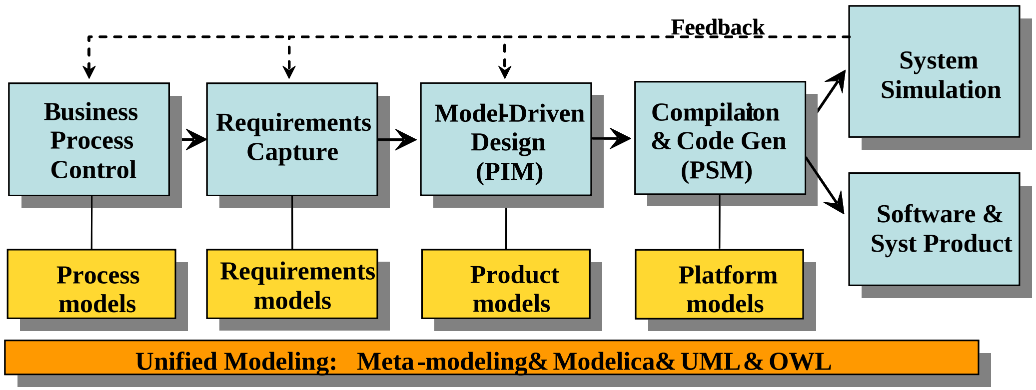
Deliverable of OpenProd task 6.14

ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

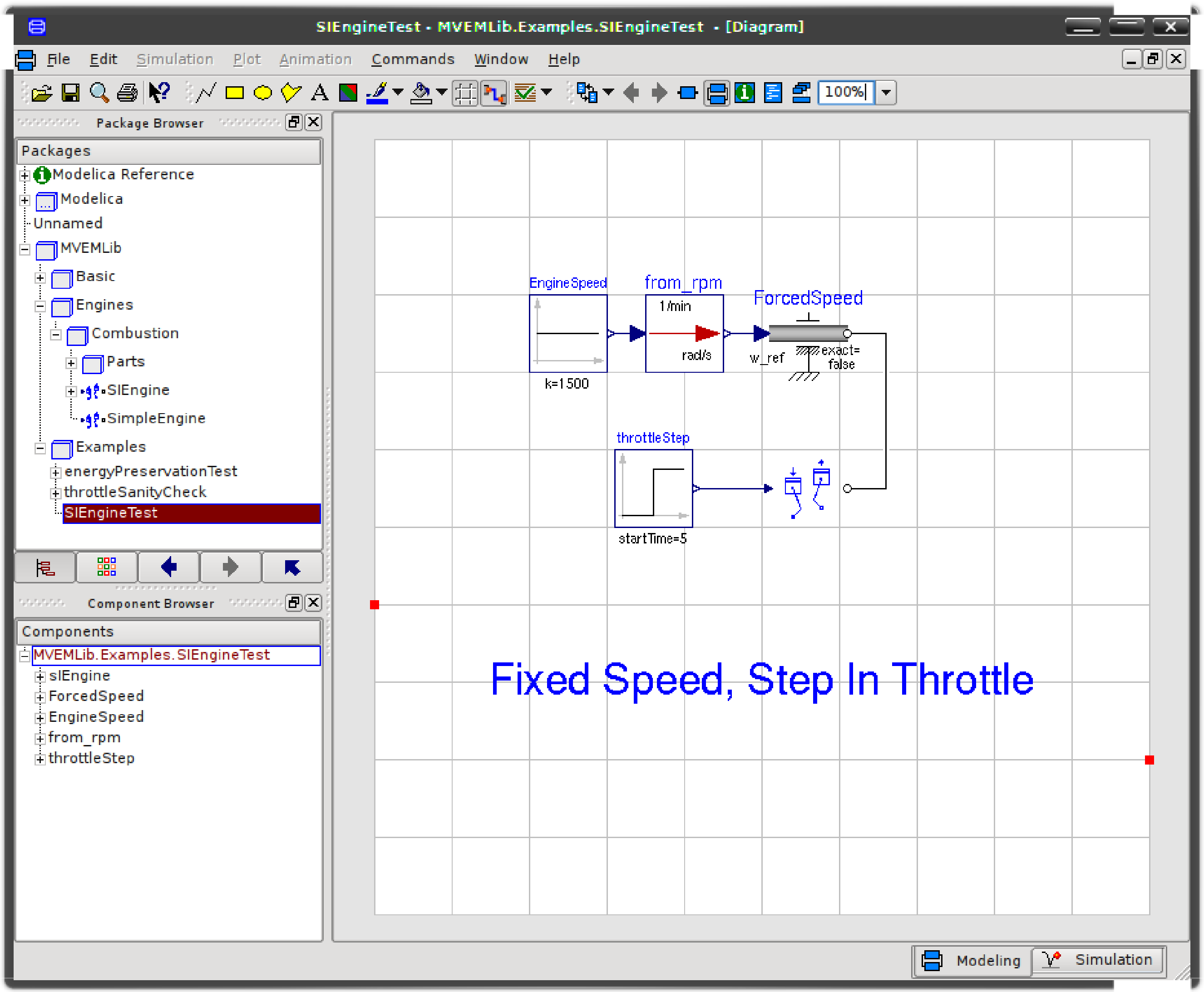


T6.14 - Demonstrators with SI and CI turbocharger engine simulators with controller

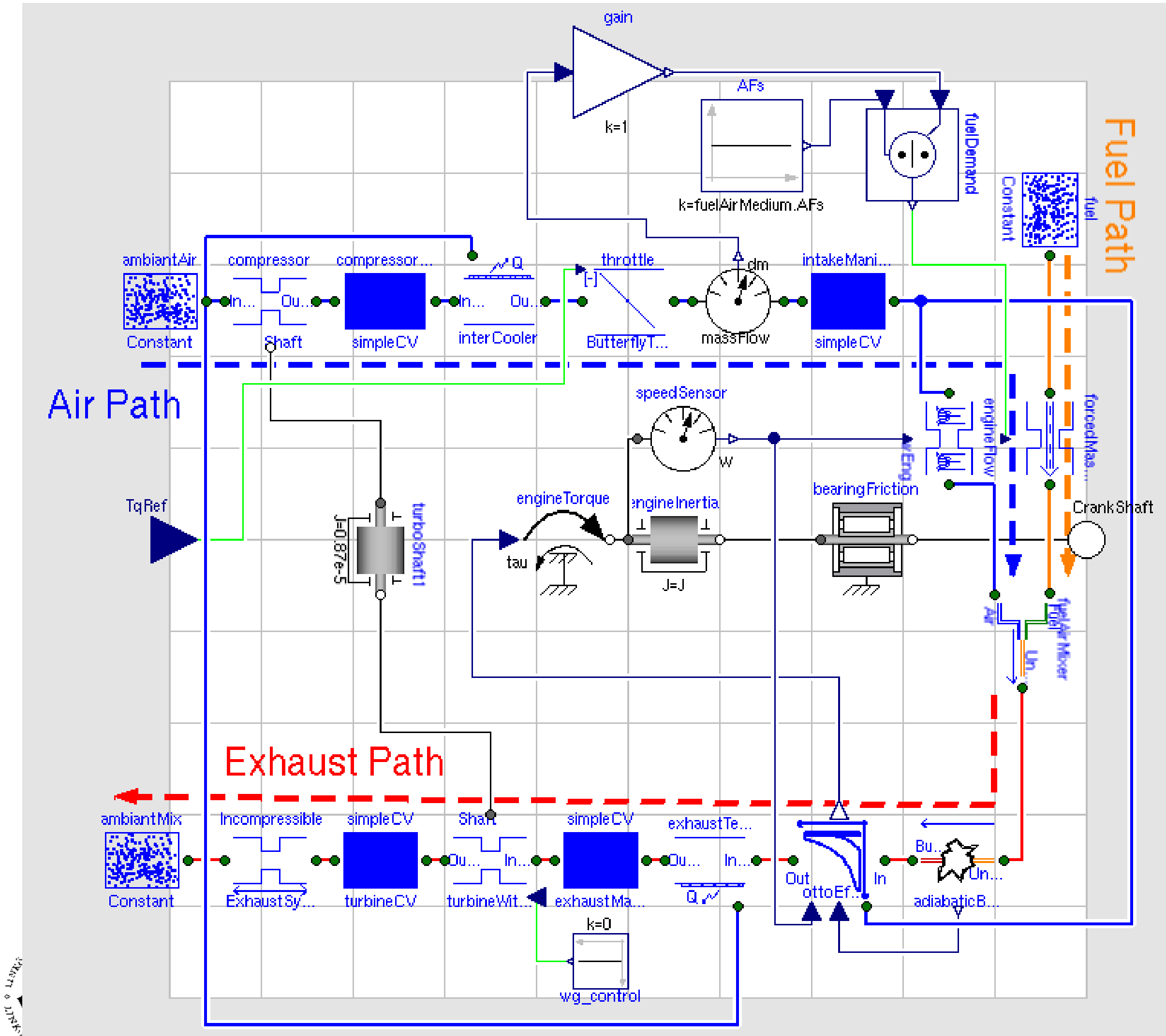
Linköping University
expanding reality

OPENPROD

MVEM Lib SI-Engine Demonstator Example



Model Overview



Model Consist of

- Separate fuel path with simple controller based on measured air mass flow
- Air paths with
  - Compressor
  - Intercooler
  - Throttle
  - Intake manifold
- Mixing of Unburned gases
- Exhaust path with
  - Combustion
  - Otto efficiency
  - Turbine
  - Exhaust system

OPENPROD

Basic classes for the demonstrator Example

- GasPort
- TwoPort
- FixedVolume
- IdealRestriction (and NonIdealRestriction)
- FuelAirMixture

  Consistant set of models for

  - Fuel
  - Air
  - Unburned gas
  - Burned gas
- FuelAirMixer
- AdiabaticBurner

GasPort model

- Similar to FluidPort model of MSL
- 0D modeling of fluid flow

```
connector GasPort
  replaceable package Medium =
         Modelica.Media.Interfaces.PartialMedium;
  SI.Pressure p "Pressure in the connector point";
  SI.Temperature T "Temperature in the connector point";
  SI.MassFraction Xi[Medium.nXi] "Mass fractions …";
  flow SI.EnthalpyFlowRate dH "Specific enthalpy flow through the connector";
  flow SI.MassFlowRate dm "Total Mass flow through the connector";
  flow SI.MassFlowRate dmXi[Medium.nXi] "Mass flows … ";
end GasPort;
```

TwoPort model

- Base class for flow componenst such as
  - Restrictions
  - Sensors

```
partial model TwoPort
  replaceable package Medium =
      Modelica.Media.Interfaces.PartialMedium
  Interfaces.GasPort InPut(redeclare replaceable package Medium = Medium)
  Interfaces.GasPort OutPut(redeclare replaceable package Medium = Medium)
end TwoPort;
```

FixedVolume model

- Models vessel with fixed size
- Keeps track of energy and masses

```
class FixedVolume extends Basic.Restrictions.Partial.TwoPort;
  Medium.BaseProperties gas
protected
  SI.Mass m "Mass of system";
  SI.Volume V "Volume of system";
  SI.Mass mXi[Medium.nXi] "Mass of respective independent gas component";
  SI.InternalEnergy U;
equation
  …
  mXi = gas.Xi * m;
  der(m)  = InPut.dm  + OutPut.dm;
  der(mXi) = InPut.dmXi + OutPut.dmXi;
  der(U) = InPut.dH + OutPut.dH;
  U = m*gas.u;
  V = VStart;
  gas.p * V = m * gas.R * gas.T;
end FixedVolume;
```

IdealRestriction model

- Base class for compressible and incompressible flow
- Ideal in the sence that enthalpy flows right trough

```
partial class IdealRestriction "Partial model for all restrictions."
   extends MVEMLib.Basic.Restrictions.Partial.TwoPort;
   Medium.BaseProperties gas;
equation
  InPut.dm   = - OutPut.dm;
  InPut.dmXi = - OutPut.dmXi;
  InPut.dmXi =   InPut.dm * gas.Xi;
  InPut.dH   = - OutPut.dH;
  InPut.dH   = InPut.dm * gas.h;
end IdealRestriction;
```

FuelAirMixture model

- Base class for set of mixture models
- Implementation requires consistent set of medium models for
  - Fuel
  - Air
  - Unburned gas
  - Burned gas
- Includes functions for calculation of burned species

```
package FuelAirMixture
  package airMedium extends
    Modelica.Media.IdealGases.MixtureGases.CombustionAir;
  end airMedium;

  package fuelMedium extends
    Modelica.Media.IdealGases.Common.SingleGasNasa(…);
  end fuelMedium;
  .
  .
end FuelAirMixture;
```

FuelAirMixture model (cont.)

```
package FuelAirMixture
.
.
  package unburnedMedium extends
    Modelica.Media.IdealGases.Common.MixtureGasNasa(…);
  end unburnedMedium;

  package burnedMedium extends
    Modelica.Media.IdealGases.MixtureGases.FlueGasSixComponents(…);
  end burnedMedium;

  function calcBurnedFractions
    input  SI.MassFraction unburned_dmX[unburnedMedium.nX];
    input  SI.Temperature  Tburned;
    output SI.MassFraction burned_dmX[burnedMedium.nX];
  end calcBurnedFractions;
end FuelAirMixture;
```

FuelAirMixer model

- Models mixing of **fuel** and **air** into **unburned** medium
- Enables use of separate air and fuel paths which can be used to minimize the number of states
- Properties at input connectors are calculated from unburned
- Properties at output connector are calculated from adiabatic mixing of fuel and air according to flow of respective species

```
model FuelAirMixer
  replaceable package fuelAirMedium = FuelAirMixture;
  Interfaces.GasPort FuelInPut(redeclare replaceable package Medium =
        fuelAirMedium.fuelMedium);
  Interfaces.GasPort AirInPut(redeclare replaceable package Medium =
        fuelAirMedium.airMedium);
  Interfaces.GasPort MixOutPut(redeclare replaceable package Medium =
        fuelAirMedium.unburnedMedium);
equation
.
.
end FuelAirMixer;
```

AdiabaticBurner model

- Models heat release by transferring gas from the unburned medium representation to the burned medium.
- Requires that medium models use `excludeEnthalpyOfFormation = false`
- Released energy comes from

```
dh = burnedMedium.specificInternalEnergy(burnedGas.state)-
     unburnedMedium.specificInternalEnergy(unburnedGas.state);
```

```
model AdiabaticBurner
  replaceable package fuelAirMedium = FuelAirMixture;
  Interfaces.GasPort UnburnedInPut(…);
  Interfaces.GasPort BurnedOutPut(…);
protected
  fuelAirMedium.unburnedMedium.BaseProperties unburnedGas;
  fuelAirMedium.burnedMedium.BaseProperties burnedGas;
equation
.
.
end AdiabaticBurner;
```

AdiabaticBurner model (cont.)


```
model AdiabaticBurner
.
.
equation
  unburnedGas.p  = burnedGas.p;
  unburnedGas.p  = BurnedOutPut.p;
  unburnedGas.T  = burnedGas.T;
  unburnedGas.T  = BurnedOutPut.T;
  unburnedGas.Xi = UnburnedInPut.dmXi/UnburnedInPut.dm;
  burnedGas.Xi   = BurnedOutPut.dmXi/BurnedOutPut.dm;

  UnburnedInPut.p    = BurnedOutPut.p;
  UnburnedInPut.T    = BurnedOutPut.T;
  UnburnedInPut.Xi   = reference_X[…];//Dummy
  BurnedOutPut.dm    = -UnburnedInPut.dm;
  BurnedOutPut.dH    = -UnburnedInPut.dH;
  BurnedOutPut.dmXi = calcBurnedFractions(UnburnedInPut.dmXi,BurnedOutPut.T);
end AdiabaticBurner;
```

Conclusions

- This Demonstrator shows how the Modelica media models can be used for mean value engine modelling of a SI (and CI) engine.
- The implementation yields a possiblity to have different medium models with different complexities in the same framework.
  - Only the FuelAirMixture package needs to be replaced to change gas representation.
- During simulation studies different gas models with different complexities can be used at different stages without re-implementing any part of the models. This enables easy reuse of code.
- The use case relies heavily on Media library, specifically `repleaceable package`
- Support in openModelica is as of now limited and the use case thus drives the development by requiring advanced features.