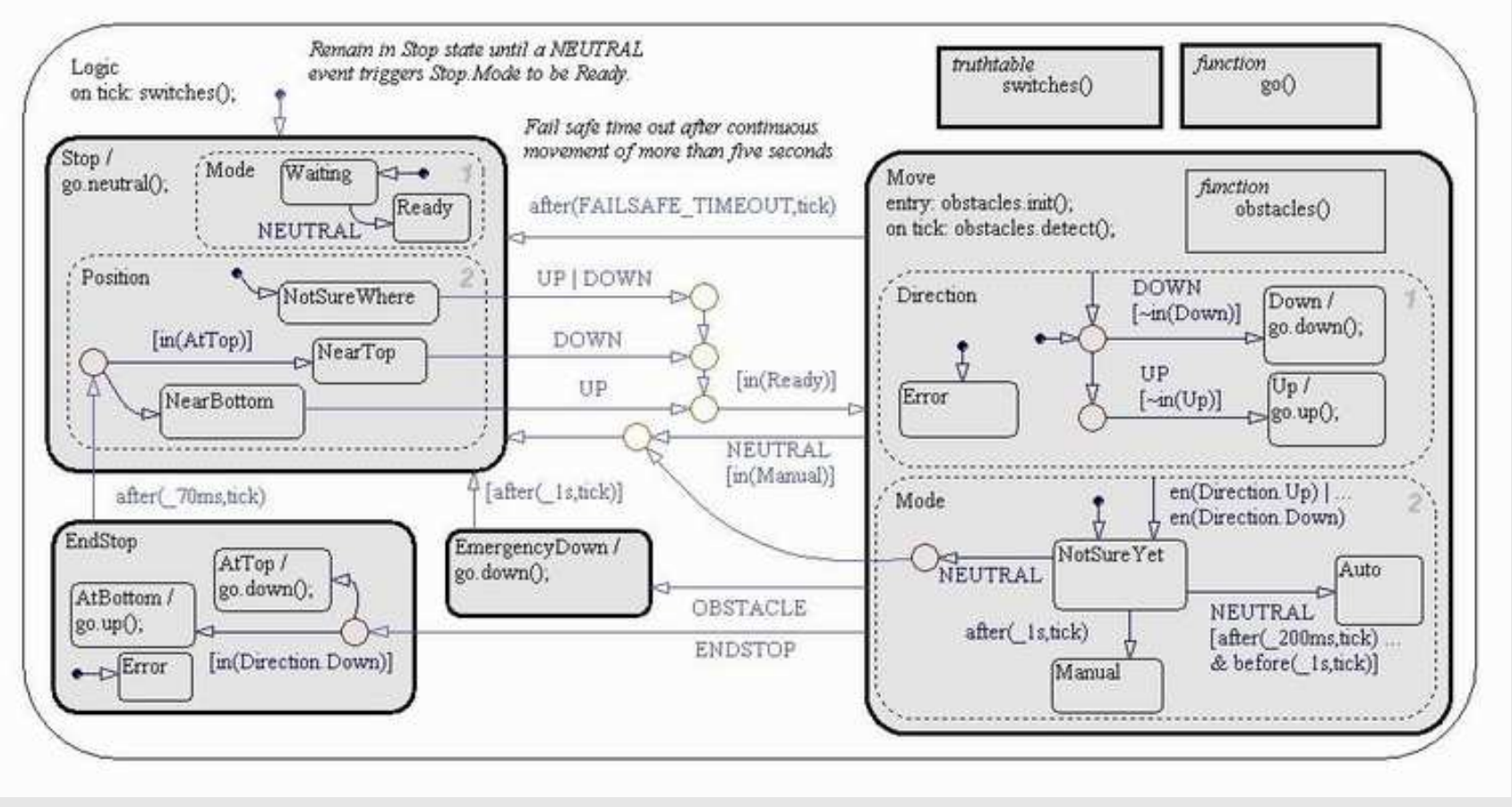
Multiple Formalisms: Power Window

Components in Different Formalisms

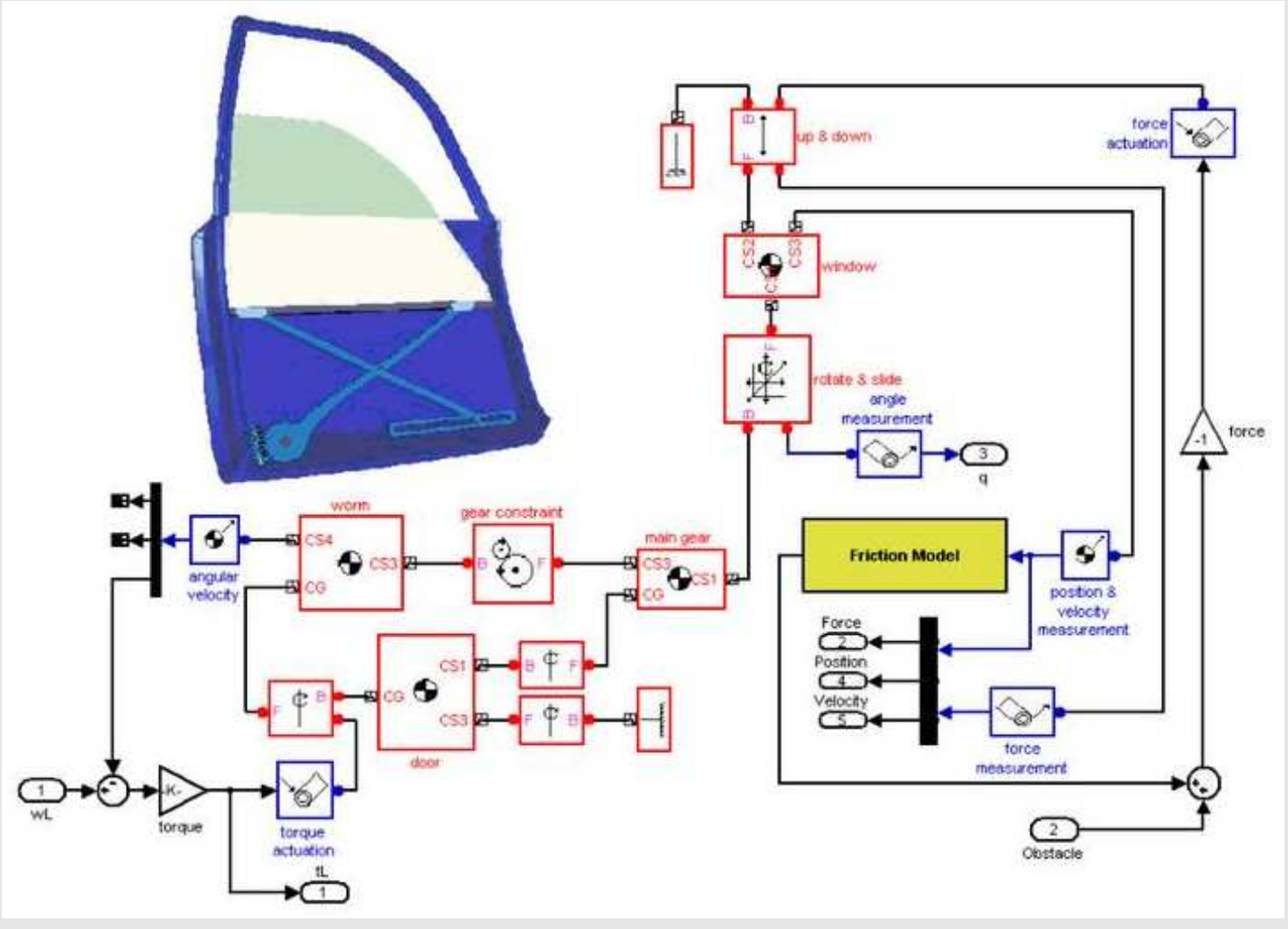


www.mathworks.com/products/demos/simulink/PowerWindow/html/PowerWindow1.html

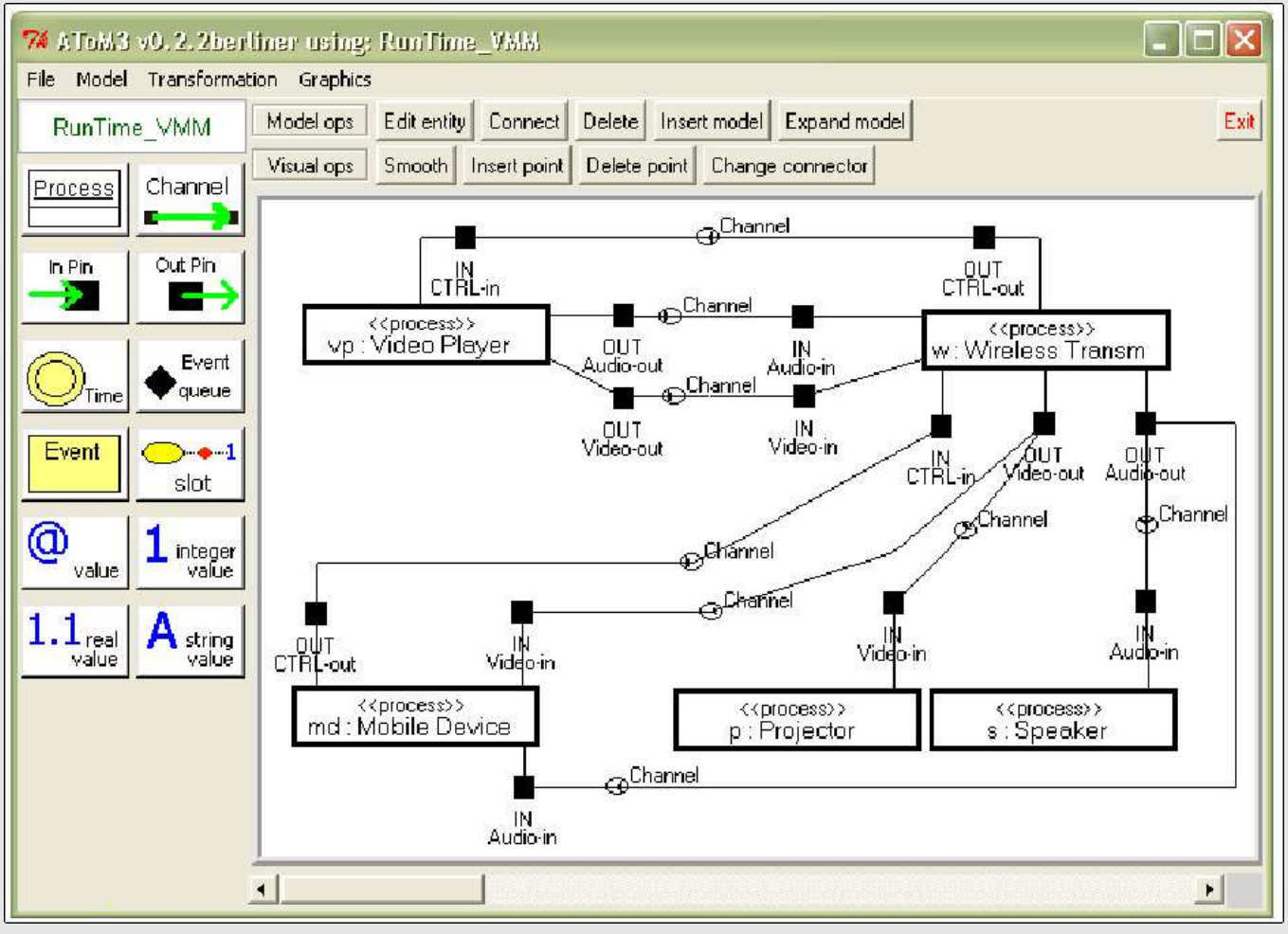
Controller, using Statechart(StateFlow) formalism



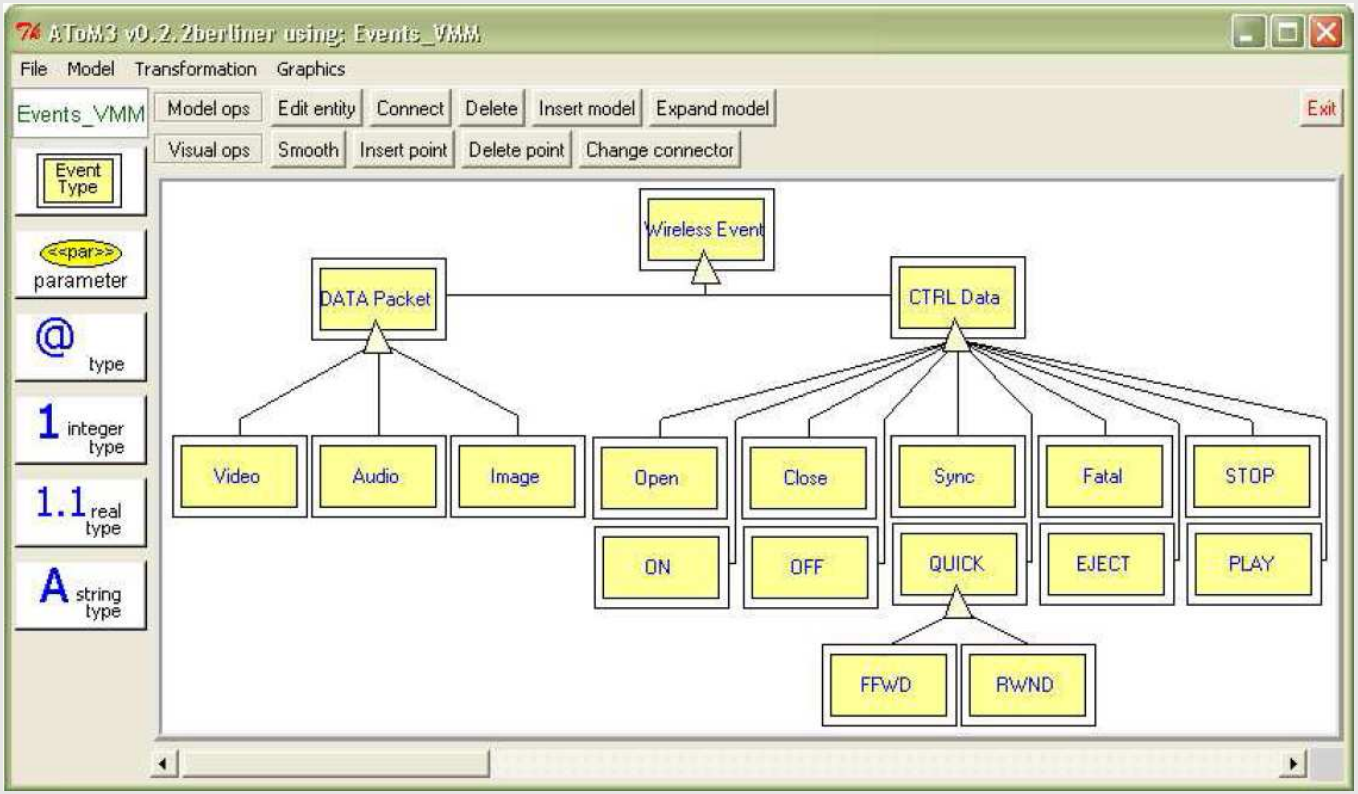
Mechanics subsystem



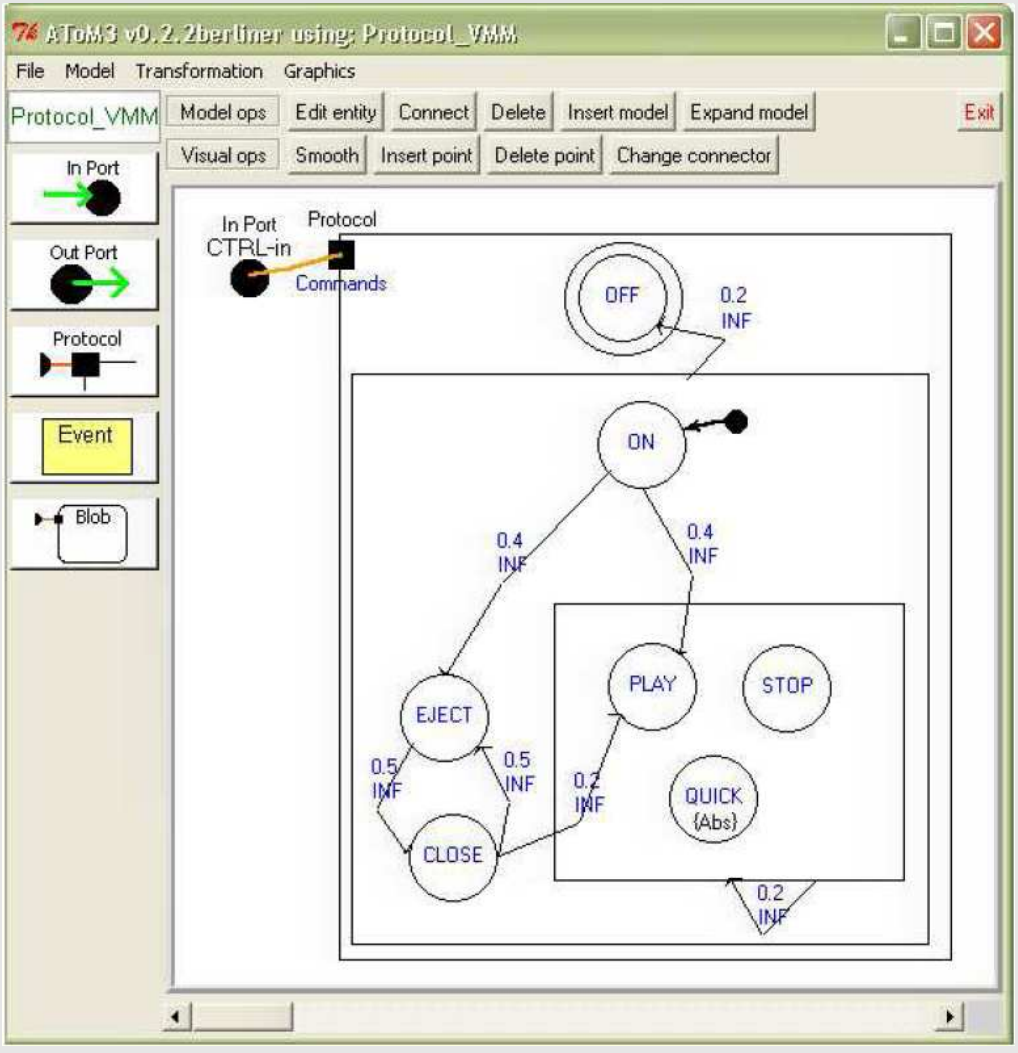
Multiple (consistent !) Views (in $\neq$ Formalisms)



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.