Omola, OmSim och K2 - en kort kurs

Nilsson, Bernt; Eborn, Jonas; Mattsson, Sven Erik

1995

Document Version:
Förlagets slutgiltiga version

ISSN 0280–5316
ISRN LUTFD2/TFRT--7535--SE

Omola, OmSim och K2 — en kort kurs

Bernt Nilsson
Jonas Eborn
Sven Erik Mattsson

Department of Automatic Control
Lund Institute of Technology
June 1995

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
TECHNICAL REPORT

Date of issue
June 1995

Document Number
ISRN LUTFD2/TFRT--7535--SE

Author(s)
Bernt Nilsson, Jonas Eborn and Sven Erik Mattsson

Supervisor

Sponsoring organisation
Sydkraft AB and NUTEK, proj.nr. 9304688-2

Title and subtitle
Omola, OmSim och K2 – en kort kurs
(Omola, OmSim and K2 – a short course)

Abstract

This report contains a set of lecture notes of a short course in object-oriented modelling in general and Omola in particular. The Omola simulation environment, OmSim, is presnted and computer exercises are included. A presentation of the K2 model database for thermal power plant modelling is also included. The course was give in october 1994 for industrial control engineers from Sydkraft Konsult AB.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
37

Recipient's notes

Security classification

Omola-kurs

lektion I

Objekt-orienterad modellering och Omola

Innehåll:

1. Bakgrund och grundläggande begrepp.
2. Omola - gunder
3. Primitiva modeller
4. Sammansatta modeller
5. OmSim

Bakgrund

Simnon utvecklat under 70-talet av Hilding Elmqvist. Bygger på CSSL. Blev produkt mha STU-lån.

Dymola är Elmqvists avhandling. Introducerar strukturerad modellering. Var "glömt" under tio år.

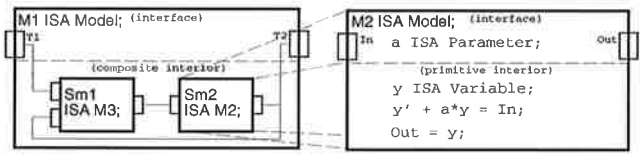
CACE var ett projekt under andra delen av 80-talet med Sven Erik Mattsson som projektledare.

Omola definerades av Mats Andersson 1989. Lägger till objektorienterade idèer till strukturerad modellering.

OmSim implementeras under 90-91.

Dymola återuppstår. Nu med objektorienterade koncept. Blir produkt mha DLR.

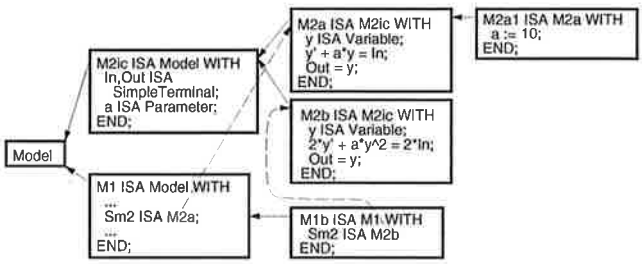
Abstraction



A model object is described in two parts.

- **Interface** describes the interaction with other models and model user.
- **Interiour** describes the model behaviour, composite or primitive.

Reuse



Inheritance facilitiates development.

- **Reuse** of models in composite models.
- **Specialization** of a predefined model into a new one.
- **Polymorphism** for the reuse of structures.

Omola I

Basic Omola

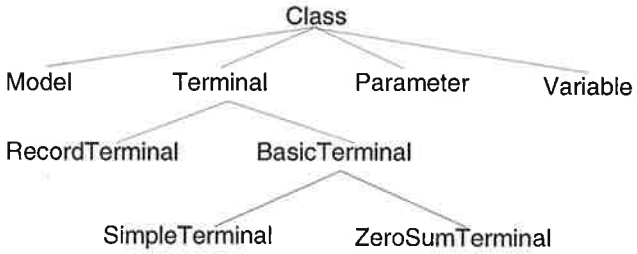
```
{name} ISA {name of super class} WITH
  {class body}
  {with list of attributes}
END;
```

The list of attributes are a set of model components:

- **terminals and parameters** for the interface.
- **submodels and connections** for composite interiour.
- **equations and variabels** for primitive interiour.
- **events and actions** for primitive discrete interiour.

Omola II

Predefined Classes



- **Model** is the super class for all model definitions.
- **Terminal** is the super class for a set of interaction classes.
- **Parameter** is for time invariant variables that the model user can change to adapt the model behaviour.
- **Variable** is for internal states and time variant properties.

There are additional classes for event dependent descriptions.

Omola III

Primitive Descriptions 1

- Parameters are used for adapt the model behaviour and they can be changed by the model user.

  ```
  Density ISA Parameter;
  ```

- Variables are time-varing and describe the states of the model.

  ```
  mass ISA Variable;
  ```

Omola IV

Primitive Descriptions 2

- Equations are used to describe the internal behaviour of a model.

  ```
  mass' = Density*(In - Out);
  mass  = Density*Area*level;
  ```

- Assignments describe a given computational causality.

  ```
  mass' := Density*(In - Out);
  level := mass/(Density*Area);
  ```

  This is similar to Simnon descriptions.

Omola V

Primitive Descriptions 3

A simple tank example:

```
Tank ISA Model WITH
%% A simple tank model.
parameters:
  Density ISA Parameter;
  Area, Pipe, g, In ISA Parameter;
  g := 9.81;
variables:
  mass ISA Variable;
  Out, level ARE Variable;
equations:
  % mass balance
  mass' = Density*(In - Out);
  mass  = Density*Area*level;
  Out   = Pipe*SQRT(2*g*level);
END;
```

Comments can be added by the use of %. The comment parameters: are called tag and is optional.

Omola VI

Submodel Interaction

Two terminals are defined as follows

```
T1,T2 ISA RecordTerminal WITH
  x ISA SimpleTermial;
  y ISA ZeroSumTerminal;
END;
```

A connections is defined as T1 AT T2 and is interpreted as follows:

$$\begin{aligned} T1.x &= T2.x \\ T1.y + T2.y &= 0 \end{aligned}$$

Omola VII

Composite models

```
TankSeries ISA Model WITH
terminals:
  In  ISA SimpleInFlow;
  Out ISA SimpleOutFlow;
submodels:
  Tank1, Tank2 ISA TankModel;
connections:
  In         AT Tank1.In;
  Tank1.Out AT Tank2.In;
  Tank2.Out AT Out;
END;
```

Omola VIII

Graphics

```
Layout ISA Class WITH
  % position in owner class
  x_pos TYPE Real;
  y_pos TYPE Real;

  % size of internal description
  x_size TYPE Real;
  y_size TYPE Real;

  % visible icon (terminals)
  invisible TYPE Integer := 0;

  % bitmap definition (optional)
  bitmap TYPE String := "valve";
END;
```

Layout is the super class to the automatically created Graphic attribute of teminals, submodels and models.

OmSim I

Composite model example

OmSim II

Modellutveckling

- Nya bibliotek görs med text-editor, Emacs, **utanför** OmSim.
- Editering av primära modeller görs med text-editor, Emacs, **inuti** OmSim.
- Skapande av nya modeller och editering av sammansatta modeller görs i MED (verktyg i OmSim).
- Ikoner görs i bitmap-editor, `bitmap` eller `xv`, **utanför** OmSim i separata filer.
  - Dymograph ersätter MED och `bitmap` i en snar framtid. Till en början som separat program **utanför** OmSim.

OmSim III

Simulering

Numeriska lösare

- explicit ODE, $\dot{x} = f(t, x)$
- implicit ODE, $B\dot{x} = f(t, x)$ (B konstant)
- DAE, $g(t, \dot{x}, x) = 0$

Modellmanipulering

- koll av ofullständigheter
- underlätta numerik:
  - sortera
  - reducera DAE
  - eliminera alg. var.
  - transformera till ODE.

OmSim IV

OCL - OmSim Command Language

Exempel på vad OCL kan:

- Skapa simulator, plot- och variabelfönster.

```
Simulator s(Tank);
Plot p(Tank);
```

- Knytas variabler till plottar.

```
P.y(height);
```

- Sätta parametrar.

```
Tank.Area := 5;
```

- Göra simuleringar.

```
s.start;
```

Omola-kurs

lektion II

Avancerad Omola 1 och 2

Simulering

Innehåll:

1. Modellkomponenter
2. Ärvning
3. Bibliotek
4. Simulering

Avancerad Omola 1:

Modellkomponenter och dess egenskaper

TYPE-begreppet

Variabler kan defineras som olika typer.

```
x TYPE Real;
i TYPE Integer;
s TYPE String := "water";
b TYPE Symbol := 'OmSim;
e TYPE (Gas, Water, Steam) := 'Gas;
r TYPE row(2) := [1, 2];
c TYPE column(3) := [1; 2; 3];
m TYPE matrix(2,2) := [1, 2; 3, 4];
d TYPE DISCRETE Real;
```

Värde–semantik

En viktig egenskap hos variabler, parametrar och enkla terminaler är värde-semantik. Dessa klasser har ett `value`-attribut (och ett `default`).

```
Variable ISA Class WITH
  value TYPE Real;
  default TYPE Real;
END;
```

Om man refererar till en variabel så är det underförstått att man refererar till dess value-attribut.

```
p ISA Parameter;
v ISA Variable;
v = 2*p;
```

Betyder egentligen `v.value = 2*p.value;`

Vektorer och Matriser

Variabel- och parameterklasser med nya typer skapas genom att göra om typ-deklarationen på value och default.

```
MatrixParameter ISA Parameter WITH
 value   TYPE matrix(2,2);
 default TYPE matrix(2,2):=zeros(2,2);
END;
```

Genom att addera en parameter kan dessa generaliseras:

```
ColumnVariable ISA Variable WITH
 m       TYPE Integer;
 value   TYPE matrix(m,1);
 default TYPE matrix(m,1):=zeros(m,1);
END;
```

På liknande sätt erhålls parametrar och variabler av andra typer.

Enkla terminaler

```
BasicTerminal ISA Terminal WITH
  value       TYPE Real;
  default     TYPE STATIC Real;
  quantity    TYPE STATIC String:="number";
  unit        TYPE STATIC String:="1";
  variability TYPE STATIC
    (TimeVarying,Parameter):='TimeVarying;
END;

SimpleTerminal ISA BasicTerminal WITH
  causality TYPE STATIC
    (Undefined,input,output):='Undefined;
END;

ZeroSumTerminal ISA BasicTerminal WITH
  direction TYPE STATIC (in,out):='in;
END;
```

Enkla terminaler motsvarar interaktionsvariabler.

Värdesemantik

Sammansatta terminaler

Komplex interaktion beskrivs med ett knippe enkla terminaler

```
RecordTerminal ISA Terminal;
```

Terminalhierarki-exempel:

```
TwoPhasePipe ISA RecordTerminal WITH
  SteamPhase ISA RecordTerminal WITH
    q ISA ZeroSumTerminal;
    T, p ISA SimpleTerminal;
  END;
  WaterPhase ISA RecordTerminal WITH
    q ISA ZeroSumTerminal;
    T, p ISA SimpleTerminal;
  END;
END;
```

Punktnotation och Parameterisering

Punktnotation utnyttjas för att nå komponenter i strukturhierarkin.

```
TankSystem ISA Model WITH
terminals:
  u, y ISA SimpleTerminal;
submodels:
  Tank1, Tank2 ISA TankModel;
parameter_assignment:
  Tank2.Area := 2*Tank1.Area;
connections:
  Tank1.In.q := u;
  y := Tank2.Out.q;
END;
```

Avancerad Omola 2:

Omola och Ärvning

Klass-begreppet

En modell som är subklass till en annan modell **ärver** alla dess attribut.

```
M1 ISA Model WITH
  mass     ISA Variable;
  enthalpy ISA Variable;
END;

M2 ISA M1 WITH
  pressure ISA Parameter;
END;

M3 ISA Model WITH
  Area ISA Parameter
  MyM2 ISA M2 WITH
    Area ISA Parameter
  END;
END;
```

M2 har tre attribut, 2 variabler som är ärvda och 1 parameter som är lokalt deklarerad.

M3 har två attribut medan MyM2 har fyra attribut.

Överskrivning

Ett ärvt attribute kan omdefineras, *skrivas över.*

```
M1 ISA Model WITH
  mass ISA Parameter;
END;

M2 ISA M1 WITH
  mass ISA Variable;
END;
```

- attribut med namn kan skrivas över, inte ekvationer och kopplingar.
- ärvda attribut kan inte tas bort.

Klasshierarki och återanvändning

Klassrelationerna bildar ett träd och kan utnyttjas på flera sätt.

- Direkt återanvändning, submodell i en struktur är en subklass till en biblioteksmodell.
- Specialisering, tillägg i en modell som är en subklass till annan.
- Polymorfism, modeller med identiska interface kan bytas ut oberoende av omgivning i strukturer.
- Organisation, klassrelationen strukturerar modellbibliotek.

Bibliotek och databaser

Biblioteksfiler: innehåller Omolaklasser.

```
LIBRARY <name of library>;
USES <lib1>, <lib2>, ... ;

{model definitions}
```

Bibliotekskataloger: ett "directory" med filer som är Omolaklasser. Flera bibliotekskataloger bildar modelldatabas.

- Bibliotek har lokal namngivning.
- För att nå ett annat bibliotek defineras det i USES listan.
- För att nå en speciell klass skrivs biblioteksnamnet före klassnamnet enligt: `Lib1::Model1`.

Scope rules I

Sökregler efter referens i uttryck:

1. lokalt definerat attribut,
2. ärvt attribut,
3. attribut i ägarklassen.

Sökregler efter superklass:

1. lokat definerad i bibliotek,
2. definition i USES-listan.

Scope rules II

För att styra sökreglerna finns vissa enkla prefix:

this:: söker efter superklass i strukturhierarkin med början i denna klass.

outer:: det samma som `this` men börjar en nivå upp i strukturhierarkin.

super:: söker en nivå upp i klasshierarkin.

Globala variabler kan nås med dubbelkolonn-prefix, ex. `::Pi`

Simulering med OmSim

OH-bilder från Sven Erik.

Numerical Solvers

ODEs

$\dot{x} = f(t, x)$

Provide a routine for $f(t, x)$.

DAEs

$F(t, x, \dot{x}) = 0$

Provide a routine for $F(t, x, \dot{x})$.

DASSL – multi-step method

RADAU5 – implicit Runge-Kutta

DASSL's Approach

A BDF approximation for $\dot{x}$ gives an algebraic equation system to solve.

Example: Backward Euler

$\dot{x}_n \approx (x_n - x_{n-1})/h$

gives

$F(t_n, x_n, (x_n - x_{n-1})/h) = 0.$

Model Compilation

1. Check language syntax and semantics.
2. Check model semantics.
3. Transform connections into equations.
4. Create objects for each variable and equation.
5. Symbolic analysis and manipulation
   - to check for mathematical completeness and consistency
   - to produce a representation suitable for numerical solution

Manipulation of DAE Systems

1. Check for structural defects.
2. Sort in computational order.
3. Calculate constants and perform numeric calculations to reduce expressions
4. Derive and sort the differentiated index one problem. For each subsystem
   (a) reduce index
   (b) manipulate to simplify
5. Decompose into dynamic and output part.
6. Transform to first order.
7. Output a result suitable for simulation.
   - common subexpressions
   - efficient routines for Jacobians

Structural Analysis

- to detect structural singularities
- to decompose the problem into a sequence of problems

"Causality Assignment"

Associate each equation with a variable;
Construct an output set.

Consider $h(a, b, c, d, e, f) = 0$.

Structure Jacobian

	a	b	c	d	e	f
h_1	*	*				
h_2	*	*				
h_3	*	*	*			
h_4	*		*	*		*
h_5				*	*	
h_6	*				*	*

Permute the equations to get a non-zero diagonal.

There are simple, efficient algorithms.

Is the Problem Singular?

- $h(v) = 0$ is structurally singular if there is no output set.
- $F(t, x, \dot{x}) = 0$ is structurally singular if $F(t, z, z) = 0$ has no output set.

Deduce Computational Order

Algebraic loops imply equations systems to solve.

Simple, efficient algorithm by Tarjan gives minimum sized subequation systems.

	a	b	c	d	e	f
h_1	*	*				
h_2	*	*				
h_3	*	*	*			
h_4	*		*	*		*
h_5				*	*	
h_6	*				*	*

Block Lower Triangular partition.

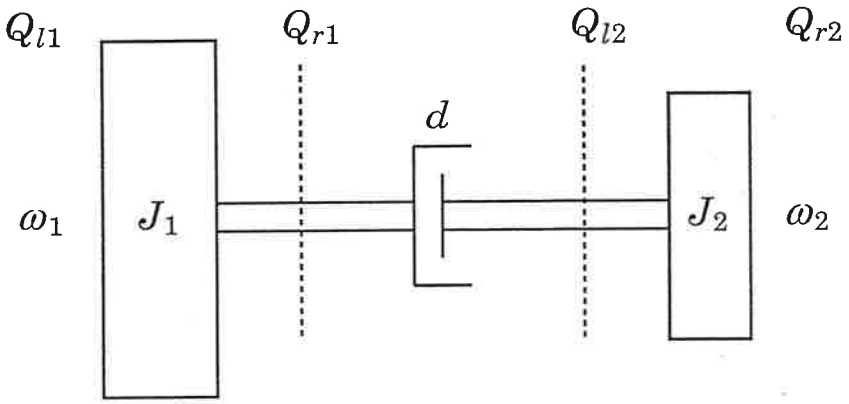
Transformation to ODE

$$g(t,p,x,\dot{x},v) = 0 \rightarrow \begin{bmatrix} \dot{x} \\ v \end{bmatrix} = G(t,p,x)$$

1. Assume t, p and x known.
2. Make a causality assignment.
3. Sort in computational order.
4. Solve the subequation systems.

If the causality assignment fails we need to differentiate equations.

Example: Rotating Masses



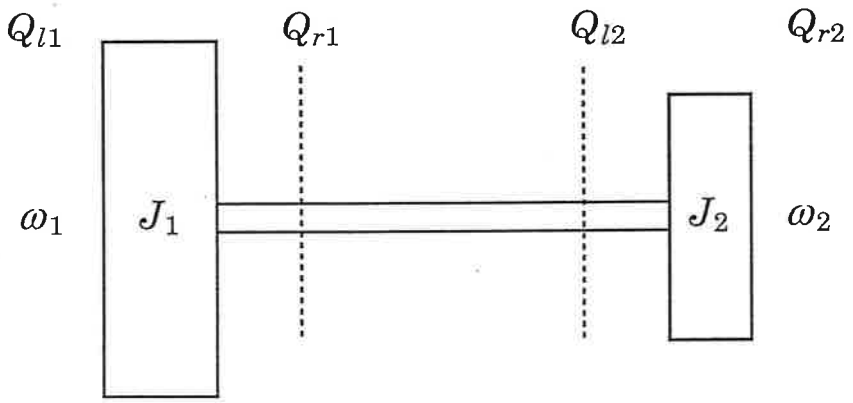
$$J_1\underline{\dot{\omega}_1} = Q_{l1} + Q_{r1}$$
$$J_2\dot{\omega}_2 = Q_{l2} + Q_{r2}$$
$$Q_{r1} = -\underline{Q_{l2}}$$
$$\underline{Q_{r1}} = d(\omega_2 - \omega_1)$$

where Q_{l1} and Q_{r2} are known time functions and J_1, J_2 and d are parameters.

Can solve for Q_{r1}, Q_{r2}, $\dot{\omega}_1$ and $\dot{\omega}_2$.

Straightforward to solve numerically.

Rigidly Connected Masses



$$
\begin{aligned}
J_1\dot{\omega}_1 &= Q_{l1} + Q_{r1} \\
J_2\dot{\omega}_2 &= Q_{l2} + Q_{r2} \\
Q_{r1} &= -Q_{l2} \\
\omega_1 &= \omega_2
\end{aligned}
$$

Well-posed if $J_1 + J_2 \neq 0$; one mass.

Solver must differentiate $\omega_1 = \omega_2$
to get for the reaction torques Q_{r1} and Q_{l2}.

Application: two gear-wheels, $n_1\omega_1 = n_2\omega_2$

DAE Index

Minimum number of times that all or part of $F(t, x, \dot{x}) = 0$ must be differentiated to solve for $\dot{x}$ as a continuous function of t and x.

Examples

The ODE, $\dot{x} = f(t, x)$ is index 0.

The semi-explicit problem

$$
\begin{aligned}
\dot{x} &= f(t, x, y) \\
0 &= g(t, x, y)
\end{aligned}
$$

is index 1 if $\partial g/\partial y$ is nonsingular.

If it is possible to solve for highest appearing derivative of each variable

- the index is 0 or 1
- and no need for differentiations

Numerical Solution of High Index Problems

Today's numerical DAE solvers fail in most cases if they have to differentiate.

They are designed to integrate,
i.e. calculate x from dx/dt
but not to differentiate
i.e. calculate dx/dt from x.

Integration and differentiation require different step length and error controls.

Modeling and the DAE Index

High index implies algebraic relations between dynamic variables.

Examples

- Complex models constructed by combining library models
- Network models
- Discretized PDE models
- Inverse problems

Note: The index is not a problem invariant.

Index Reduction

- Use Pantelides's algorithm to find the differentiated index 1 problem.
- Use the differentiated equations
  - numerical drift-off may show up
  - methods for stabilization
- Symbolic differentiation and elimination
  - not always possible
- Dummy derivatives.
  - keep the original equations
  - use differentiated equations to implicitly define certain derivatives algebraically

Manipulation of DAE Systems

1. Check for structural defects.
2. Sort in computational order.
3. Calculate constants and perform numeric calculations to reduce expressions
4. Derive and sort the differentiated index one problem. For each subsystem
   (a) reduce index
   (b) manipulate to simplify
5. Decompose into dynamic and output part.
6. Transform to first order.
7. Output a result suitable for simulation.
   - common subexpressions
   - efficient routines for Jacobians

Omola-kurs

lektion III

Händelser och diskreta system

Innehåll:

1. Händelse-begreppet
2. Logik och sekvenser
3. Tidsdiskreta system
4. Hybrid-simulering

Händelser

When-händelser sker **om** "condition" är sann.

```
WHEN <condition> DO <action> END;
```

OnEvent-händelser sker **när** "condition" blir sann. (Flank-trigg)

```
OnEvent <condition> DO <action> END;
```

Händelser med namn

Händelser kan också ges namn.

```
E ISAN Event;
OnEvent x > 0.5 CAUSE E;
OnEvent E DO new(x) := 0 END;
```

Händelser kan propageras genom kopplingar i händelseterminal.

```
M ISA Model WITH
  T1 ISA EventInput;
  T2 ISA EventOutput;
  OnEvent T1 CAUSE T2;
END;
```

Villkor

Villkorsbeskrivningar:

- Enkla olikheter
- Sammansatta olikheter med AND och OR
- Händelse med namn

Konsekvenser

Konsekvensbeskrivningar:

- Skapandet av en ny händelse med CAUSE.
- Beräkning av konsekvenser med DO ... END;.
  - Skapa tidsberoende händelse med SCHEDULE(E,1);
  - Beräkning av nya diskreta variabelvärden med new-operatorn.

```
X TYPE DISCRETE Integer;
OnEvent E DO
  new(X) := X + 1;
END;
```

  - Beräkning av nya tillstånd.

```
x ISA Variable;
x' = -x;
OnEvent E DO
  new(x) := 10;
END;
```

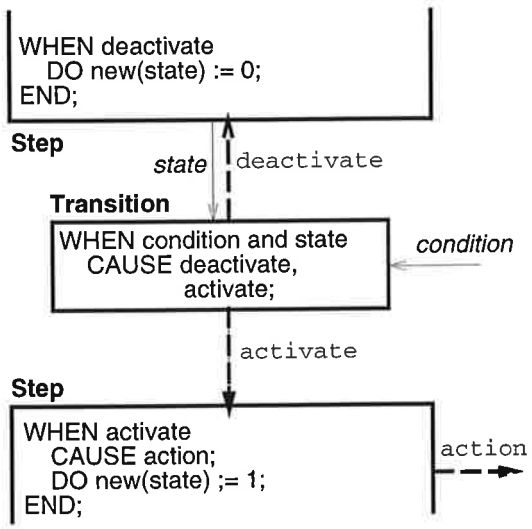
Logik

Exempel

```
AndGrind ISA Model WITH
  In1, In2 ISA SimpleTerminal WITH
    value   TYPE DISCRETE (true,false);
    default TYPE (true,false) := 'false;
    causality := 'input;
  END;
  Out ISA this::In1 WITH
    causality := 'output;
  END;
  WHEN In1=='true AND In2=='true DO
    new(Out) := 'true;
  END;
  WHEN In1=='false OR In2=='false DO
    new(Out) := 'false;
  END;
END;
```

Sekvenser

Grafcet-exempel



Sampling

Tidsdiskret modell:

```
M ISA Model WITH
  h ISA Parameter;
  Init, Sample ISAN Event;
  OnEvent Init, Sample DO
    SCHEDULE(Sample,h);
  END;
END;
```

- Schedule-operatorn lägger in händelsen i en tidskö.
- Modellen genererar en Sample-händelse varje h:te tidsintervall.

Digital PI

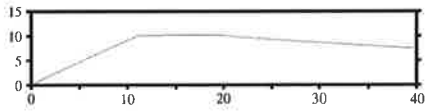
En (mycket) enkel digital PI-regulator:

```
PImodel ISA Model WITH
  yref, y, u    ISA SimpleTerminal;
  K, Ti, h      ISA Parameter;
  Init, Sample ISAN Event;
  e, p, i, v    TYPE DISCRETE Real;
  OnEvent Init, Sample DO
    new(e) := yref - y;
    new(p) := K*e;
    new(i) := i + K*h*e/Ti;
    new(v) := new(p) + i;
    SCHEDULE(Sample,h);
  END;
  u := v;
END;
```

- De interna variablerna är diskreta.
- e och new(e) representerar värdet före och efter händelsen.
- Framåtapproximation av I-del.

Händelser i kontinuerliga modeller

```
TankModel ISA Model WITH
termnals:
  In        ISA SimpleInFlow;
  Out, Out2 ISA SimpleOutFlow;
variables:
  h ISA Variable;
  Over TYPE DISCRETE Integer;
equations:
  h'   = In - Out - Out2;
  Out  = 0.01*SQRT(2*9.81*h);
  Out2 = Over*0.5*SQRT(2*9.81*(h-10));
events:
  Init, OverFlow, Normal ISAN Event;
  OnEvent h>10 CAUSE OverFlow;
  OnEvent h<10 CAUSE Normal;
  OnEvent OverFlow DO new(Over):=1; END;
  OnEvent Normal   DO new(Over):=0; END;
END;
```



Simulering av hybrid-system

1. Kontinuerlig simuleringsmod:
   (a) Integrering av kontinuerliga tillstånd,
   (b) Om någon indikatorfunktion når ett nollställe stoppas integreringen.
2. Diskret simuleringsmod:
   (a) Den händelse som beskrivs av indikatorfunktionen avfyras och alla dess följdhändelser.
   (b) Alla konsekvenser beräknas och därefter startas den kontinuerliga simuleringen igen.

Omola and OmSim Exercise

A introductory exercise in OmSim. The goal is to get familiar with the Omola modeling language and the OmSim simulation environment.

1. Get started

Start and Stop

OmSim can be started by the command `omsim`. This result in the creation of two windows: Omola Class Browser and OmSim log window. Select the `quit` alternative in the *File* menu in the browser to quit OmSim.

Omola Class Brower

There is two libraries in the browser: Scratch and Base. **Select Base!** Base contains predefined Omola classes which are super classes for user defined models and model components.

In the *Display* menu there are choices for examination of classes. Select a class and lock at the code by the `Omola` alternative.

In the *Show Tree ¿¿* on can select other displaying tools for class tree browsing. For example mark **Class** in the Base library and then select Inheritance in the Show Tree submenu.

2. Primitive models

This part describe the development and simulation of a simple primitive model. It is the two tank process form the first course in Control Engineering.

Model Development of a primitive model

Describe the simple two tank process. To create a new primitive model inside OmSim one have to do the following steps.

1. Open a MED editor in the *Tools* menu.
2. Mark `Model` in the Base library.
3. Select `New` in the *Edit* menu. This creates a new class called `Unnamed` which is a subclass of `Model`.
4. Push a button on the class name in MED and select *Info...* in the pop-up menu. Change the name of the model and the storage file name. Push set name button.*(In our case use the name TwoTankModel.)*
5. Remove the MED editor by making selecting `Cancel` in the *File* menu (in MED).
6. You will find the new class in the Scratch library. Mark the model and select `Emacs` in the Tools menu and the model appear in your text editor.
7. Enter your model interior, variable and parameter declarations and equation expressions. See the listing below. Send back the model to OmSim by `C-x #`.

This is a Omola model of the simple two tank system.

```
TwoTankModel ISA Model WITH
  qmax          ISA Parameter WITH default:= 2.7e-6; END;
  Area1, Area2 ISA Parameter WITH default:= 2.7e-3; END;
  a1, a2        ISA Parameter WITH default:= 7e-6; END;
```

```
  g             ISA Parameter WITH value:=9.81; END;
  u             ISA Parameter;
  u.default := 5;
  h1, h2 ISA Variable;
  h1' = ( qmax*u          - a1*SQRT(2*g*h1) )/Area1;
  h2' = ( a1*SQRT(2*g*h1) - a2*SQRT(2*g*h2) )/Area2;
END;
```

Simulation of a primitive model.

To simulate a model and plot the result one have to do the following steps.

1. Mark the model that you want to simulate and select `Simulate` in the *Tools* menu. This creates a Simulator window. `Instantiating... Done!` appear in the log window if the model is OK. Error message will appear here otherwise. *(You remove the simulator by selecting `Cancel` in the Config menu (in Simulator).)*
2. Create an Access window, for instance States. In our case there are two states: `h1` and `h2`.
3. Create a plot window in the *In/Out* menu. Connect both the states to the plot window by pushing the mouse button on a variable in an access window and make selection in the pop-up menu.
4. Enter proper simulation time in the Simulator window and start the simulation by pushing the `Start` button. *In our case enter 300 in stop time and start. After the simulation push `Rescale`.*
5. Make a change of the `u` parameter in a Parameter Access window. Enter 3 for `u` and push the `Enter` button.
6. Push the `Start` button in the Simulator again and see the result in the plot window.

3. Composite Models

This third part describe the development and simulation of composite models. It also discusses the use of OCL.

Model Library

A pure textual description of a composite model is found in file `tanklib.om`. This is the Omola version of the Simnon example found in the CCS book by Åström and Wittenmark. The Simnon version and the corresponding Omola version are found in the last section in this report.

Look at the library file, `tanklib.om`, in our editor. The file begins with a library head giving the library a name, `TankLib`. Load the library into OmSim by selecting *Load...* in the *File* menu. Mark the library file in the subwindow.

The library contains three models: TankModel, DpiModel and RegTank. TankModel is a description of a tank with one inflow and one outflow. DpiModel is a digital PI controller. RegTank is a composite model with one tank submodel and one digital PI controller submodel. The tank and the controller are connected using the AT operator on terminals. One example is `Reg.y AT Tank.h`. This indicates that the measurement signal in Reg are the same as the height in the Tank.

Simulation Command Language

Simulations can be driven by a command language called OCL. This corresponds to MACROs in Simnon. On example of an OCL for simulation of the RegTank model is listed below.

```
BEGIN
  TankLib::RegTank m;
  SIMULATOR TankSim(m);
  TankSim.stoptime := 200;
  TankSim.display;
  PLOTTER Level(m);
  Level.y(href,Reg.y);
  PLOTTER Flow(m);
  Flow.y(Tank.qin,Reg.u);
  TankSim.start;
  Level.rescale;
  Flow.rescale;
END;
```

In the first statement a model is given an internal short name, `m`. Then a simulator is created called `TankSim` and given a stop time and a display window. Two plot windows are also created for showing the level and the flow. The simulation is started and the plot windows are rescaled.

An OCL is executed when the file is loaded into OmSim. Select *Load...* in the *File* menu and mark the right OCL file (tanksim1.ocl). As a user you have to interactively place the windows on the screen. Add following to the OCL to automatically place the windows on the screen.

```
  TankSim.display(400,800);
  Level.display(400,500);
  Flow.display(700,500);
```

4. Composite Models with Graphics

The describe composite models it is often desired to make a graphical description illustrating the structure of connected submodels.

Composite Model Development

Until now everything have been without graphics. Graphic information is added to models in a `Graphic` attribute. All models and terminals automatically get a graphic attribute. Some parts of the graphic information is automatically generated and some must be added by the user.

All classes the have a graphical representation must have an attribute which means models, submodels, terminals and connections. The graphical information is added in the Graphic attribute as:

```
LayOut ISA Class WITH
  x_pos TYPE Real;
  y_pos TYPE Real;
  x_size TYPE Real;
  y_size TYPE Real;
  invisible TYPE Integer := 0;
  bitmap TYPE String := "filename";
```

```
END;

Graphic ISA LayOut;
```

To enter graphical information do as follows:

1. Mark the model that you want to edit and select a MED editor. (ex: TankModel)
2. TankModel is already defined without graphic information. All undefined classes are therefore found bottom left.
3. Push a mouse bottom on a terminal class at bottom left and select `Move` in the pop-up menu. Drag the terminal to a desired location. Do the same thing with the other terminals in `TankModel` and in `DpiModel`.
4. Mark `RegTank` in the browser and select `Existing` in *File* in MED. Drag the two submodels, Tank and Reg, to desired locations.
5. Delete the connections by the pop-up menu. Define new ones by selecting `Connect` and push mouse buttons on corresponding terminals.
6. Save the library file by selecting *Save...* in the *File* menu

The models get the default icon which is a block. Special designed icons can be generated in a bitmap editor.

A tank library with graphics added is found in `tanklibgr.om` Mark `RegTank` and select a MED tool. Load the library into your editor and note the following things:

- The Graphic attribute of `TankModel` has a bitmap reference to `icontank`.
- The terminals in `TankModel` has graphic attributes with graphic positions.
- The controller has the corresponding graphical information. The terminals has `invisible` set to 1 which means that they are not seen on the icon level.
- The submodels in `RegTank` have graphical position information. The graphical information of a connection is represented as a matrix.

Often it is not important to know this. To develop a graphical representation of the model one have to know about all these graphical details. The MED editor has certain limitations that makes it more easy to develop the graphics using textual representation.

5. Simulation Problems

Simulation of stiff ODE and DAE is illustrated in this section and the numerical solvers are discussed.

Continuous systems

In the Omola library `TankLib2`, which is similar to the previous library, there is a continuous PI controller. The continuous PI is connected to the tank in the `RegTank2` model.

Select the model and simulate it (0 - 100).

In the *Config* menu select the *Options...* alternative. This result in a subwindow. Here it is possible to enter a window name, change control parameters to the solver, select solver, seed the random number generator. Select an ODE method an compare the previous result. Default solver is DASRT which is a DAE solver.

Stiff problems

In the TankModel3 extra valve dynamics is add. The valve is supposed to be a first order system with a time constant.

```
valve' = 1/Tv*(valvepos - valve);
```

Let us compare one explicit and one implicit solver with each other. Do the following:

1. Select the new composite model RegTank3.
2. Select New Model in the *Config* menu and make a new connection of the tank height the plot window.
3. Create a new simulator with a tank height plot window.
4. Control that one simulator has an explicit solver and that the other one has an implicit.
5. Make two simulations of the tank system with the same parameter settings.
6. Change the time constant in the valve dynamics to $Tv = 0.01$ and make two new simulations.

We notice that the implicit solver is independent of the choice of Tv and that the explicit solver becomes very slow when Tv decrease. As Cleve Moler puts it *"Stiffness is painful to watch"*.

Algebraic problems

If the valve dynamics is chosen to be very fast it is tempting to choose $Tv = 0$. The previous model will complain because it appear in the denominator and it results in division by zero.

This can be taken care of by rewriting the equation on the following form.

```
Tv*valve' = valvpos - valve;
```

If Tv is selected to be zero the left hand side is zero and the differential equation becomes an algebraic equation. This is done in TankModel4 and can be simulated by RegTank4.

Simulate RegTank4 with different values on Tv including zero. Notice that the performance is almost independent of the chose of Tv.

DAE with index problems

If Tv is selected to be zero the system can seen as $B\dot{x} = f(x, u)$ where B is constant but not invertible. A system with this property is called a DAE with index one.

Systems can have higher index. One example is to add an extra tank. In normal cases we have to add the flow description between the tanks. If the flow is very large then it is tempting to assume that the two tank heights are the same.

```
h' = (qin - qout - qtank)/area;
area2 * h2' = qtank;
h2 = h;
```

qtank is the flow between the two tanks. The second tank don't have any inlet or outlet and finally are the heights set equal.

Two solve this system the solver have to differentiate, $h2 = h$, once to find that the two differential equations can be set equal. This result in a calculation of qtank which must be known before the integration of h2. When h2 is known h is known.

In TankModel5 and RegTank5 you will find these changes in the model. Simulate the DAE index-2 problem. Note the number of states.

6. CCS tank example

CCS Simnon example

```
CONTINUOUS SYSTEM tank
  INPUT v aout
  OUTPUT h
  DER dh
  valve = IF v<0 THEN 0 ELSE IF v>1 THEN 1 ELSE v
  qin = qmqx*valve
  qout = aout*SQRT(2*g*MAX(h,0))
  dh = (qin-qout)/area
  qmax : 1
  g : 9.81
  area : 10
END
DISCRETE SYSTEM dpi
  INPUT y yref
  OUTPUT u
  STATE inte
  NEW ninte
  TIME t
  TSAMP ts
  e = yref - y
  i1 = inte+k*e*h/ti
  u = k*e+i1
  ninite = i1
  ts = t+h
  k : 1
  ti : 1000
  h : 1
END
CONNECTING SYSTEM regtank
  TIME t
  aout[tank] = IF t<100 THEN a1 ELSE a2
  a1 : 0.01
  a2 : 0.05
  yref[dpi] = href
  href : 2
  y[dpi] = h[tank]
  v[tank] = u[dpi]
END
MACRO tanksim
  SYST tank dpi regtank
  STORE h[tank] href qin
  SIMU 0 200
  SPLIT 2 1
  SHOW h[tank] href
  AXES
  SHOW qin
END
```

CCS compatible Omola example

```
TankModel ISA Model WITH
  v, aout, h ISA SimpleTerminal;
  qmax ISA Parameter WITH default:=1;  END;
  area ISA Parameter WITH default:=10; END;
  g    ISA Parameter WITH value:=1;    END;
  valve, qin, qout TYPE Real;
  valve = IF v<0 THEN 0 ELSE IF v>1 THEN 1 ELSE v;
  qin   = qmax*valve;
  qout  = aout*SQRT(2*g*MAX(h,0));
  h'    = (qin-qout)/area;
END;
DpiModel ISA Model WITH
  y, yref, u ISA SimpleTerminal;
  k  ISA Parameter WITH default:=1;    END;
  ti ISA Parameter WITH default:=1000; END;
  h  ISA Parameter WITH default:=1;    END;
  inte, e, v TYPE DISCRETE Real;
  Init, Sample ISAN Event;
  ONEVENT Init OR Sample DO
    new(e)    := yref - y;
    new(inte) := inte+k*e*h/ti;
    new(v)    := k*new(e)+inte;
    SCHEDULE(Sample,h);
  END;
  u := v;
END;
RegTank ISA Model WITH
  a1   ISA Parameter WITH default:=0.01; END;
  a2   ISA Parameter WITH default:=0.05; END;
  href ISA Parameter WITH default:=2;    END;
  Tank ISA TankModel;
  Reg ISA DpiModel;
  Tank.aout = IF t<100 THEN a1 ELSE a2;
  Reg.yref  = href;
  Reg.y AT Tank.h;
  Reg.u AT Tank.v;
END;
BEGIN
  ccstank::RegTank m;
  SIMULATOR s(m);
  s.stoptime:=200;
  s.display;
  PLOTTER p1(m);
  p1.y(href,Reg.y);
  PLOTTER p2(m);
  p2.y(Tank.qin,Reg.u);
  s.start;
END;
```

```
LIBRARY TankLib;
%%
%% A Omola library for a tank example
%% found in Simnon tutorials and in
%% the CCS book by Astrom and Wittenmark.

TankModel ISA Model WITH
  v, aout, h ISA SimpleTerminal;
  qmax ISA Parameter WITH default:=1;    END;
  area ISA Parameter WITH default:=10;   END;
  g    ISA Parameter WITH default:=9.81; END;
  valve, qin, qout ISA Variable;
  valve = IF v<0 THEN 0 ELSE IF v>1 THEN 1 ELSE v;
  qin   = qmax*valve;
  qout  = aout*SQRT(2*g*MAX(h,0));
  h'    = (qin - qout)/area;
END;

DpiModel ISA Model WITH
  y, yref, u ISA SimpleTerminal;
  k  ISA Parameter WITH default:=1;    END;
  ti ISA Parameter WITH default:=1000; END;
  h  ISA Parameter WITH default:=1;    END;
  inte, e, v TYPE DISCRETE REAL;
  Init, Sample ISAN Event;
  ONEVENT Init OR Sample DO
    new(e) := yref - y;
    new(inte) := inte + k*e*h/ti;
    new(v) := k*new(e) + inte;
    SCHEDULE(Sample,h);
  END;
  u := v;
END;

RegTank ISA Model WITH
  a1   ISA Parameter WITH default:=0.01; END;
  a2   ISA Parameter WITH default:=0.05; END;
  href ISA Parameter WITH default:=2;    END;
  Tank ISA TankModel;
  Reg ISA DpiModel;
  Tank.aout = IF Base::Time<100 THEN a1 ELSE a2;
  Reg.yref := href;
  Reg.y AT Tank.h;
  Reg.u AT Tank.v;
END;

%-----------------------------------------------------------------
```

```
LIBRARY TankLib2;
%%
%% A Omola library for a tank example
%% found in Simnon tutorials and in
%% the CCS book by Åström and Wittenmark.

TankModel ISA Model WITH
  v, aout, h ISA SimpleTerminal;
  qmax ISA Parameter WITH default:=1;    END;
  area ISA Parameter WITH default:=10;   END;
  g    ISA Parameter WITH default:=9.81; END;
  valve, qin, qout ISA Variable;
  valve = IF v<0 THEN 0 ELSE IF v>1 THEN 1 ELSE v;
  qin   = qmax*valve;
  qout  = aout*SQRT(2*g*MAX(h,0));
  h'    = (qin - qout)/area;
END;

DpiModel ISA Model WITH
  y, yref, u ISA SimpleTerminal;
  k  ISA Parameter WITH default:=1;    END;
  ti ISA Parameter WITH default:=1000; END;
  h  ISA Parameter WITH default:=1;    END;
  inte, e, v TYPE DISCRETE REAL;
  Init, Sample ISAN Event;
  ONEVENT Init OR Sample DO
    new(e) := yref - y;
    new(inte) := inte + k*e*h/ti;
    new(v) := k*new(e) + inte;
    SCHEDULE(Sample,h);
  END;
  u := v;
END;

RegTank ISA Model WITH
  a1   ISA Parameter WITH default:=0.01; END;
  a2   ISA Parameter WITH default:=0.05; END;
  href ISA Parameter WITH default:=2;    END;
  Tank ISA TankModel;
  Reg ISA DpiModel;
  Tank.aout = IF Base::Time<100 THEN a1 ELSE a2;
  Reg.yref := href;
  Reg.y AT Tank.h;
  Reg.u AT Tank.v;
END;

%-------------------------------------------------------------------

PiModel ISA Model WITH
  y, yref, u ISA SimpleTerminal;
  k  ISA Parameter WITH default:=1;    END;
  ti ISA Parameter WITH default:=1000; END;
  inte, e TYPE REAL;
  e     = yref - y;
  inte' = e*k/ti;
  u = k*e + inte;
END;

RegTank2 ISA RegTank WITH
  Reg ISA PiModel;
END;

%-------------------------------------------------------------------

TankModel3 ISA Model WITH
  v, aout, h ISA SimpleTerminal;
  qmax, Tv ISA Parameter WITH default:=1;    END;
  area ISA Parameter WITH default:=10;   END;
  g    ISA Parameter WITH default:=9.81; END;
  valve, valvepos, qin, qout ISA Variable;
  valvepos = IF v<0 THEN 0 ELSE IF v>1 THEN 1 ELSE v;
  valve' = 1/Tv*(valvepos - valve);
  qin   = qmax*valve;
  qout  = aout*SQRT(2*g*MAX(h,0));
  h'    = (qin - qout)/area;
END;

RegTank3 ISA RegTank2 WITH
  Tank ISA TankModel3;
END;

%-------------------------------------------------------------------

TankModel4 ISA Model WITH
  v, aout, h ISA SimpleTerminal;
  qmax ISA Parameter WITH default:=1;    END;
  Tv   ISA Parameter WITH default:=0;    END;
  area ISA Parameter WITH default:=10;   END;
  g    ISA Parameter WITH default:=9.81; END;
  valve, valvepos, qin, qout ISA Variable;
  valvepos = IF v<0 THEN 0 ELSE IF v>1 THEN 1 ELSE v;
  Tv*valve' = (valvepos - valve);
  qin   = qmax*valve;
  qout  = aout*SQRT(2*g*MAX(h,0));
  h'    = (qin - qout)/area;
END;

RegTank4 ISA RegTank2 WITH
  Tank ISA TankModel4;
END;

%-------------------------------------------------------------------

TankModel5 ISA Model WITH
  v, aout, h, h2 ISA SimpleTerminal;
  qmax ISA Parameter WITH default:=1;    END;
  Tv   ISA Parameter WITH default:=0;    END;
  area, area2 ISA Parameter WITH default:=10;   END;
  g    ISA Parameter WITH default:=9.81; END;
  valve, valvepos, qin, qout, qtank ISA Variable;
  valvepos = IF v<0 THEN 0 ELSE IF v>1 THEN 1 ELSE v;
  Tv*valve' = (valvepos - valve);
  qin  = qmax*valve;
  qout = aout*SQRT(2*g*MAX(h,0));
  h'  = (qin - qout - qtank)/area;
  area2*h2'  = qtank;
  h   = h2;
END;

RegTank5 ISA RegTank2 WITH
  Tank ISA TankModel5;
END;
```

```
LIBRARY TankLibGr;
%%
%% A Omola library for a tank example
%% found in Simnon tutorials and in
%% the CCS book by Astrom and Wittenmark.

TankModel ISA Model WITH
icon:
  Graphic ISA super::Graphic WITH bitmap TYPE String := "icontank"; END;
terminals:
  v ISA SimpleTerminal WITH
    Graphic ISA super::Graphic WITH
      x_pos := 0.0;
      y_pos := 150.0;
    END;
  END;
  aout ISA SimpleTerminal WITH
    Graphic ISA super::Graphic WITH
      x_pos := 200.0;
      y_pos := 300.0;
    END;
  END;
  h ISA SimpleTerminal WITH
    Graphic ISA super::Graphic WITH
      x_pos := 400.0;
      y_pos := 150.0;
    END;
  END;
parameters:
  qmax ISA Parameter WITH default := 1; END;
  area ISA Parameter WITH default := 10; END;
  g ISA Parameter WITH default := 9.81; END;
variables:
  valve ISA Variable;
  qin ISA Variable;
  qout ISA Variable;
equations:
  valve = (if v < 0 then 0 else (if v > 1 then 1 else v));
  qin = qmax*valve;
  qout = aout*sqrt(2*g*max(h, 0));
  h' = (qin - qout)/area;
END;

DpiModel ISA Model WITH
icon:
  Graphic ISA super::Graphic WITH bitmap TYPE String := "iconpireg"; END;
terminals:
  y ISA SimpleTerminal WITH
    Graphic ISA super::Graphic WITH
      x_pos := 0.0;
      y_pos := 100.0;
      invisible := 1;
    END;
  END;
  yref ISA SimpleTerminal WITH
    Graphic ISA super::Graphic WITH
      x_pos := 0.0;
      y_pos := 225.0;
      invisible := 1;
    END;
  END;
  u ISA SimpleTerminal WITH
    Graphic ISA super::Graphic WITH
      x_pos := 400.0;
      y_pos := 150.0;
      invisible := 1;
    END;
  END;
parameters:
  k ISA Parameter WITH default := 1; END;
  ti ISA Parameter WITH default := 1000; END;
  h ISA Parameter WITH default := 1; END;
variables:
  inte TYPE DISCRETE Real;
  e TYPE DISCRETE Real;
  v TYPE DISCRETE Real;
events:
  Init ISA Event;
  Sample ISA Event;
discrete_behaviour:
  ONEVENT Init or Sample DO
    new(e) := yref - y;
    new(inte) := inte + k*e*h/ti;
    new(v) := k*new(e) + inte;
    schedule(Sample, h);
  END;
  u := v;
END;

RegTank ISA Model WITH
icon:
  Graphic ISA super::Graphic;
parameters:
  a1 ISA Parameter WITH default := 0.01; END;
  a2 ISA Parameter WITH default := 0.05; END;
  href ISA Parameter WITH default := 2; END;
submodels:
  Tank ISA TankModel WITH
    Graphic ISA super::Graphic WITH
      x_pos := 300.0;
      y_pos := 150.0;
    END;
  END;
  Reg ISA DpiModel WITH
    Graphic ISA super::Graphic WITH
      x_pos := 125.0;
      y_pos := 150.0;
    END;
  END;
assignment:
  Reg.yref := href;
  Tank.aout = IF Base::Time<100 THEN a1 ELSE a2;
connections:
  C1 ISA Base::Connection WITH
    Tank.h AT Reg.y;
    bpoints TYPE STATIC Matrix[6, 2] :=
      [349.0, 149.0; 377.0, 149.0; 377.0, 82.0;
       47.0, 82.0; 47.0, 137.0; 75.0, 137.0];
  END;
  C2 ISA Base::Connection WITH
    Reg.u AT Tank.v;
    bpoints TYPE STATIC Matrix[2, 2] := [137.0, 150.0; 275.0, 150.0];
  END;
END;

%-------------------------------------------------------------------
```

K2 - en modelldatabas för kraftsystem.

Innehåll:

1. organisation
2. biblioteksbeskrivningar

K2 - organisationsprinciper

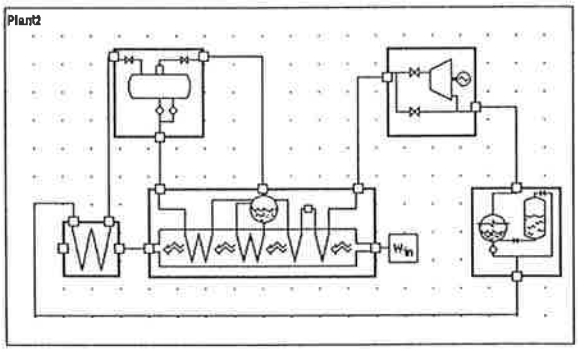
granularitet –
system, enhet, subenhet.

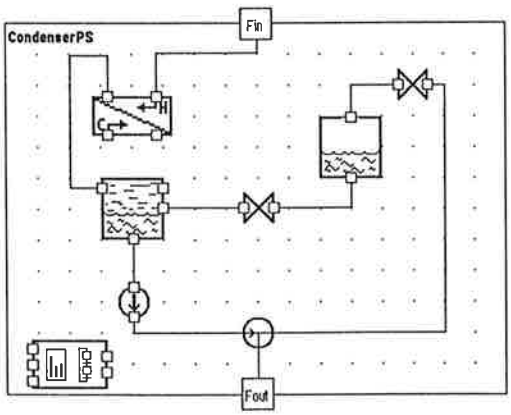
konceptualitet –
compartment, media, flödesresistor.

gränssnitt –
`CompartmentIC`, `Compartment2IC`.

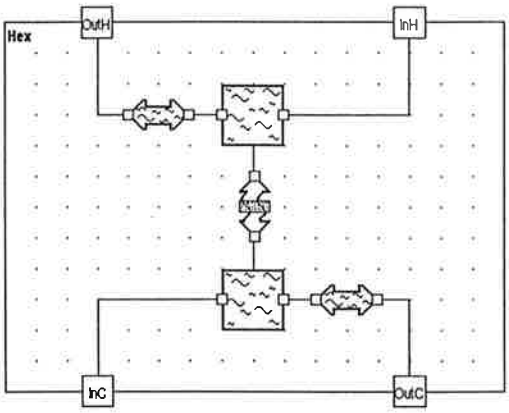
modellklass –
`WaterCompartmentFM`, `GasCompartmentFM`.

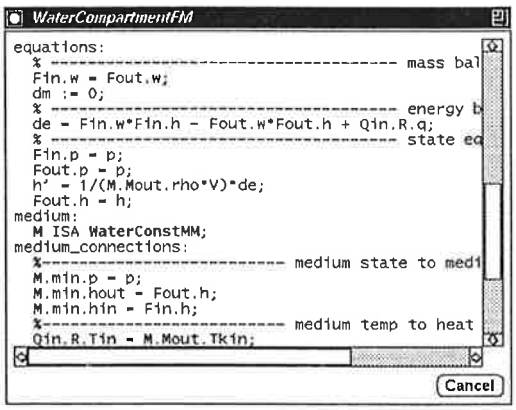
Systemnivå - flowsheet





Enhets- och subenhetsnivå





K2 - bibliotek

Biblioteksuppdelningen följer i stort konceptualitets-uppdelningen.

- Bibliotek av subenheter.
  `K2MediumLib`, `K2CompartmentLib`, `K2FlowLib`, `K2HeatFlowLib`
- Bibliotek av enheter
  `K2FlowUnitLib`, `K2HeatUnitLib`, `K2TurbineLib`
- Dessutom några bibliotek för klassträd, terminaler och funktioner.
  `K2ClassTreeLib`, `K2TerminalLib`, `K2BasicLib`

K2MediumLib - mediamodeller

Mediamodeller används som ett strukturerat sätt att få tillgång till mediaberoende storheter och variabler. De använder ångtabeller och andra funktionsapproximationer.

Gränssnittet består av tre `RecordTerminal`:

Min – `p, hin, hout`

Mout – `rho, Tkin, Tkout, ap, ah`

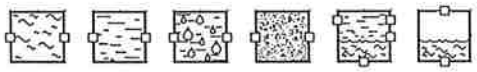
Mout2 – `hw, hs, rhow, rhos, alpha`

Dessutom finns det särskilda flödesmediamodeller som har ett lite annorlunda gränssnitt.

Min – `p, h`

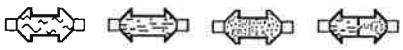
Mout – `rho, Cp, kappa, my, lambda`

K2CompartmentLib - volymsmodeller



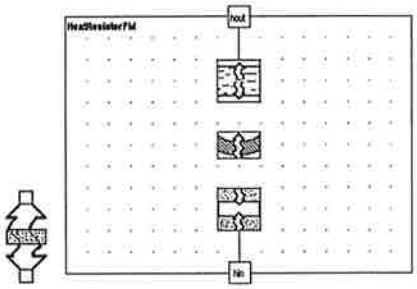
- entalpi- och tryckdynamik via mass- och energibalans
- vatten antas inkompressibelt
- drum med utbildad vätskenivå
- tank med konstant lufttryck, volymsdynamik ersätter tryck

K2FlowLib - flödesmodeller



- statiska flödesbeskrivningar
- inkompressibelt för små tryckfall
- kritisk beskr. för stora tryckfall
- förlustfaktorer
  - konstant i ventiler etc.
  - laminärt, $\sim$Re
  - turbulent, rörgrovhet

K2HeatFlowLib - värmeöverföring



- logaritmisk medeltemperatur
- konvektivt och konduktivt värmemotstånd
  - mediaberoende, Re Pr Nu
  - cylindrisk geometri

K2FlowUnitLib

`WaterPump` – effektstyrd

`WaterValve` – inkompr, konstant förlustfaktor

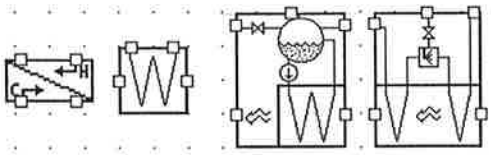
`CritValve` – isentropiskt/kritiskt flöde

`FlowSplit` – dela i två flöden

`FlowJunction` – mix av flöden m olika entalpi

`SprayTemp` – mix av olika faser

K2HeatUnitLib



`HeatExchanger` – enkel vvx, vatten - vatten

`Economizer` – vatten - gas

`Superheater` – ånga - gas

`Boiler` – tvungen strömning, nivåreglering

`SuperHeaterSystem` – dubbel vvx med tempreglering

K2TerminalLib

Terminaler som beskriver media- och värmeflöde.

`FlowInTC` – `w, p, h, M`
där `M` är en media beroende terminal.

`HeatTransferInTC` –
`R.q, Tin, Tout`
`M.p, w, Gmix`

K2BasicLib

Funktioner och globala konstanter.

LogMean – logaritmiska medeltemperaturen

Th – polynomapprox. av avgastemperatur.

konstanter – pi, g, R, M, T0, p0 m.fl

K2EndTerminalLib

Terminaler som binder 'lösa' terminaler. De namnges på följande sätt:

Media – Water, Steam, Gas etc.

Anslutning – Comp, Flow beroende på vilken typ av subenhet terminalen ska anslutas till.

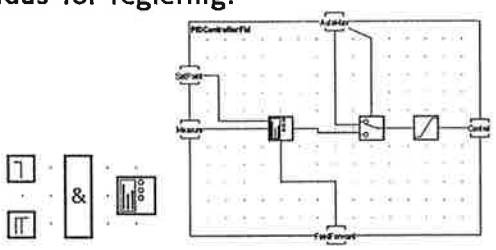
Variabel – W, P om det är flöde eller tryck som ska bindas.

Riktning – In, Out vid in-terminaler måste också entalpin ges ett värde.

Exempel – WaterCompWin, SteamFlowPout

ControlSystemLib

I reglerdatabasen finns enheter som kan användas för reglering.



Logiska operatorer

Less – mindre än ref-värde.

GreaterHyst – större än ref-värde med hysteres.

And – Logiskt OCH av två digitala signaler.

PID-regulatorer

PIDcontroller – PID med analog algoritm, manuell mod, framkoppling och begränsare.

K2 Exercise

An introductory exercise using the K2 library.

1. Get started

Starting OmSim with K2

The K2 model library is stored as an Omola database and can be used by setting the environment variable **OMSIM_DBPATH** to point at the library nodes containing the k2db, k2appdb and controldb model data bases. Check this by issuing the command **printenv OMSIM_DBPATH**. When you start OmSim you also need to tell the program which of the libraries in the databases you want to use. Do this with the command **omsim k2lib.om**.

When the browser appears it contains all the libraries that can be used. By clicking at a library you can also see the classes it contains.

To look around in the HRSG model, choose `K2Plants` and the class `Plant2`. Then open a Model Editor by choosing **MED** in the *Tools* menu. You can now examine the inside of the model by pointing at an object and then choosing DISPLAY-¿MED in the mouse menu.

2. Composite models of predefined classes

This part describe the development and simulation of a simple composite model. It is the two tank process from the first course in Control Engineering.

Model development of a composite model

Describe the simple two tank process. To create a new composite model inside OmSim you do the following steps.

1. Open a Model editor in the *Tools* menu.
2. Mark `Model` in the `Base` library.
3. Select **New** in the *Edit* menu. This creates a new class called `Unnamed` which is a subclass of `Model`.
4. Push the left button on the class name in MED and select *Info...* in the pop-up menu. Change the name of the model. Push **Set name**. *(In our case use the name TwoTanks.)*

   Now we want to insert the subunits into the model, i.e. compartments and flow modules.
5. Mark `OpenCompartmentFM` in `K2CompartmentLib`.
6. Click **Insert** and place a tank in the workspace twice.
7. Mark `WaterFlowResistorFM` in `K2FlowLib`.
8. Click **Insert** and place the flow modules at the outflow of the two tanks.
9. You can change the names of the submodels to shorter and simpler ones by clicking on the icons and selecting *Info...* in the pop-up menu just like before.

   We also need to 'tie up the loose ends', add endterminals, and make all the connections.
10. Mark `WaterCompWinTC` in `K2EndTerminalLib`.

11. Click **Insert** and place the terminal above the first tank.
12. Mark `WaterFlowPoutTC` in `K2EndTerminalLib`.
13. Click **Insert** and place the terminal at the outflow of the second flow module.
14. Use **Connect** to make all connections in the model.
15. Remove the MED editor by selecting **Cancel** in the *File* menu (in MED).
16. You will find the new class in the Scratch library. Save it by choosing **Save file..** in the *File* menu and clicking on the filename.

The current version of OmSim saves new models in library files only. To make it a database file and enter parameters to your model we must exit OmSim and edit the file in Emacs.

1. Open the file in Emacs by pressing **C-x C-f** and typing the filename.
2. Remove the first lines with the `LIBRARY` statement.
3. Enter your parameter declarations and expressions. See the listing below. Please note the altered lines within the automatically generated code. Save the model under the new name with **C-x C-w**. The name of the file must be the same as the class name.

This is a listing of the Omola model of the simple two tank system.

```
TwoTanks ISA Base::Model WITH
  Graphic ISA super::Graphic;
parameters:
  wmax ISA Base::Parameter WITH default := 0.0135;END;
  Area ISA Base::Parameter WITH default := 0.0027; END;
  a ISA Base::Parameter WITH default := 7e-6; END;
  l ISA Base::Parameter WITH default := 0.24; END;
  hin ISA Base::Parameter WITH default := 200000; END;
  u ISA Base::Parameter;
submodels:
  Tank1 ISA K2CompartmentLib::OpenCompartmentFM WITH
    Graphic ISA super::Graphic WITH
      x_pos := 151.0;
      y_pos := 201.0;
    END;
    A := outer::Area; % altered
  END;
  Tank2 ISA this::Tank1 WITH % altered
    Graphic ISA super::Graphic WITH
      x_pos := 250.0;
      y_pos := 100.0;
    END;
  END;
  Flow1 ISA K2FlowLib::WaterFlowResistorFM WITH
    Graphic ISA super::Graphic WITH
      x_pos := 201.0;
      y_pos := 151.0;
    END;
    diameter := 2*sqrt(outer::a/K2BasicLib::pi);% altered
    length := outer::l; % altered
  END;
  Flow2 ISA this::Flow1 WITH % altered
    Graphic ISA super::Graphic WITH
```

```
      x_pos := 326.0;
      y_pos := 51.0;
    END;
  END;
terminals:
  T1 ISA K2EndTerminalLib::WaterCompWinTC WITH
    Graphic ISA super::Graphic WITH
      x_pos := 151.0;
      y_pos := 276.0;
    END;
  END;
  T2 ISA K2EndTerminalLib::WaterFlowPoutTC WITH
    Graphic ISA super::Graphic WITH
      x_pos := 376.0;
      y_pos := 51.0;
    END;
  END;
connections:
  T1.wRef := u*wmax; % altered
  T1.href := hin; % altered
  T2.pref := K2BasicLib::p0; % altered
  C1 ISA Base::Connection WITH
    T1 AT Tank1.Fin;
    bpoints TYPE STATIC Matrix[2, 2] := [150.0, 275.0; 150.0, 224.0];
  END;
  C2 ISA Base::Connection WITH
    Tank1.Fout AT Flow1.Fin;
    bpoints TYPE STATIC Matrix[3, 2] := [150.0, 175.0; 150.0, 150.0; 175.0, 150.0];
  END;
  C3 ISA Base::Connection WITH
    Flow1.Fout AT Tank2.Fin;
    bpoints TYPE STATIC Matrix[3, 2] := [224.0, 150.0; 249.0, 150.0; 249.0, 124.0];
  END;
  C4 ISA Base::Connection WITH
    Tank2.Fout AT Flow2.Fin;
    bpoints TYPE STATIC Matrix[3, 2] := [249.0, 75.0; 249.0, 50.0; 300.0, 50.0];
  END;
  C5 ISA Base::Connection WITH
    Flow2.Fout AT T2;
    bpoints TYPE STATIC Matrix[2, 2] := [350.0, 50.0; 375.0, 50.0];
  END;
END;
```

Simulation of a composite model.

To simulate a model and plot the result you have to do the following.

1. Mark the model that you want to simulate and select `Simulate` in the *Tools* menu. `Instantiating... Done!` appear in the log window if the model is OK. Error message will appear here otherwise. *(You remove the simulator by selecting `Cancel` in the Config menu (in Simulator).)*
2. Create the Access windows, both **All** and **States**. In our case there will be six states: enthalpy and volume of the two tanks together with the flow and

pressure of the endterminals.

3. Create a plot window in the *In/Out* menu. Connect the levels of the tanks to the plot window by pushing the mouse button on a variable in an access window and selecting **Connect** in the pop-up menu. You can create an access with just the levels by using **Search** in the *Access* menu.
4. Enter the value 1 for the parameter u in the Access window.
5. This version of the tank model can not handle zero volume. Enter initial values of the states in the States window. The volume in each tank could be 1e-6, the enthalpy 200000, the inflow 0.0135 and the pressure out should be 101325.
6. Enter proper simulation time in the Simulator window and start the simulation by pushing the `Start` button. *In our case enter 500 in stop time and start. After the simulation push `Rescale`.*
7. Make a change of the u parameter in a Parameter Access window. Enter 0.6 for u and push the `Enter` button.
8. Push the `Start` button in the Simulator again and see the result in the plot window.