Modelling and Simulation to tackle Complexity

Hans Vangheluwe

1. **Modelling and Simulation**
   - Modelling and Simulation for . . .
   - The Modelling Relationship
2. **Causes of Complexity**
   - Large Number of Components
   - Diversity of Components
   - Non-compositional/Emergent Behaviour
   - Uncertainty
3. **Dealing with Complexity**
   - Multiple Abstraction Levels
   - Optimal Formalism
   - Multi-Formalism
   - Multiple Views/Aspects
4. **Multi-Paradigm Modelling**

Simulation . . . when too costly/dangerous

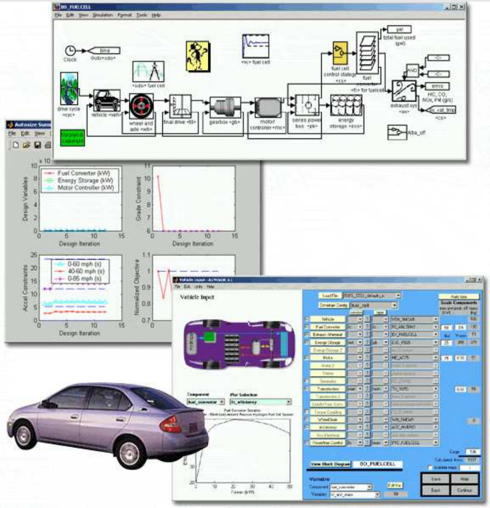
analysis $\leftrightarrow$ design

Simulation . . . real experiment not ethical

"physical" simulation, training

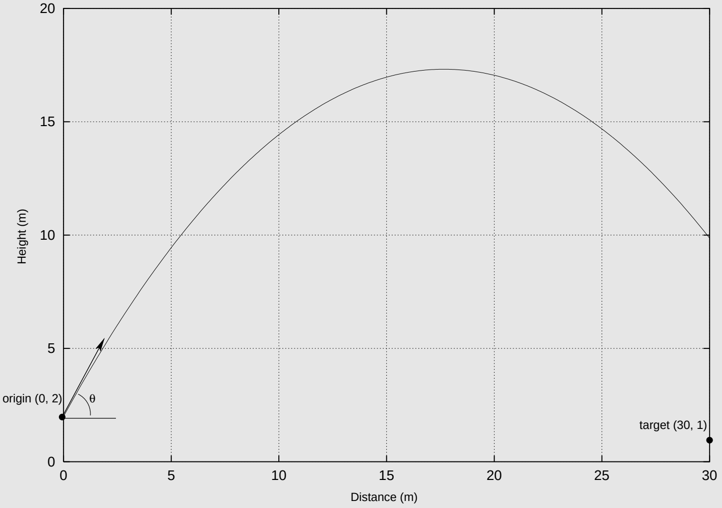
Simulation . . . evaluate alternatives

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

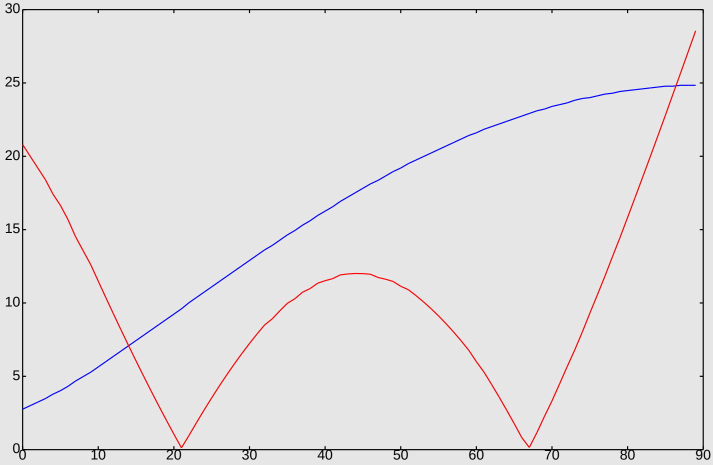


essence: “shooting” problems

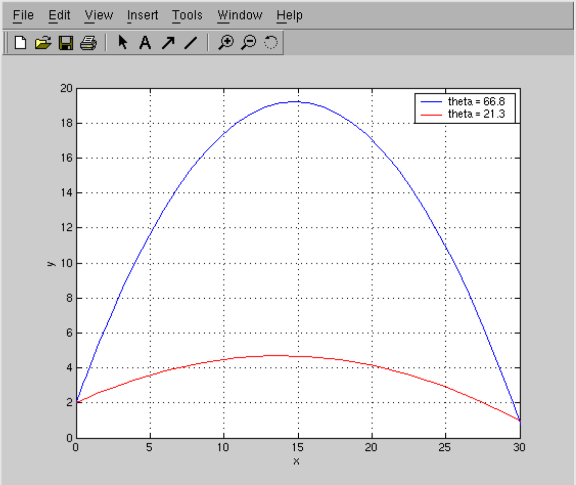
defining a “hit”



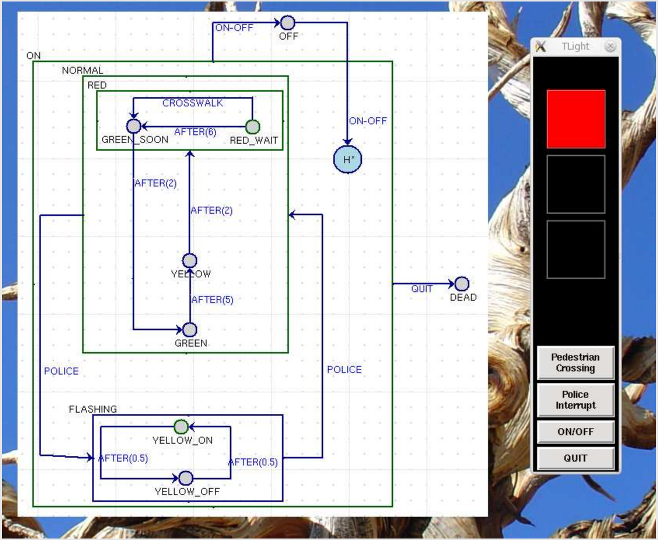
optimizing a “performance metric”



optimal solution. . . s



Modelling/Simulation . . . and code/app Synthesis



The spectrum of uses of models

- Documentation
- Formal Verification (all models, all behaviours)
- Model Checking (one model, all behaviours)
- Test Generation
- **Simulation** (one model, one behaviour)
  . . . calibration, validation, optimization, . . .
- Application Synthesis

Modelling and Simulation for . . .

Requirements ("What?")

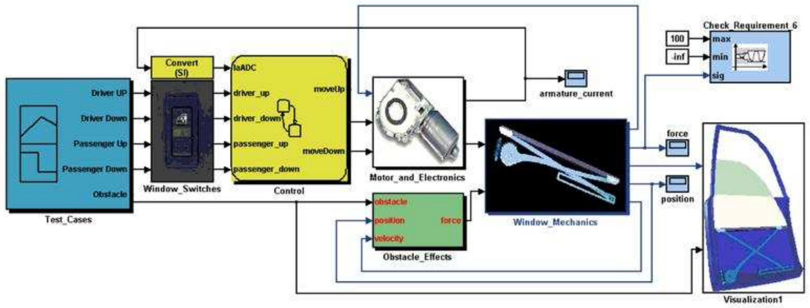
- Detached or Semi-detached
- Style (classical, modern, . . .)
- Number of Floors
- Number of rooms of different types (bedrooms, bathrooms, . . .)
- Garage, Storage, . . .
- Cellar
- Energy-saving measures
- . . .

Design ("How?")

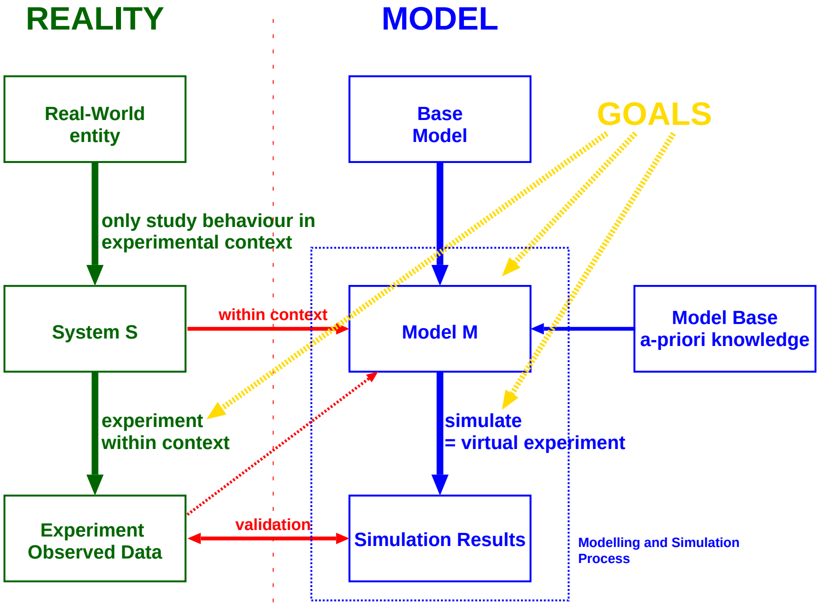
System Boundaries

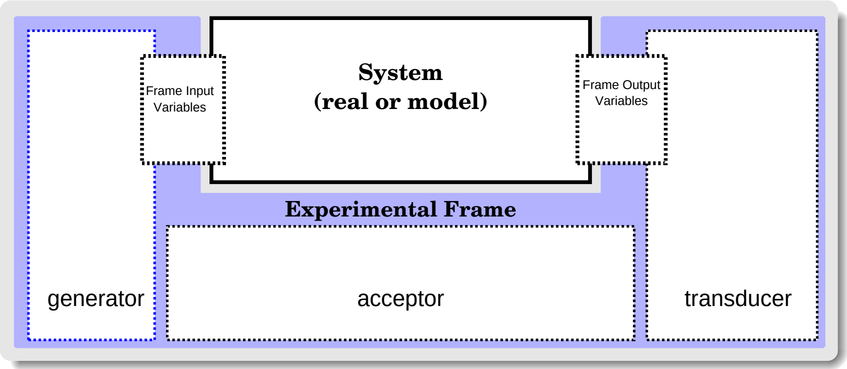
- **System** to be built/studied
- **Environment** with which the system interacts

System vs. "Plant"



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


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



- set of all “contexts” in which model is valid
- includes experiment descriptions: parameters, initial conditions

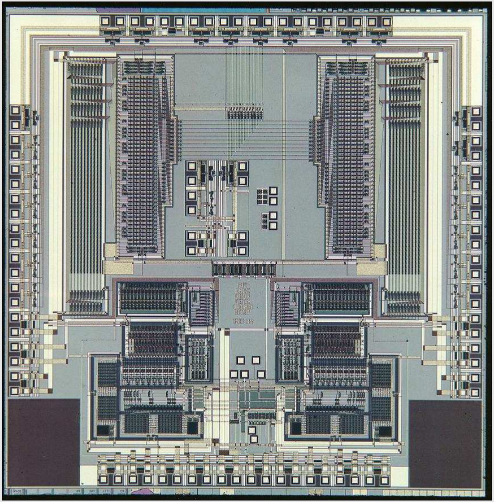
$\sim$ re-use, testing

Dealing with Complexity

Crowds

www.3dm3.com

Number of Components – hierarchical (de-)composition

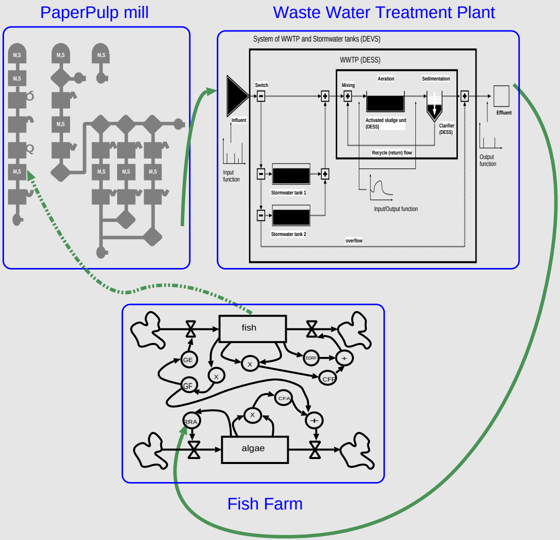


Diversity of Components: Power Window

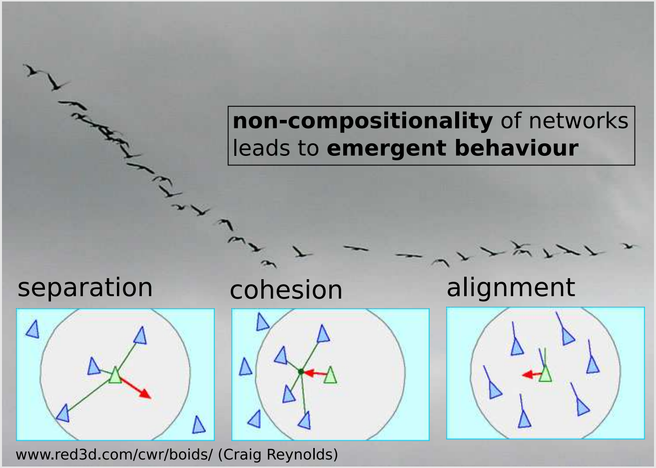
Diversity of Components: Paper Mill

www.gov.karelia.ru

Paper Mill Model



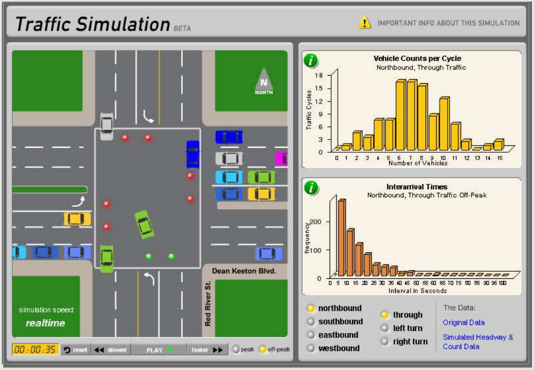
Non-compositional/Emergent Behaviour



Engineered Emergent Behaviour

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

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



`www.engr.utexas.edu/trafficSims/`

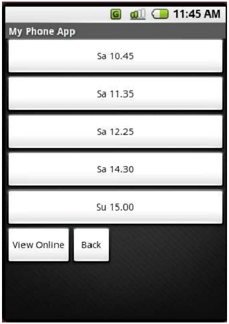
- uncertainty $\neq$ imprecise $\neq$ not rigorous

Guiding principle (~ physics: principle of minimal action)

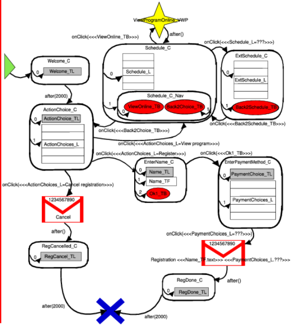
minimize **accidental complexity**,
only **essential complexity** remains

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`



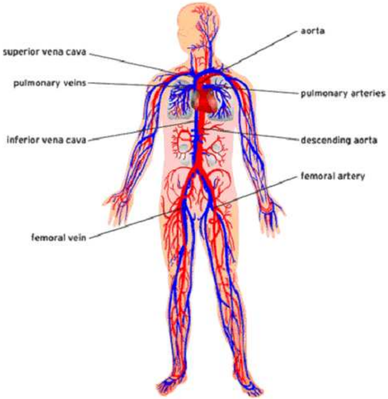
```java
package android.ger_code.PhoneApp;

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";
}
```

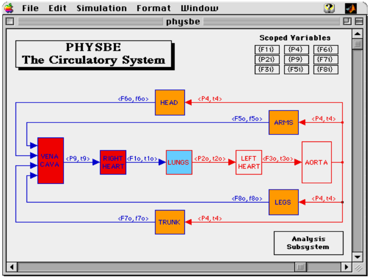


Solutions

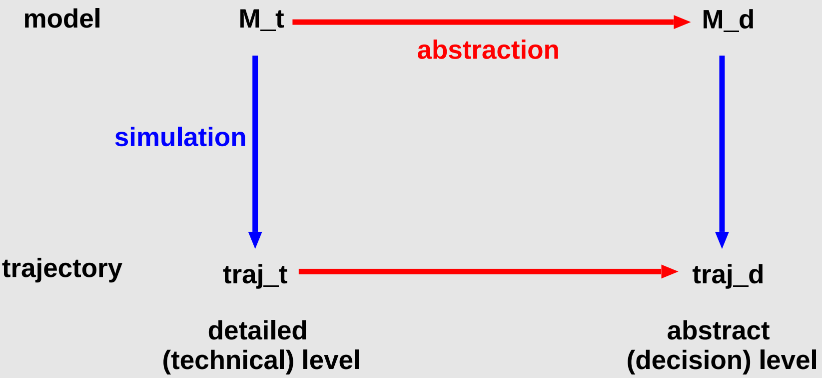
- multiple abstraction levels
- optimal formalism
- multiple formalisms
- multiple views

Different Abstraction Levels – properties preserved





Levels of Abstraction/Views: Morphism



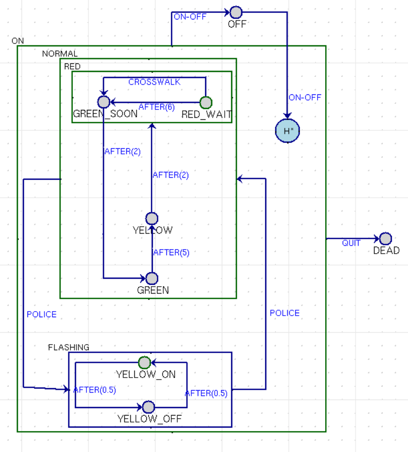
Abstraction Relationship

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.

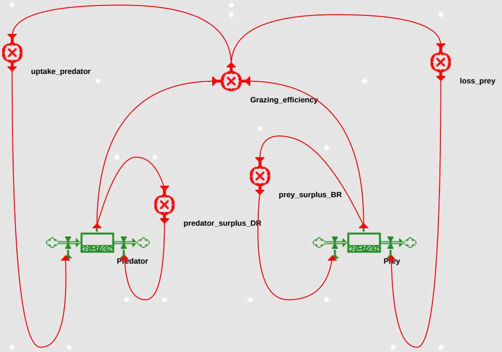
Abstraction and its opposite, *refinement* are
relative to a non-empty set of questions (properties) P.

- 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

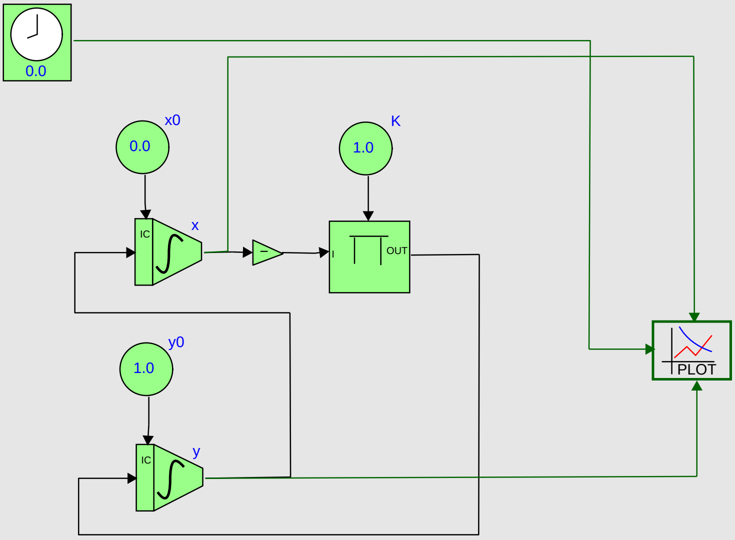


Forrester System Dynamics model of Predator-Prey interaction

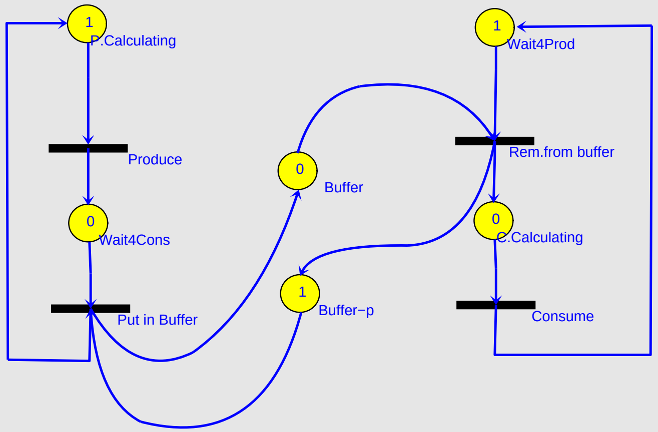


2−species predator−prey system

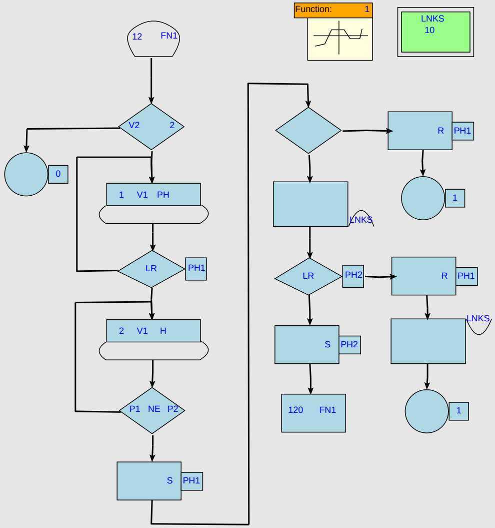
Causal Block Diagram model of Harmonic Oscillator



Petri Net model of Producer – Consumer

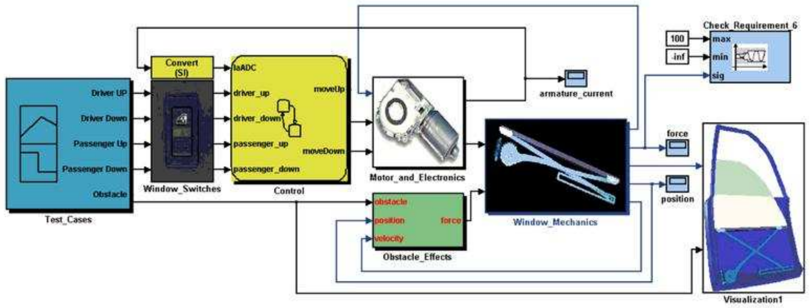


GPSS model of Telephone Exchange



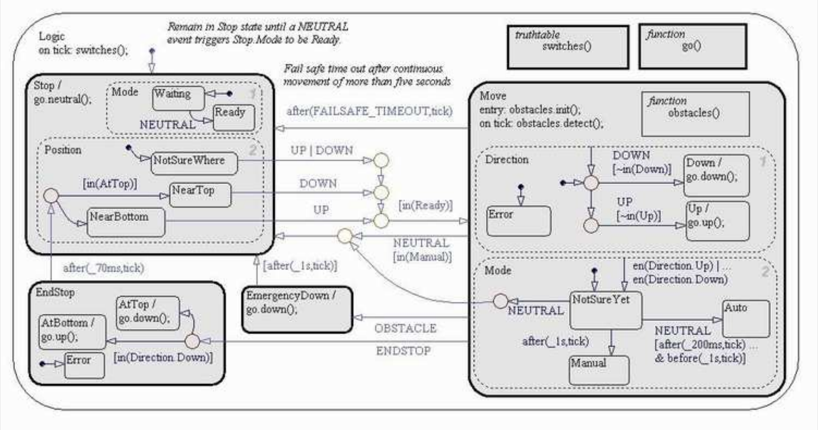
Multiple Formalisms: Power Window

Components in Different Formalisms

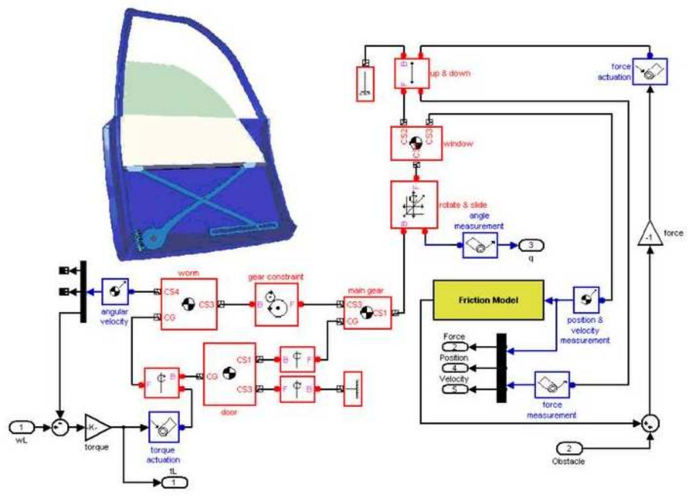


www.mathworks.com/products/demos/simulink/PowerWindow/html/PowerWindow1.html

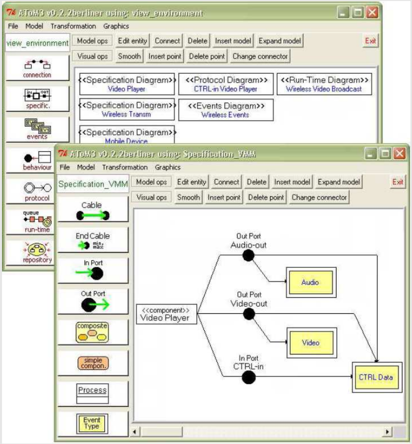
Controller, using Statechart(StateFlow) formalism



Mechanics subsystem

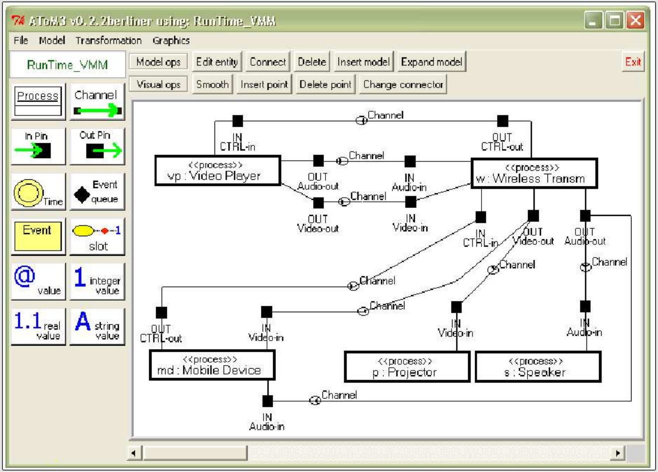


Multiple (consistent !) Views (in $\neq$ Formalisms)

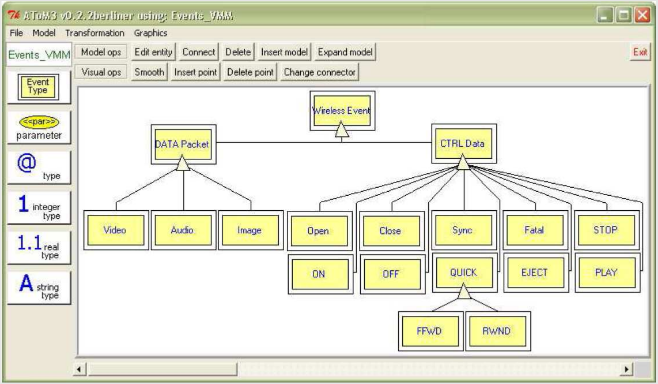


(work by Esther Guerra and Juan de Lara)

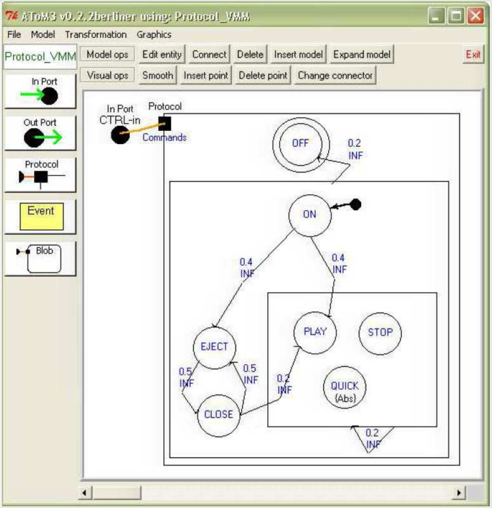
View: Runtime Diagram



View: Events Diagram



View: Protocol Statechart



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
(minimize *accidental complexity*)

- at the most appropriate **level of abstraction**
- using the most appropriate **formalism(s)**
  Differential Algebraic Equations, Petri Nets, Bond Graphs, Statecharts, CSP, Queueing Networks, 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.