System Dynamic Business Process Modelling and Simulation Tool based on OpenModelica

Tommi Karhela, Teemu Lempinen, Hannu Niemistö
VTT

Background

- OpenProd WP2
  - Ontology based Simulation Tool prototype for Complex Business and Work Processes.

- Requirement analysis in summer 2009
- First prototypes on Simantics platform in fall 2009

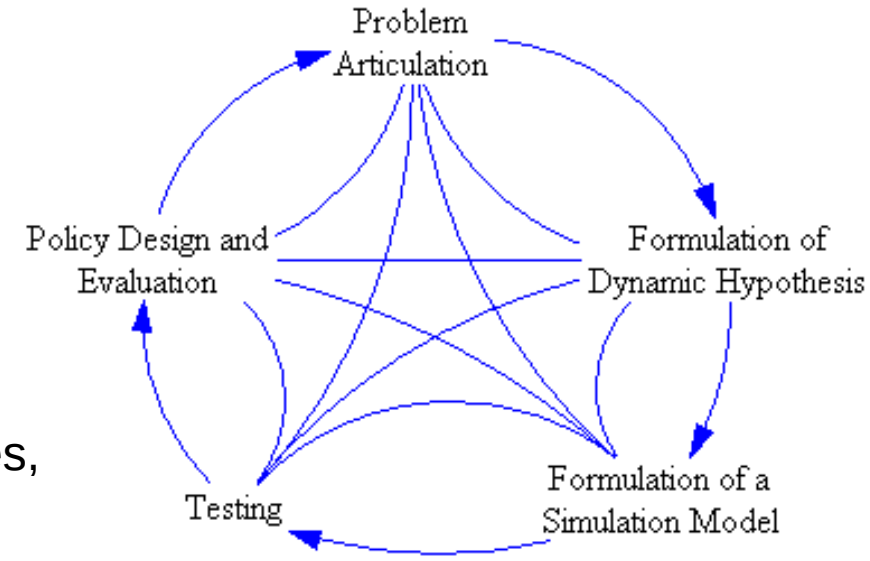
What is System Dynamics?

- A method of creating models of dynamical systems, in particular for work and business processes
- A graphical notation for the structure of ordinary differential equations.

- Created in 1950s by Professor Jay Forrester
- Empathizes the structure of the diffential equations in understanding their global behavior.
  - Balancing and reinforcing causal loops.
- Usually it is only possible to make models that behave qualitatively correct

Modeling business processes

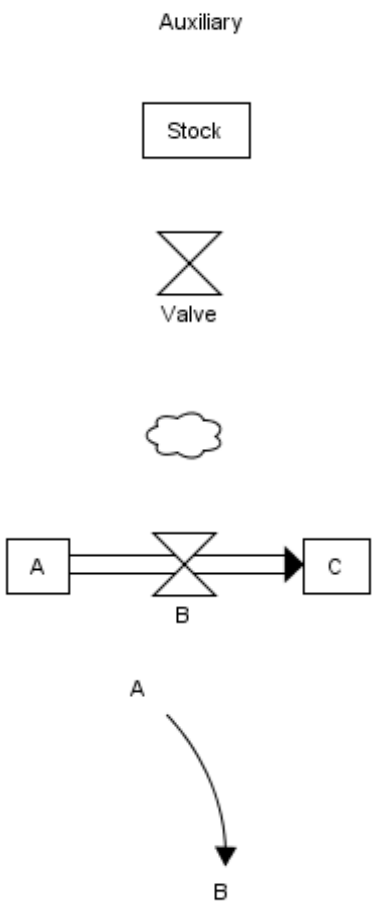
- Problem Articulation
  - Problem, variables, time scale, interfaces…
  - (Model only aspects relevant to the problem)
- Formulation of Dynamic Hypothesis
  - Hypothesis, causal relations
- Formulation of a Simulation Model
  - Structure, submodels, parameter estimation, ...
- Testing
  - Comparison to historical data, sensitivity...
- Policy Design and Evaluation
  - Scenarios, new policies, strategies, ”what if”-simulation, sensitivity in different situations



Modeling business processes

- Modeling business processes is usually an interplay between a model user who knows how the process works and a model designer who knows how to model the process.
- A common language is needed for model user and model designer.

The basic ingredients



Auxiliary (stateless variable)

Stock (stateful variable)

Valve

- auxiliary that can be connected to flow

Cloud

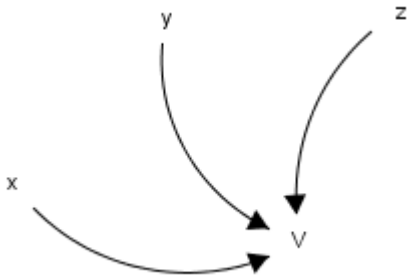
- represents a flow in or out of the system

Flow

Dependency

The basic semantics of system dynamic notation


```
der(Stock) = Incoming - Outgoing;
```




```
V = ... x ... y ... z ... ;
```

Dependencies do not represent equations, but restrict which variables can be used in equations.

Requirements for a system dynamic tool

- Configuring a model
  - Support established system dynamic notation
    - Not possible using standard Modelica diagram annotations, because the different semantics of connections
  - Hierarchical modules and model reuse
  - User defined functions
- Checking the validity of a model
  - Validating equations against dependencies
  - Physical units

Requirements for a system dynamic tool

- Simulating a model
  - Capability for stiff systems
  - Simulate on change
  - Simulate step by step
  - Partial simulation
- Analyzing a model
  - Dependency analysis
  - Sensitivity analysis
- Creating and using operating user interface
- Creating and playing games based on a model

Requirements for a system dynamic tool

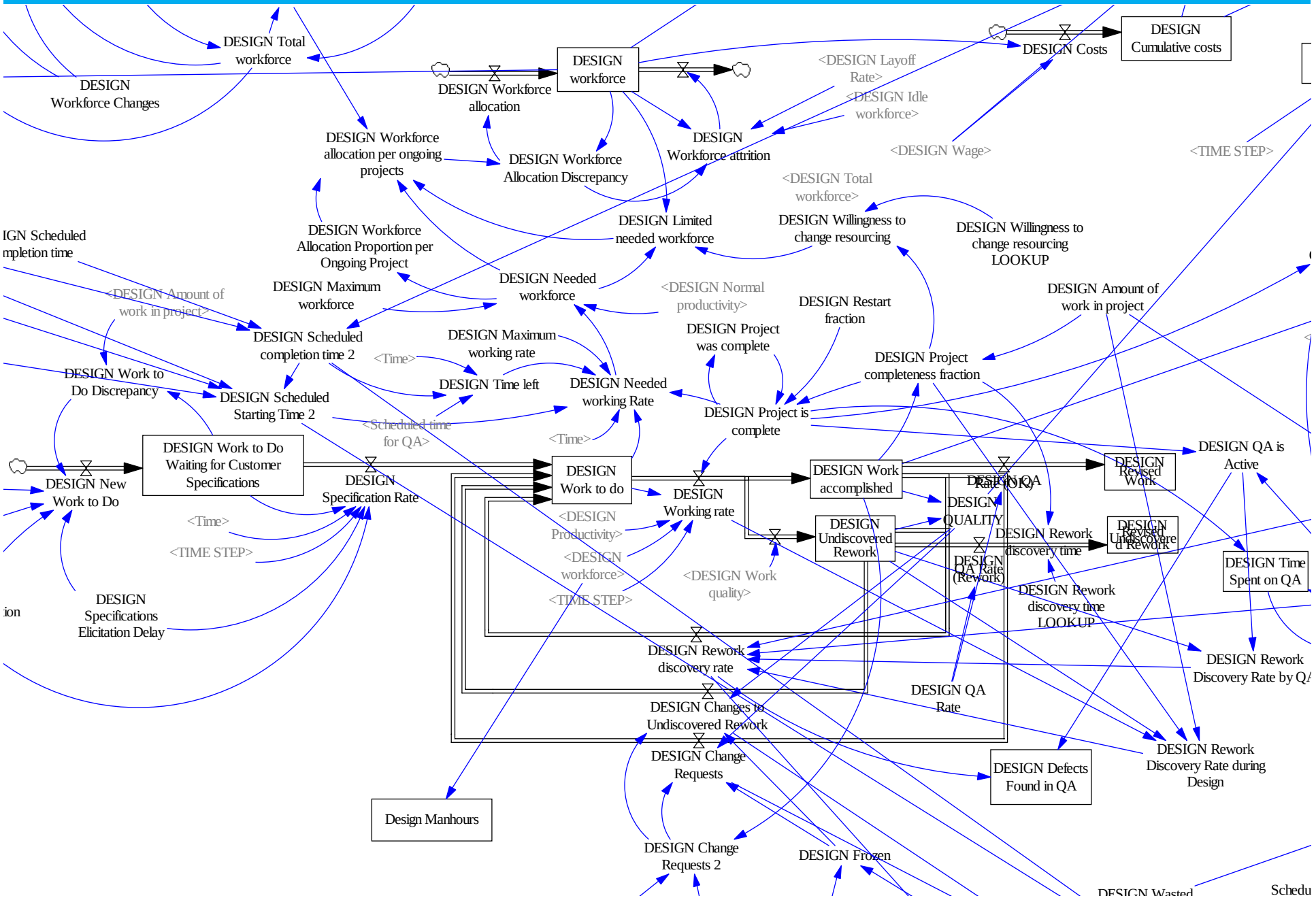
- Team features
  - Publishing a model configuration
  - Publishing the model operating interface on web
  - Shared modeling
  - Shared module libraries
  - Shared simulation
- External interfaces
  - Enterprice Resource Planning systems (ERP)
- Documentation of the model
- Mathematical analysis
- Integration to other formalisms

Implementation

- Uses Simantics platform for user interface and model persistence
  - Equation editor is custom for system dynamic tool
- Simulates models automatically on changes using OpenModelica
  - A standardized backend
  - Capable of solving stiff systems
  - Compiled models simulate quickly: useful for sensitivity analysis.

Hierarchical modules and model reuse

- Compared to physical systems, the systems studied in system dynamics have usually less locality.
- This leads to ”spaghetti models”

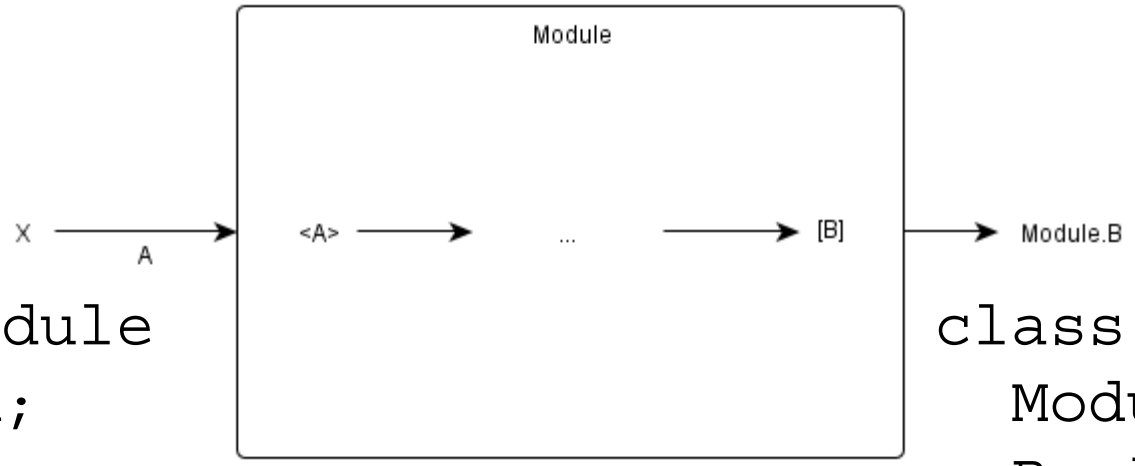

DESIGN Total workforce
DESIGN Workforce Changes
DESIGN Workforce allocation
DESIGN workforce
DESIGN Costs
DESIGN Cumulative costs
<DESIGN Layoff Rate>
<DESIGN Idle workforce>
DESIGN Workforce attrition
<DESIGN Wage>
<TIME STEP>
DESIGN Workforce allocation per ongoing projects
DESIGN Workforce Allocation Discrepancy
<DESIGN Total workforce>
DESIGN Limited needed workforce
DESIGN Willingness to change resourcing
DESIGN Willingness to change resourcing LOOKUP
DESIGN Workforce Allocation Proportion per Ongoing Project
DESIGN Maximum workforce
DESIGN Needed workforce
<DESIGN Normal productivity>
DESIGN Restart fraction
DESIGN Amount of work in project
<DESIGN Amount of work in project>
DESIGN Scheduled completion time 2
DESIGN Maximum working rate
<Time>
DESIGN Project was complete
DESIGN Project completeness fraction
DESIGN Work to Do Discrepancy
DESIGN Time left
DESIGN Needed working Rate
DESIGN Scheduled Starting Time 2
DESIGN Project is complete
<Scheduled time for QA>
<Time>
DESIGN QA is Active
DESIGN Work to Do Waiting for Customer Specifications
DESIGN Work to do
DESIGN Work accomplished
DESIGN Revised Work
DESIGN New Work to Do
DESIGN Specification Rate
DESIGN Working rate
DESIGN QA Rate (OK)
DESIGN QUALITY
<DESIGN Productivity>
DESIGN Undiscovered Rework
DESIGN Rework discovery time
DESIGN Revised Undiscovered Rework
DESIGN Time Spent on QA
<Time>
<TIME STEP>
<DESIGN workforce>
<DESIGN Work quality>
DESIGN QA Rate (Rework)
DESIGN Rework discovery time LOOKUP
DESIGN Specifications Elicitation Delay
DESIGN Rework discovery rate
DESIGN Rework Discovery Rate by QA
DESIGN QA Rate
DESIGN Changes to Undiscovered Rework
DESIGN Rework Discovery Rate during Design
DESIGN Change Requests
DESIGN Defects Found in QA
Design Manhours
DESIGN Change Requests 2
DESIGN Frozen
DESIGN Wasted
Schedu

Hierarchical modules and model reuse

Different approaches:

- Create symbols with terminals for modules (like in Modelica)
  - Problem: because the lack of locality, much more terminals needed than in typical process/automation components
  - Would also be stylistically different to basic system dynamic components
- Create new special auxiliary variables that lift variables in the modules

Module example




```
class Module
  Real A;
  Real B;
protected
  ...
equation
  B = ...;
  ...
end Module;
```

```
class Module
  Module M1;
  Real X;
equation
  M1.A = X;
  ... M1.B ...;
end Module
```

Other ways for managing dependencies

- Inner/outer
- Notation for dependencies to multiple targets

Operating user interfaces

- Rapid UI-development for models
- Game interfaces where different roles have different viewpoints to the model

Integration to other formalisms

- Activity diagrams (BPMN)
- Used to model business processes
- With certain restrictions, it is possible to generate system dynamic model that computes expected value of the states of activity diagram.
- Problems:
  - Stateful delays
  - Joins, in particular, if the diagram contains different kind of delays