

Variability Modeling and Automated Simulation Experiments

Faculty of Engineering / Research Group CEA

Thorsten Pawletta
Hendrik Folkerts
Christina Deatcu

E-Mail:
 {thorsten.pawletta, christina.deatcu}@hs-wismar.de
hendrik.folkerts@cea-wismar.de

Web:
www.hs-wismar.de / www.cea-wismar.de





Outline

1. Introduction
2. The case study
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
9. Conclusion



Outline

1. **Introduction** (T. Pawletta)
2. The case study
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
9. Conclusion



Introduction



Introduction

- Systems, such as IOT, CPS or Industry 4.0, have a **high degree of variability**
 - Modern cars have more than 50 electronic control units (ECU), each may be instantiated in at least 10.000 different ways.⁽¹⁾
 - **Millions of configurations (variants)**

⁽¹⁾ Oster S. (2011) „Feature Model-based Software Product Line Testing“. PhD thesis, TU Darmstadt.



Introduction

- Systems, such as IOT, CPS or Industry 4.0, have a **high degree of variability**
 - Modern cars have more than 50 electronic control units (ECU), each may be instantiated in at least 10.000 different ways.⁽¹⁾
 - **Millions of configurations (variants)**
- **Variability** is defined in Software Engineering as:
the ability of a system to be configurable, extendable or adaptable depending on its purpose and objective. → **SPL engineering (SPLE)**

⁽¹⁾ Oster S. (2011) „Feature Model-based Software Product Line Testing“. PhD thesis, TU Darmstadt.



Introduction

- Systems, such as IOT, CPS or Industry 4.0, have a **high degree of variability**
 - Modern cars have more than 50 electronic control units (ECU), each may be instantiated in at least 10.000 different ways.⁽¹⁾
 - **Millions of configurations (variants)**
- **Variability** is defined in Software Engineering as:
the ability of a system to be configurable, extendable or adaptable depending on its purpose and objective. → **SPL engineering (SPLE)**
- **M&S:** Management of model families (set of models with common features)

⁽¹⁾ Oster S. (2011) „Feature Model-based Software Product Line Testing“. PhD thesis, TU Darmstadt.



Introduction

- Systems, such as IOT, CPS or Industry 4.0, have a **high degree of variability**
 - Modern cars have more than 50 electronic control units (ECU), each may be instantiated in at least 10.000 different ways.⁽¹⁾
→ **Millions of configurations (variants)**
- **Variability** is defined in Software Engineering as:
the ability of a system to be configurable, extendable or adaptable depending on its purpose and objective. → **SPL engineering (SPLE)**
- **M&S:** Management of model families (set of models with common features)
 - Today methods often transferred 1:1 from SPLE to M&S domain

⁽¹⁾ Oster S. (2011) „Feature Model-based Software Product Line Testing“. PhD thesis, TU Darmstadt.



Introduction

- Systems, such as IOT, CPS or Industry 4.0, have a **high degree of variability**
 - Modern cars have more than 50 electronic control units (ECU), each may be instantiated in at least 10.000 different ways.⁽¹⁾
→ **Millions of configurations (variants)**
- **Variability** is defined in Software Engineering as:
the ability of a system to be configurable, extendable or adaptable depending on its purpose and objective. → **SPL engineering (SPLE)**
- **M&S:** Management of model families (set of models with common features)
 - Today methods often transferred 1:1 from SPLE to M&S domain
 - More suitable are methods from **model based systems engineering** (→ **SES/MB approach**)

⁽¹⁾ Oster S. (2011) „Feature Model-based Software Product Line Testing“. PhD thesis, TU Darmstadt.



Introduction (2)



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - Specific paradigms apply to M&S:



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - **Specific paradigms apply to M&S:**
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - **Specific paradigms apply to M&S:**
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)
 - Separation of model and experiment (use of a model in different context)



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - **Specific paradigms apply to M&S:**
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)
 - Separation of model and experiment (use of a model in different context)
 - Separation of model and simulator



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - Specific paradigms apply to M&S:
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)
 - Separation of model and experiment (use of a model in different context)
 - Separation of model and simulator
 - Modular-hierarchical modeling of configurations
(variants of structure & parameter settings)



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - Specific paradigms apply to M&S:
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)
 - Separation of model and experiment (use of a model in different context)
 - Separation of model and simulator
 - Modular-hierarchical modeling of configurations
(variants of structure & parameter settings)
- **Why should a model be used with different simulators?**



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - Specific paradigms apply to M&S:
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)
 - Separation of model and experiment (use of a model in different context)
 - Separation of model and simulator
 - Modular-hierarchical modeling of configurations
(variants of structure & parameter settings)
- **Why should a model be used with different simulators?**
 - domain dependency, (?) simulator correctness



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - Specific paradigms apply to M&S:
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)
 - Separation of model and experiment (use of a model in different context)
 - Separation of model and simulator
 - Modular-hierarchical modeling of configurations
(variants of structure & parameter settings)
- **Why should a model be used with different simulators?**
 - domain dependency, (?) simulator correctness
- **Study of variants is time-consuming**



Introduction (2)

- **Why shouldn't SPLE be transferred 1:1 to the M&S domain?**
 - Specific paradigms apply to M&S:
 - Separation of model structures & model dynamics
(violation in 150% modeling approach)
 - Separation of model and experiment (use of a model in different context)
 - Separation of model and simulator
 - Modular-hierarchical modeling of configurations
(variants of structure & parameter settings)
- **Why should a model be used with different simulators?**
 - domain dependency, (?) simulator correctness
- **Study of variants is time-consuming** 
 - **Automation** of simulation experiments (selection of configurations, model generation & simulation, analysis of simulation results)



Outline

1. Introduction
2. **The case study** (T. Pawletta)
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
9. Conclusion

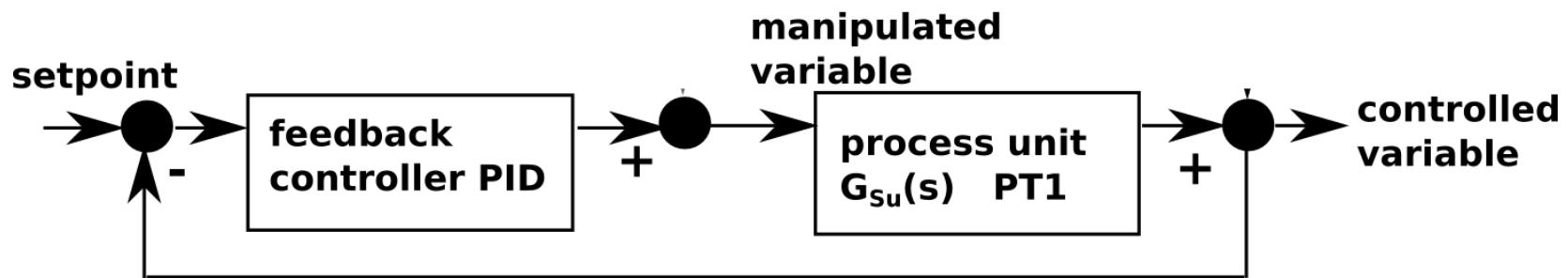


Case Study



Case Study

- Feedback control system

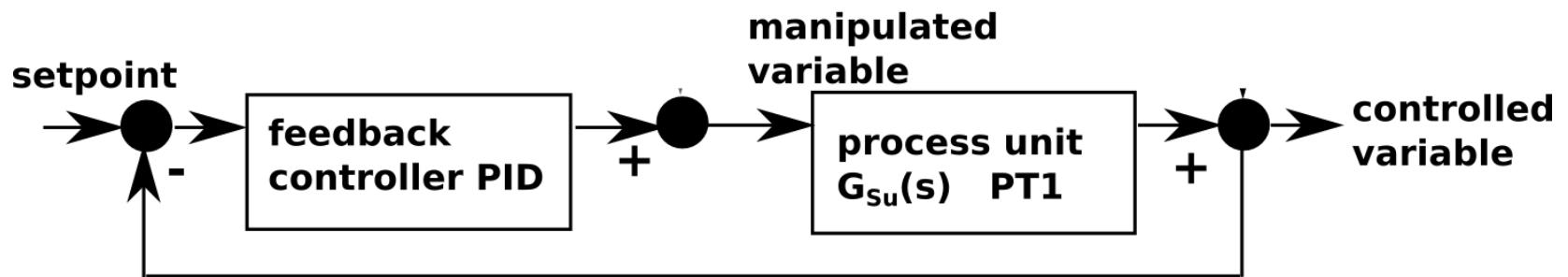




Case Study

- Feedback control system
- Described by transfer functions

$$G_{Su}(s) = \frac{1}{20 \cdot s + 1}$$



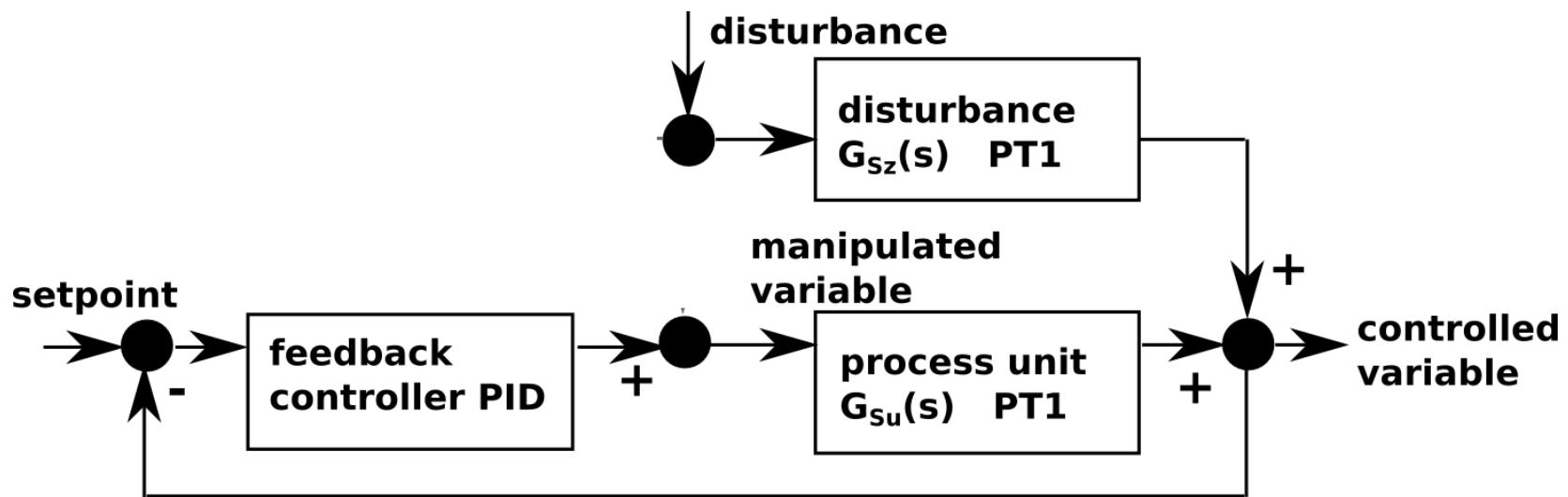


Case Study

- Feedback control system
- Described by transfer functions
- Influenced by disturbances

$$G_{Su}(s) = \frac{1}{20 \cdot s + 1}$$

$$G_{Sz}(s) = \frac{1}{10 \cdot s + 1}$$





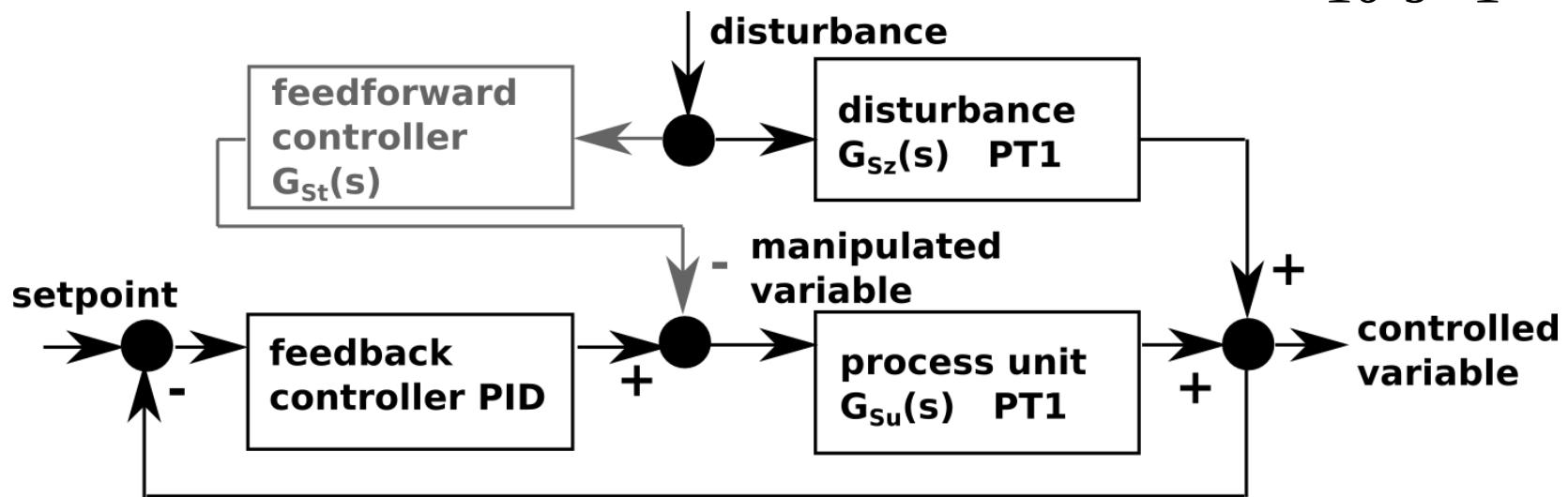
Case Study

- Feedback control system
- Described by transfer functions
- Influenced by disturbances
- Measurable disturbances
 - Compensated with feedforward control

$$G_{Su}(s) = \frac{1}{20 \cdot s + 1}$$

$$G_{Sz}(s) = \frac{1}{10 \cdot s + 1}$$

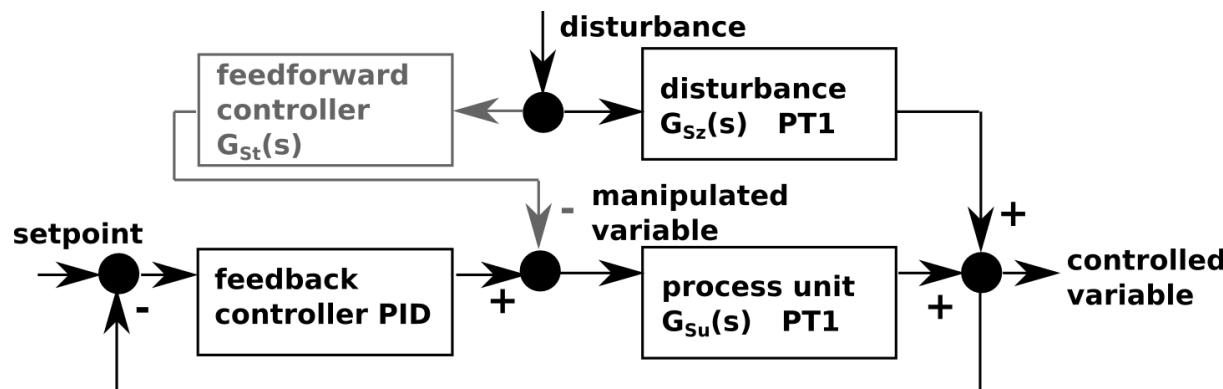
$$G_{St}(s) = \frac{20 \cdot s + 1}{10 \cdot s + 1}$$





Case Study (2)

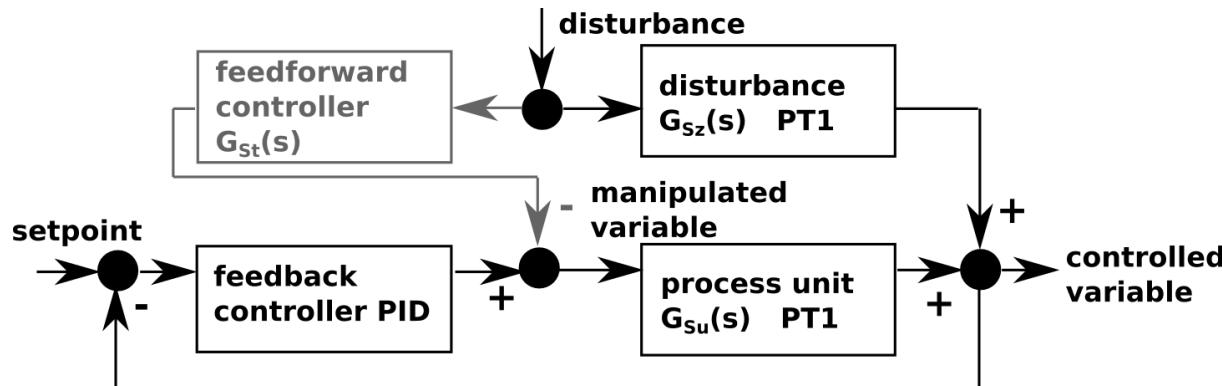
- **Two system structure variants**
 - Without feedforward control: $\text{feedforward}=0$
 - With feedforward control: $\text{feedforward}=1$





Case Study (2)

- **Two system structure variants**
 - Without feedforward control: $\text{feedforward}=0$
 - With feedforward control: $\text{feedforward}=1$
- For every structure variant
 - **Different parameter configurations of PID controller**
(we consider two)

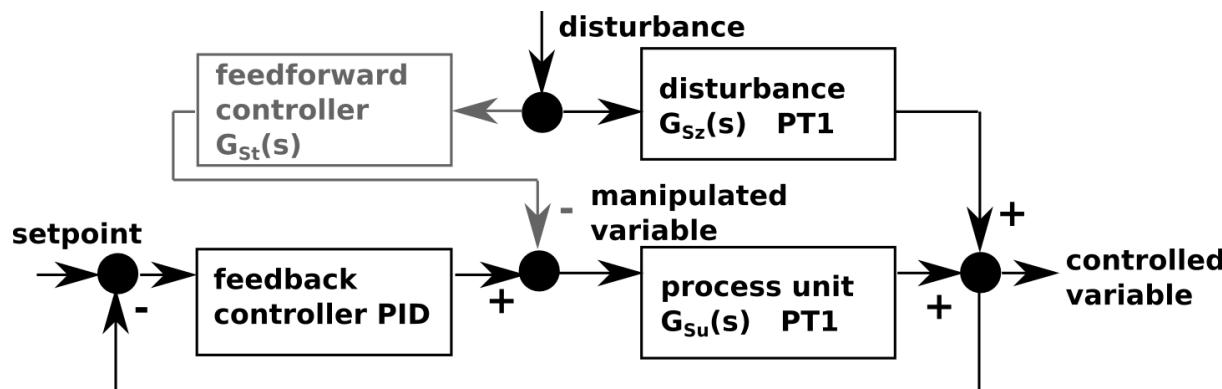




Case Study (2)

Design objective:
Find best control configuration.

- **Two system structure variants**
 - Without feedforward control: $\text{feedforward}=0$
 - With feedforward control: $\text{feedforward}=1$
- For every structure variant
 - **Different parameter configurations of PID controller**
(we consider two)



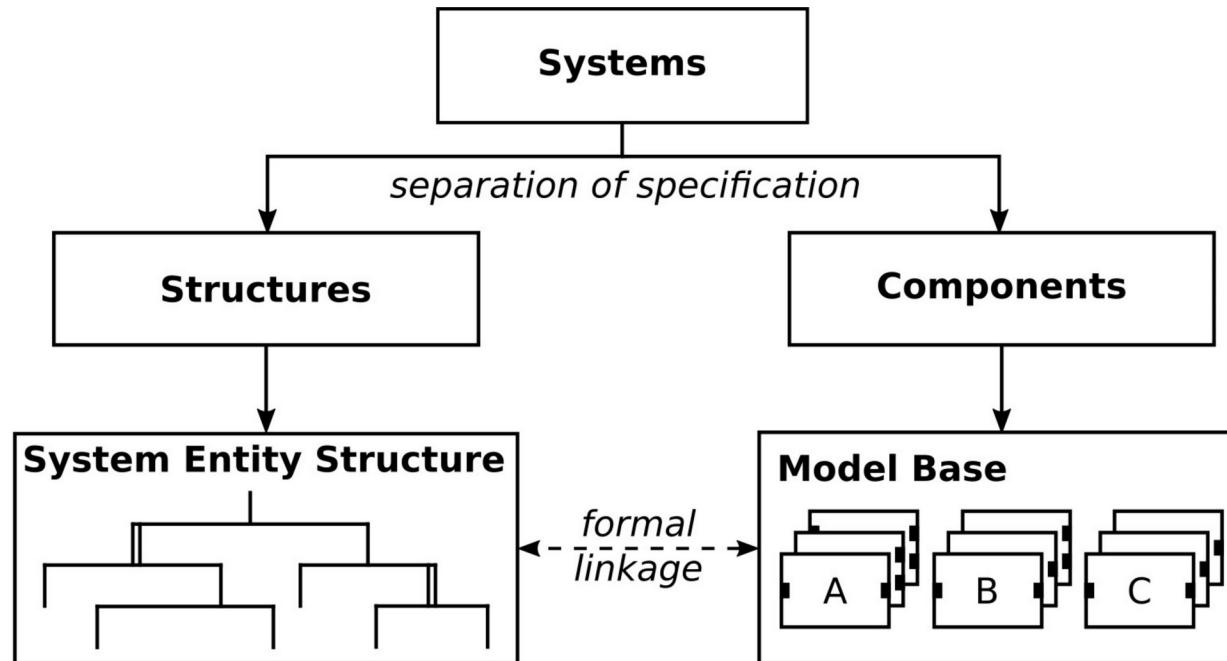


Outline

1. Introduction
2. The case study
- 3. Basics of SES/MB based modeling** (T. Pawletta)
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
9. Conclusion

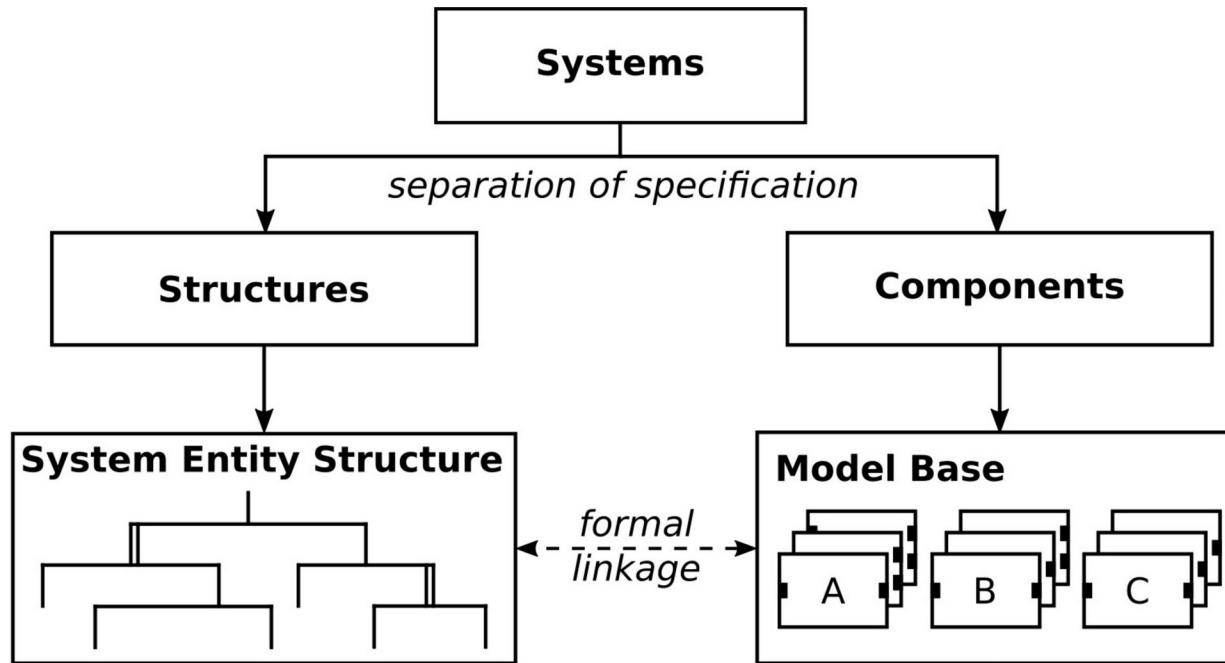


SES/MB Modeling Approach





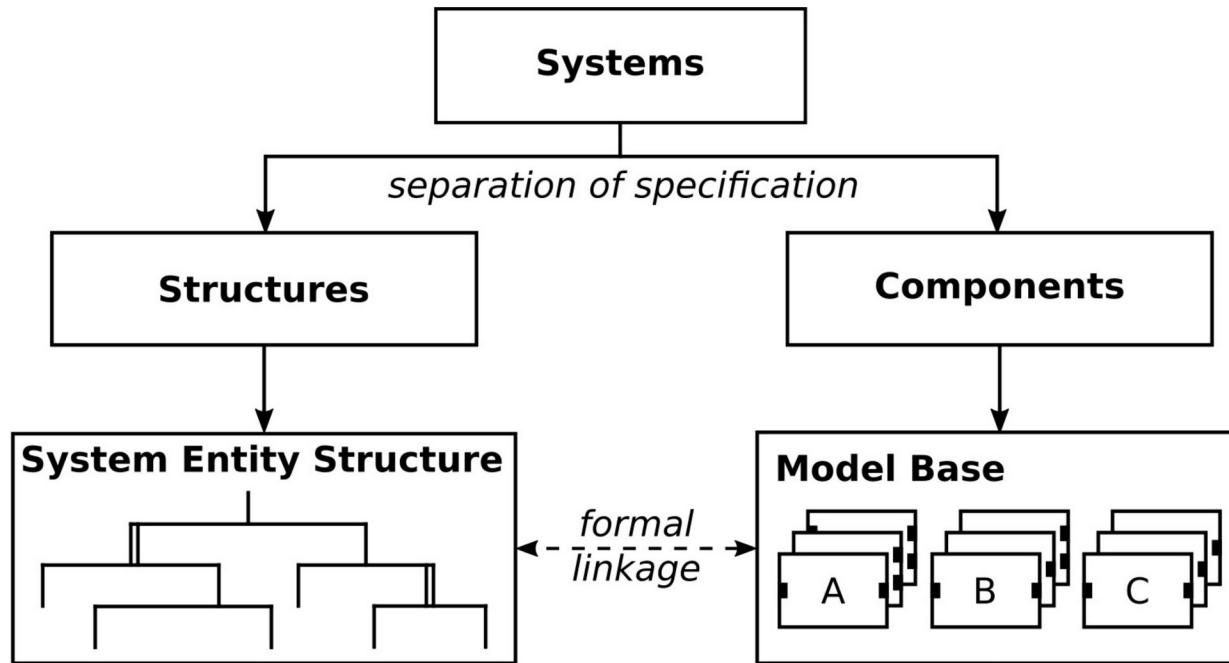
SES/MB Modeling Approach



- **SES** describes permissible structure & parameter variants (simulator-independent)



SES/MB Modeling Approach



- **SES** describes permissible structure & parameter variants (simulator-independent)
- **MB** defines basic dynamic models (usually simulator dependent)



Basics of the System Entity Structure (SES)



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)
- **SES is a tree structure**



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)
- **SES is a tree structure**
 - Well defined by axioms



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)
- **SES is a tree structure**
 - Well defined by axioms
 - Two types of nodes
 - Entity nodes
 - Descriptive nodes



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)
- **SES is a tree structure**
 - Well defined by axioms
 - Two types of nodes
 - Entity nodes
 - Descriptive nodes

Entity nodes
real or imaginary
objects

Descriptive nodes
Aspect
(Multi-aspect)
Specialization



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)
- **SES is a tree structure**
 - Well defined by axioms
 - Two types of nodes
 - Entity nodes
 - Descriptive nodes
 - Three types of edges (relations betw. nodes)

Entity nodes
real or imaginary
objects

Descriptive nodes
Aspect
(Multi-aspect)
Specialization



Basics of the System Entity Structure (SES)

- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)
- **SES is a tree structure**
 - Well defined by axioms
 - Two types of nodes
 - Entity nodes
 - Descriptive nodes
 - Three types of edges (relations betw. nodes)
 - Node/Edge specific attributes

Entity nodes
real or imaginary
objects

Descriptive nodes
Aspect
(Multi-aspect)
Specialization



Basics of the System Entity Structure (SES)

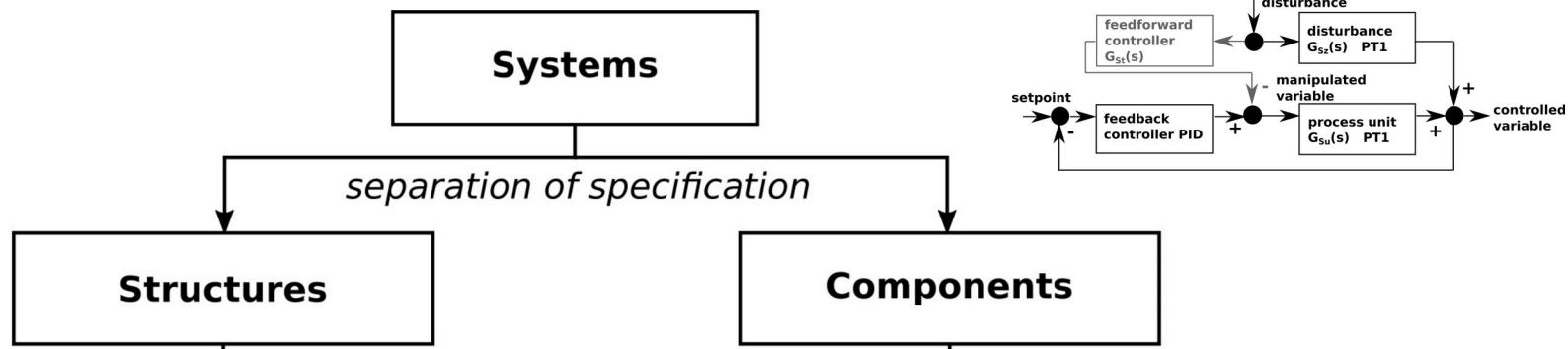
- SES introduced by B.P. Zeigler and J. Rozenblit
- Amongst others extended by research group CEA (Wismar)
- **SES is a tree structure**
 - Well defined by axioms
 - Two types of nodes
 - Entity nodes
 - Descriptive nodes
 - Three types of edges (relations betw. nodes)
 - Node/Edge specific attributes
 - Global variables, functions, constraints, ...

Entity nodes
real or imaginary
objects

Descriptive nodes
Aspect
(Multi-aspect)
Specialization

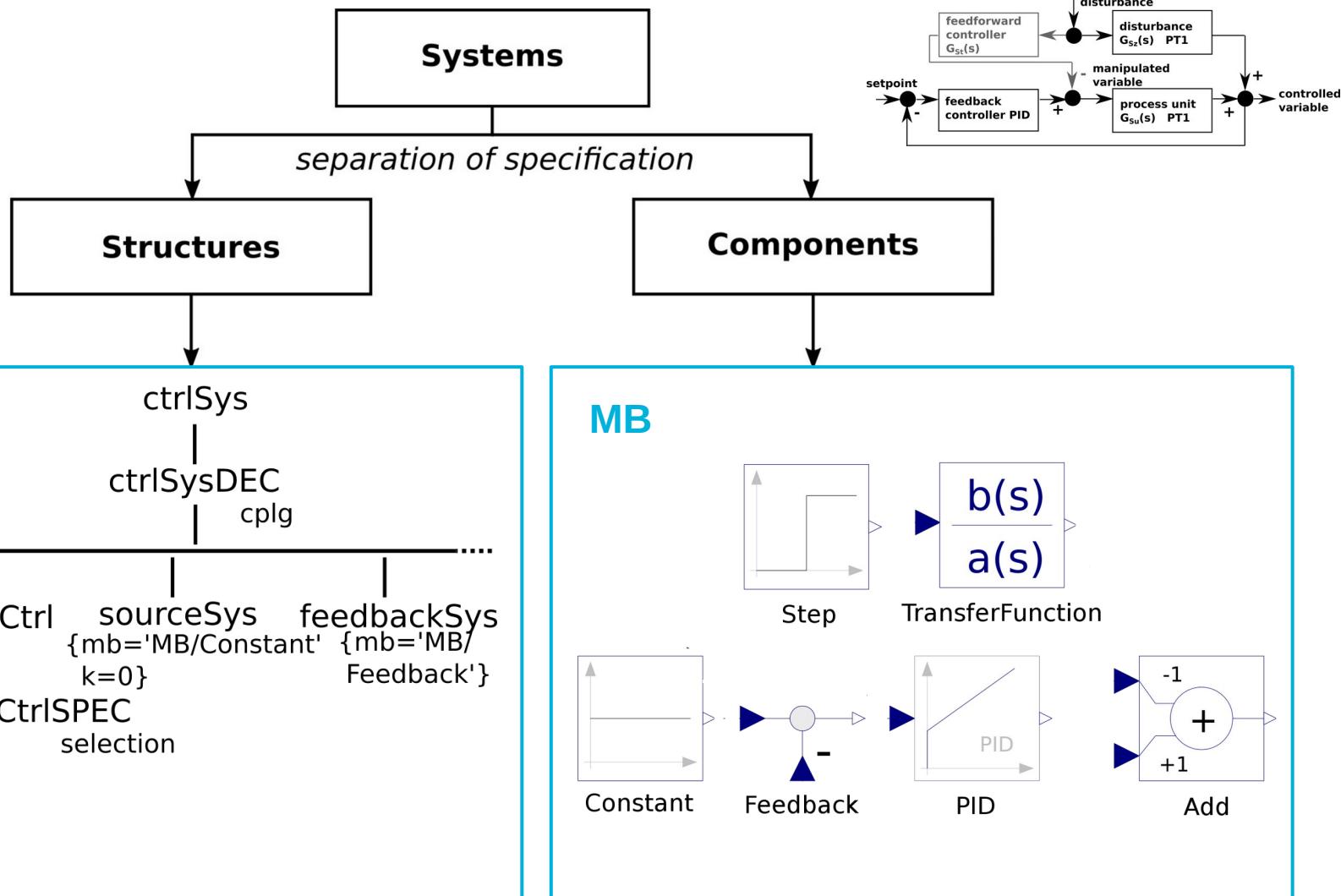


SES/MB-based Modeling of the Case Study



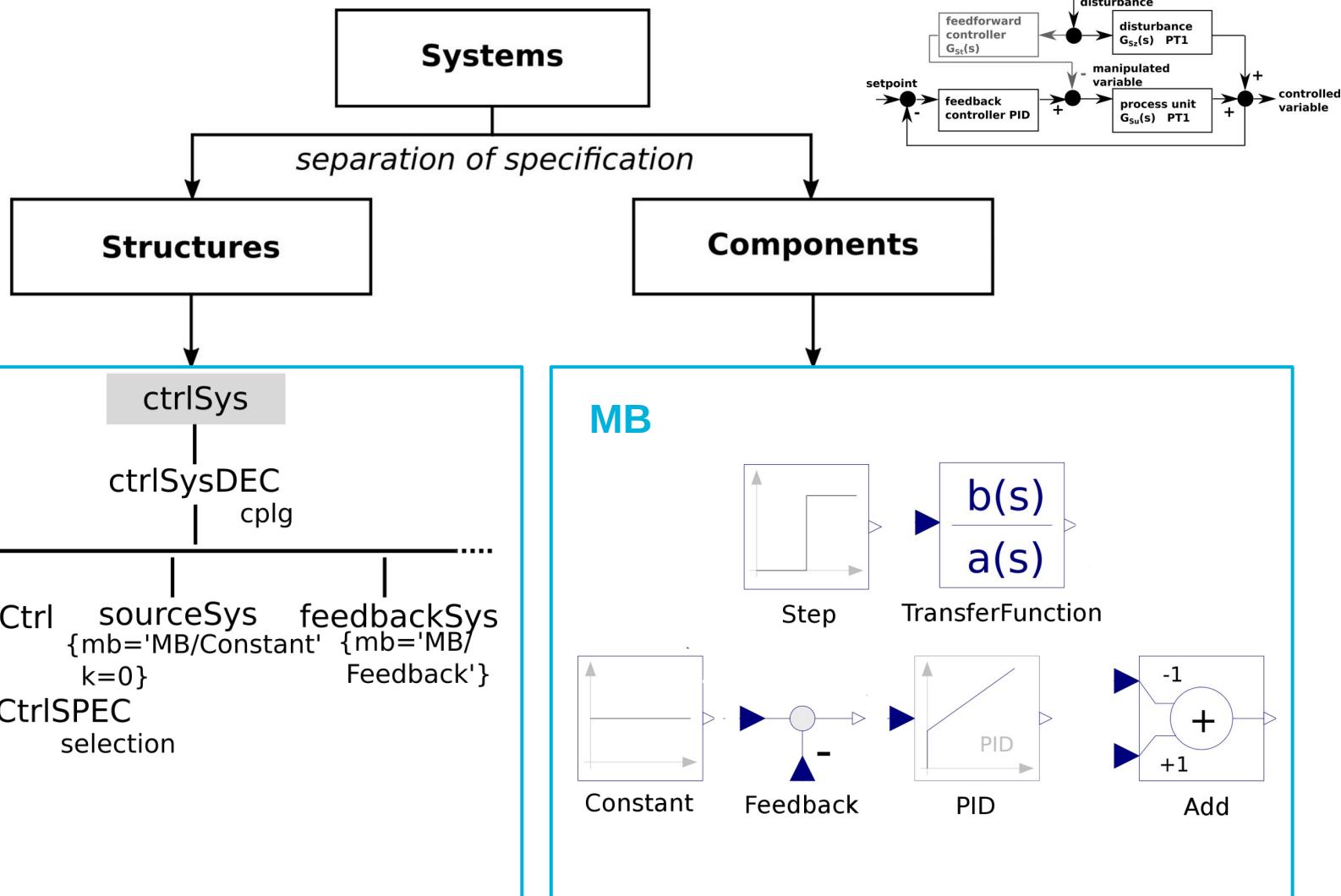


SES/MB-based Modeling of the Case Study



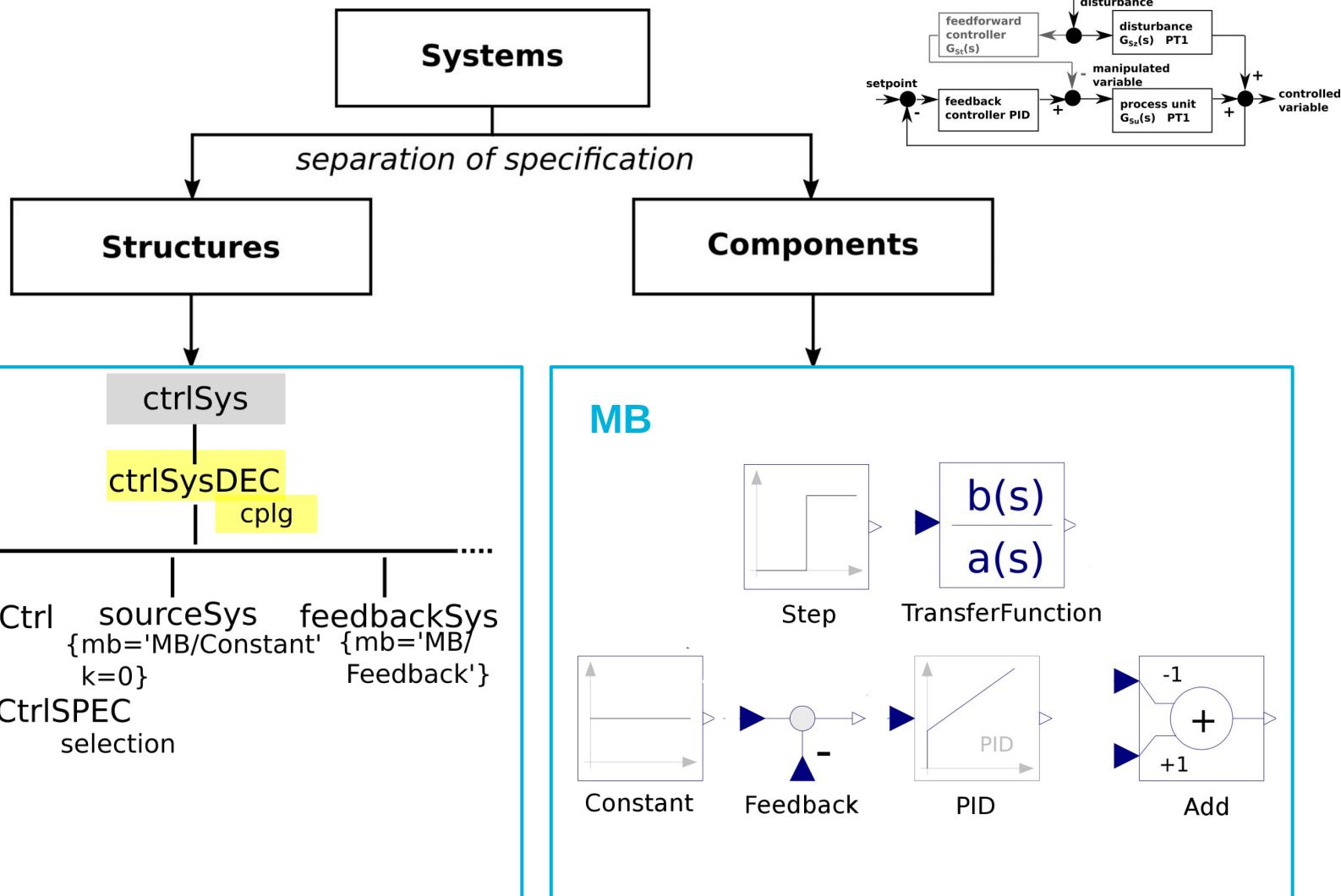


SES/MB-based Modeling of the Case Study



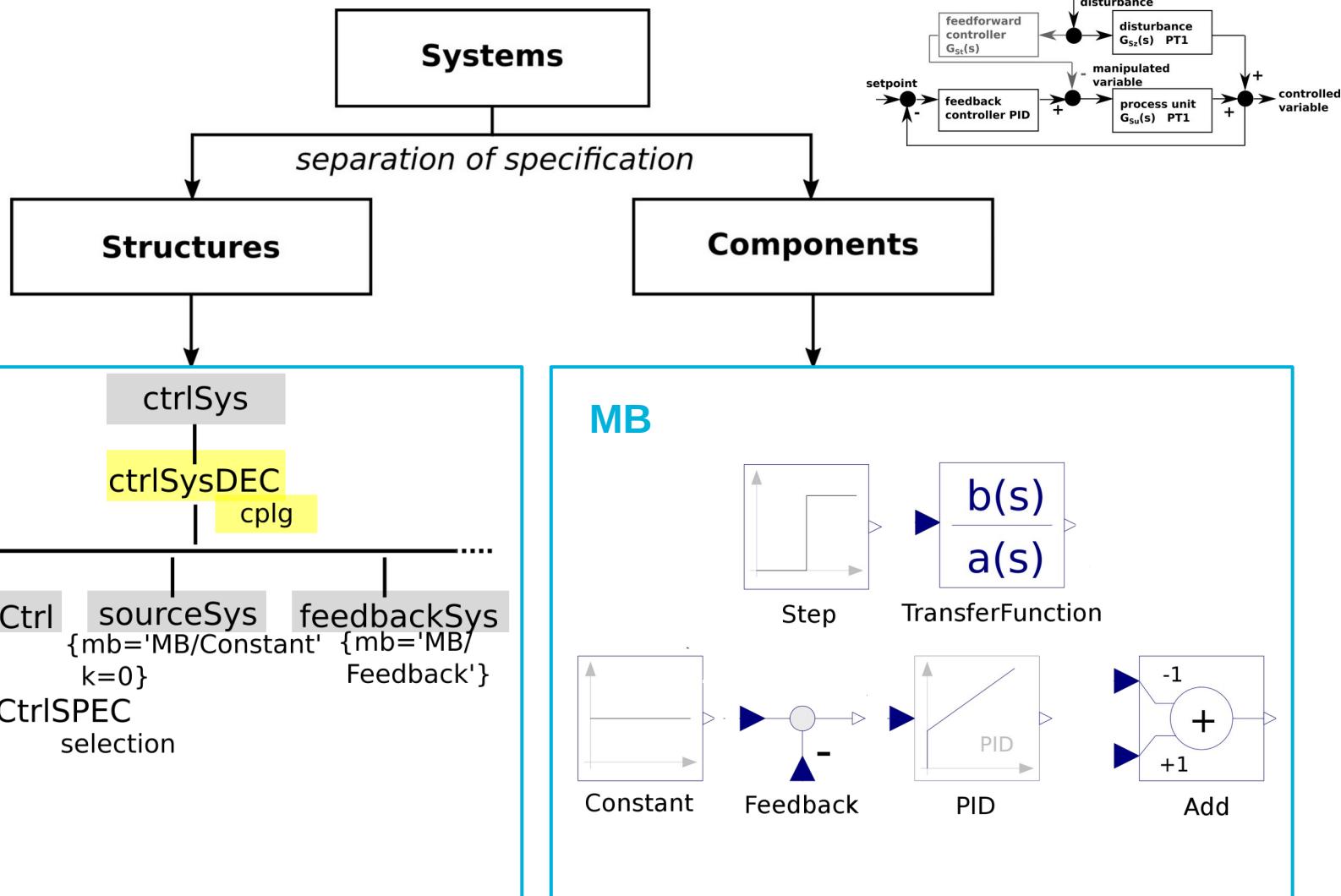


SES/MB-based Modeling of the Case Study



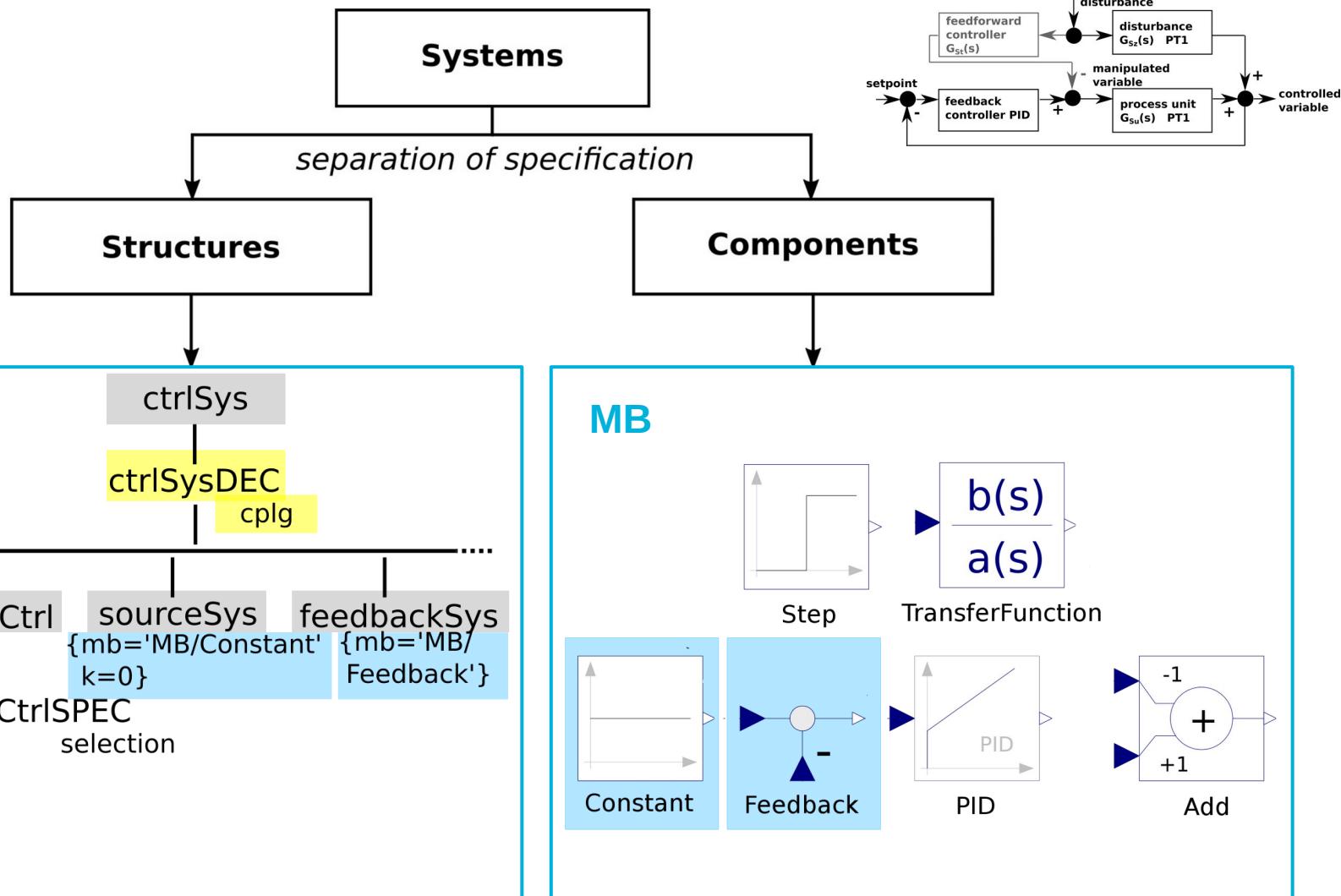


SES/MB-based Modeling of the Case Study



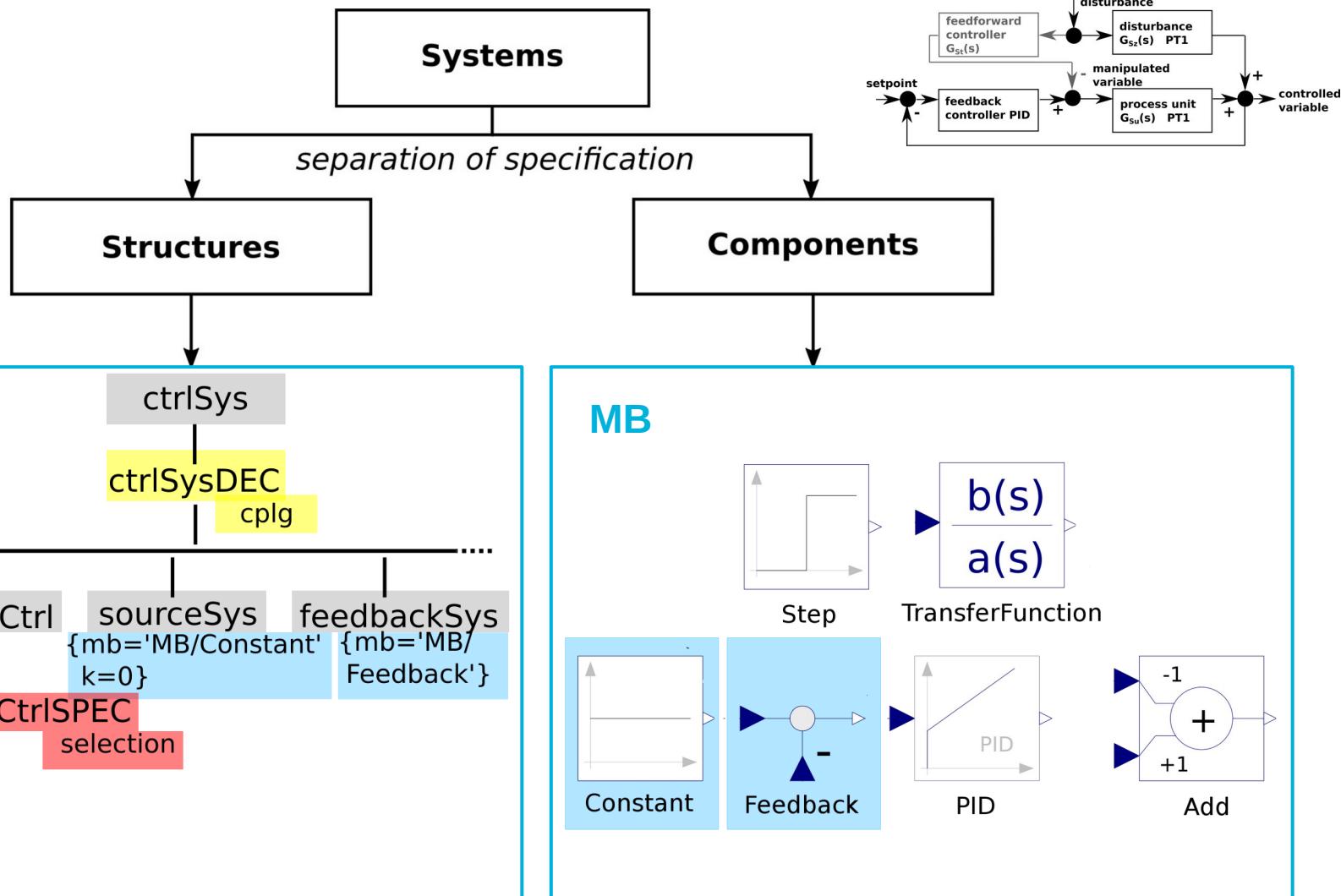


SES/MB-based Modeling of the Case Study



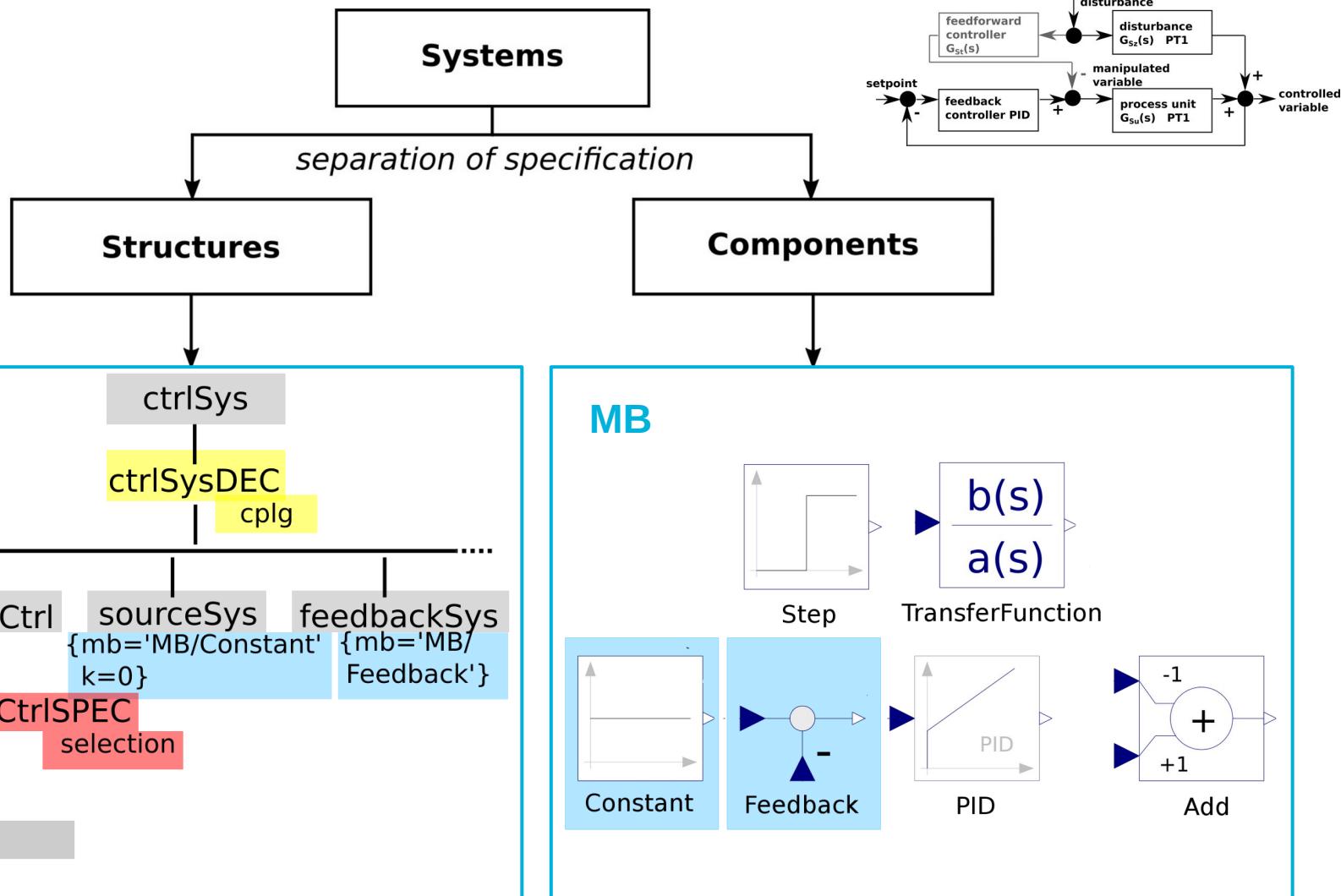


SES/MB-based Modeling of the Case Study





SES/MB-based Modeling of the Case Study



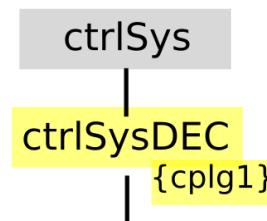


More Detailed Extract of the SES

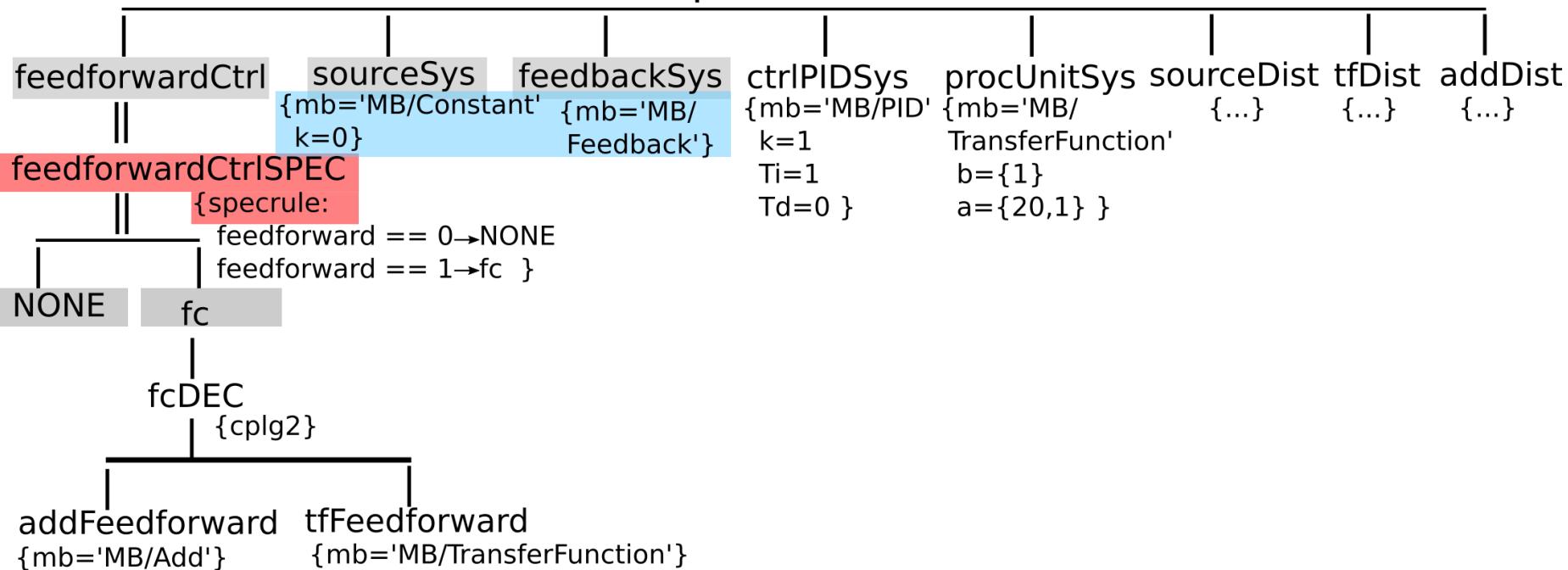
SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	



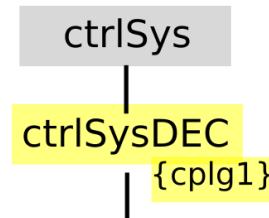


More Detailed Extract of the SES

SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	

feedforwardCtrl **sourceSys** **feedbackSys**

||

{mb='MB/Constant'
k=0}{mb='MB/
Feedback'}**ctrlPIDSys**{mb='MB/PID'
k=1
Ti=1
Td=0 }**procUnitSys**{mb='MB/
TransferFunction'
b={1}
a={20,1} }**sourceDist**

{...}

tfDist

{...}

addDist

{...}

feedforwardCtrlSPEC

||

{specrule:

feedforward == 0→NONE
feedforward == 1→fc }**NONE****fc****fcDEC**

{cplg2}

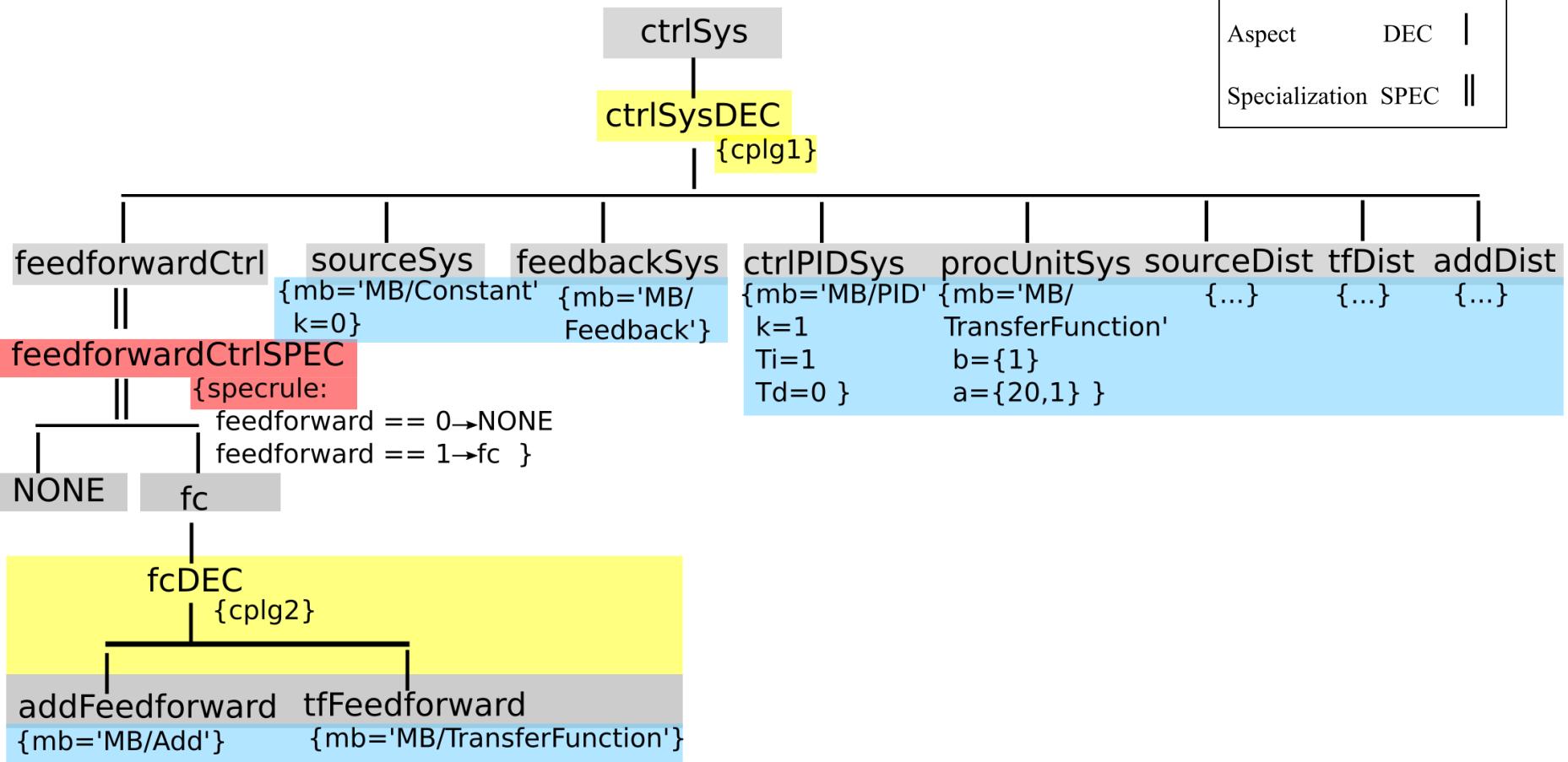
addFeedforward
{mb='MB/Add'}**tfFeedforward**
{mb='MB/TransferFunction'}



More Detailed Extract of the SES

SESSESVAR={feedforward}
SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	



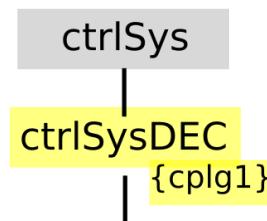


More Detailed Extract of the SES

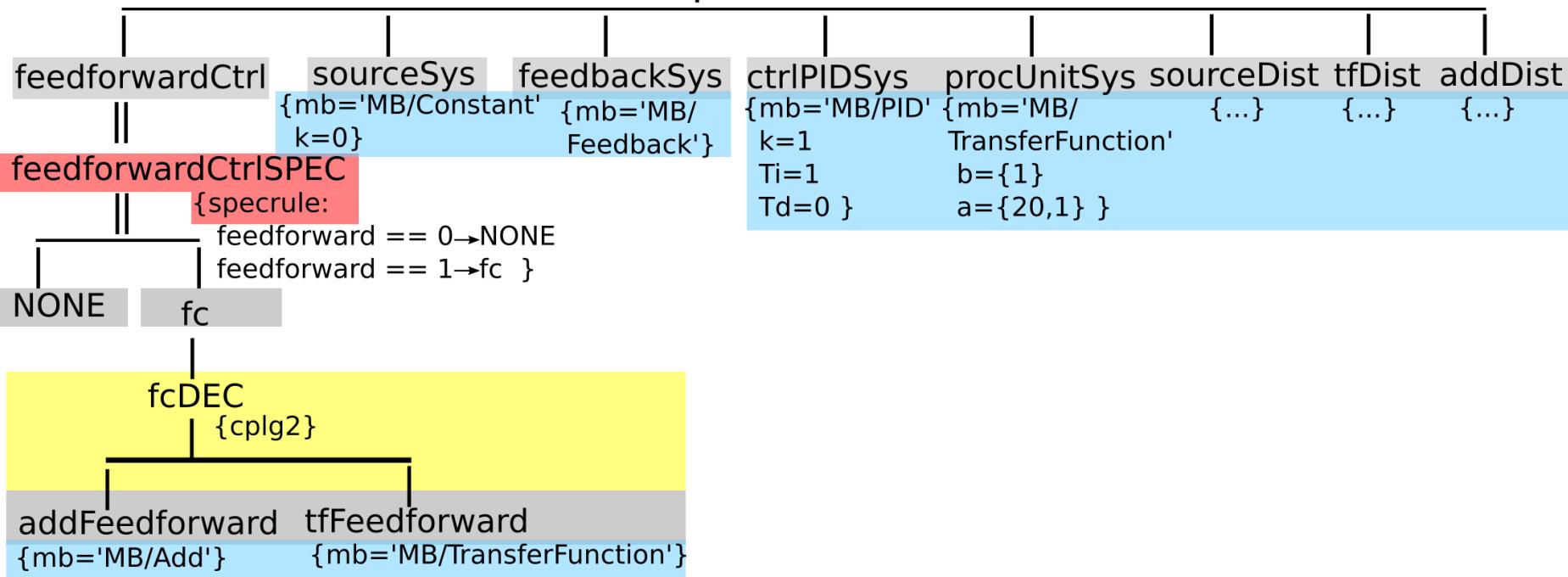
SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	



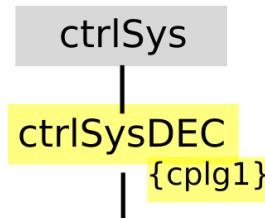


More Detailed Extract of the SES

SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	

```
feedforwardCtrl | sourceSys | feedbackSys
```

```
||
```

```
{mb='MB/Constant'  
k=0}
```

```
{mb='MB/  
Feedback'}
```

```
ctrlPIDSys
```

```
{mb='MB/PID'  
k=1  
Ti=1  
Td=0 }
```

```
procUnitSys
```

```
{mb='MB/  
TransferFunction'  
b={1}  
a={20,1} }
```

```
sourceDist
```

```
{...}
```

```
tfDist
```

```
{...}
```

```
addDist
```

```
{...}
```

```
feedforwardCtrlSPEC
```

```
||
```

```
{specrule:
```

```
feedforward == 0→NONE  
feedforward == 1→fc }
```

```
NONE
```

```
fc
```

```
fcDEC
```

```
{cplg2}
```

```
addFeedforward  
{mb='MB/Add'}
```

```
tfFeedforward  
{mb='MB/TransferFunction'}
```

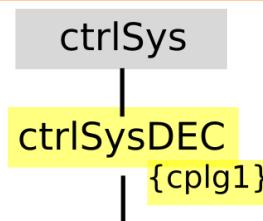


More Detailed Extract of the SES

SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	

```
feedforwardCtrl | sourceSys | feedbackSys
```

```
||
```

```
{mb='MB/Constant'  
k=0}
```

```
{mb='MB/  
Feedback'}
```

```
ctrlPIDSys
```

```
{mb='MB/PID'  
k=1  
Ti=1  
Td=0 }
```

```
procUnitSys
```

```
{mb='MB/  
TransferFunction'  
b={1}  
a={20,1} }
```

```
sourceDist
```

```
{...}
```

```
tfDist
```

```
{...}
```

```
addDist
```

```
{...}
```

```
feedforwardCtrlSPEC
```

```
||
```

```
{specrule:
```

```
feedforward == 0→NONE  
feedforward == 1→fc }
```

```
NONE fc
```

```
fcDEC
```

```
{cplg2}
```

```
addFeedforward tfFeedforward
```

```
{mb='MB/Add'}
```

```
{mb='MB/TransferFunction'}
```

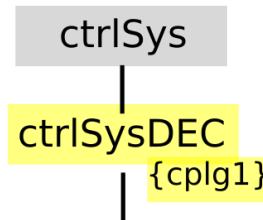


More Detailed Extract of the SES

SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	

```
feedforwardCtrl | sourceSys | feedbackSys
```

```
||
```

```
{mb='MB/Constant'  
k=0}
```

```
{mb='MB/  
Feedback'}
```

```
feedforwardCtrlSPEC
```

```
||
```

```
{specrule:
```

```
feedforward == 0→NONE
```

```
feedforward == 1→fc }
```

```
NONE | fc
```

```
fcDEC
```

```
{cplg2}
```

```
addFeedforward  
{mb='MB/Add'}
```

```
tfFeedforward  
{mb='MB/TransferFunction'}
```

```
ctrlPIDSys | procUnitSys | sourceDist | tfDist | addDist
```

```
{mb='MB/PID'  
k=1  
Ti=1  
Td=0 }
```

```
{mb='MB/
```

```
TransferFunction'
```

```
{...}
```

```
{...}
```

```
{...}
```

- Coupling attribute `cplg1` is specified using an SES function depending on `feedforward`

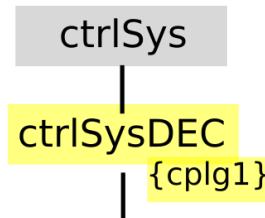


More Detailed Extract of the SES

SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	

```
feedforwardCtrl | sourceSys | feedbackSys
```

```
||
```

```
{mb='MB/Constant'  
k=0}
```

```
{mb='MB/  
Feedback'}
```

```
feedforwardCtrlSPEC
```

```
||
```

```
{specrule:
```

```
feedforward == 0→NONE  
feedforward == 1→fc }
```

```
NONE | fc
```

```
fcDEC
```

```
{cplg2}
```

```
addFeedforward | tfFeedforward
```

```
{mb='MB/Add'}
```

```
{mb='MB/TransferFunction'}
```

```
ctrlPIDSys | procUnitSys | sourceDist | tfDist | addDist
```

```
{mb='MB/PID'  
k=1  
Ti=1  
Td=0 }
```

```
{mb='MB/  
TransferFunction'
```

```
{...}
```

```
{...}
```

```
{...}
```

- Coupling attribute `cplg1` is specified using an SES function depending on `feedforward`
- Different parameter settings are not shown in this extract of SES

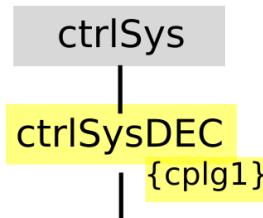


More Detailed Extract of the SES

SES

SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}



Type	Key	Suffix Edge
Aspect	DEC	
Specialization	SPEC	

feedforwardCtrl | sourceSys | feedbackSys

||

{mb='MB/Constant'
k=0}

{mb='MB/
Feedback'}

feedforwardCtrlSPEC

||

{specrule:

feedforward == 0→NONE
feedforward == 1→fc }

NONE | fc

fcDEC
{cplg2}

addFeedforward
{mb='MB/Add'}

tfFeedforward
{mb='MB/TransferFunction'}

ctrlPIDSys | procUnitSys | sourceDist | tfDist | addDist

{mb='MB/PID'
k=1
Ti=1
Td=0 } | {mb='MB/
TransferFunction'
b={1}
a={20,1} }

{...} | {...} | {...}

- Coupling attribute `cplg1` is specified using an SES function depending on `feedforward`
- Different parameter settings are not shown in this extract of SES



Demonstration
with software tool



Outline

1. Introduction
2. The case study
3. Basics of SES/MB based modeling
4. **Practical modeling: implementation of an SES**
(H. Folkerts)
5. Model selection and model generation
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
9. Conclusion



Python Toolset

- Available: <https://github.com/hendrikfolkerts>
- Tools
 - **SESToPy** → SES editor and IDE
 - **SESViewEl** → SES tree viewer
 - SESMoPy
 - SESEuPy
 - SESEcPy



Demonstration of SESToPy with SESViewEI (case study)

- Connect SESToPy with SESViewEI (show SESToPy and SESViewEI next to each other)
- Add sub node, add sibling node, change type of node, rename node, delete node, inflate tree, deflate tree
- Edit entity node, descriptive node (aspect, specialization)
- Empty current model
- Save/Load model (JSON) → load Feedback.jsonsestree example
- (Export/Import model (XML))
- Maximize SESToPy
- Use the feedback example to show:
 - SES Variables, Semantic Conditions
 - Selection rules → here: specrule
 - NONE node
 - Attributes, mb-attribute (decouple name of node and name of basic model)
 - Coupling list (composition of basic models)
 - SES function to set couplings (dynamic coupling) → procedural knowledge
- Mention merging

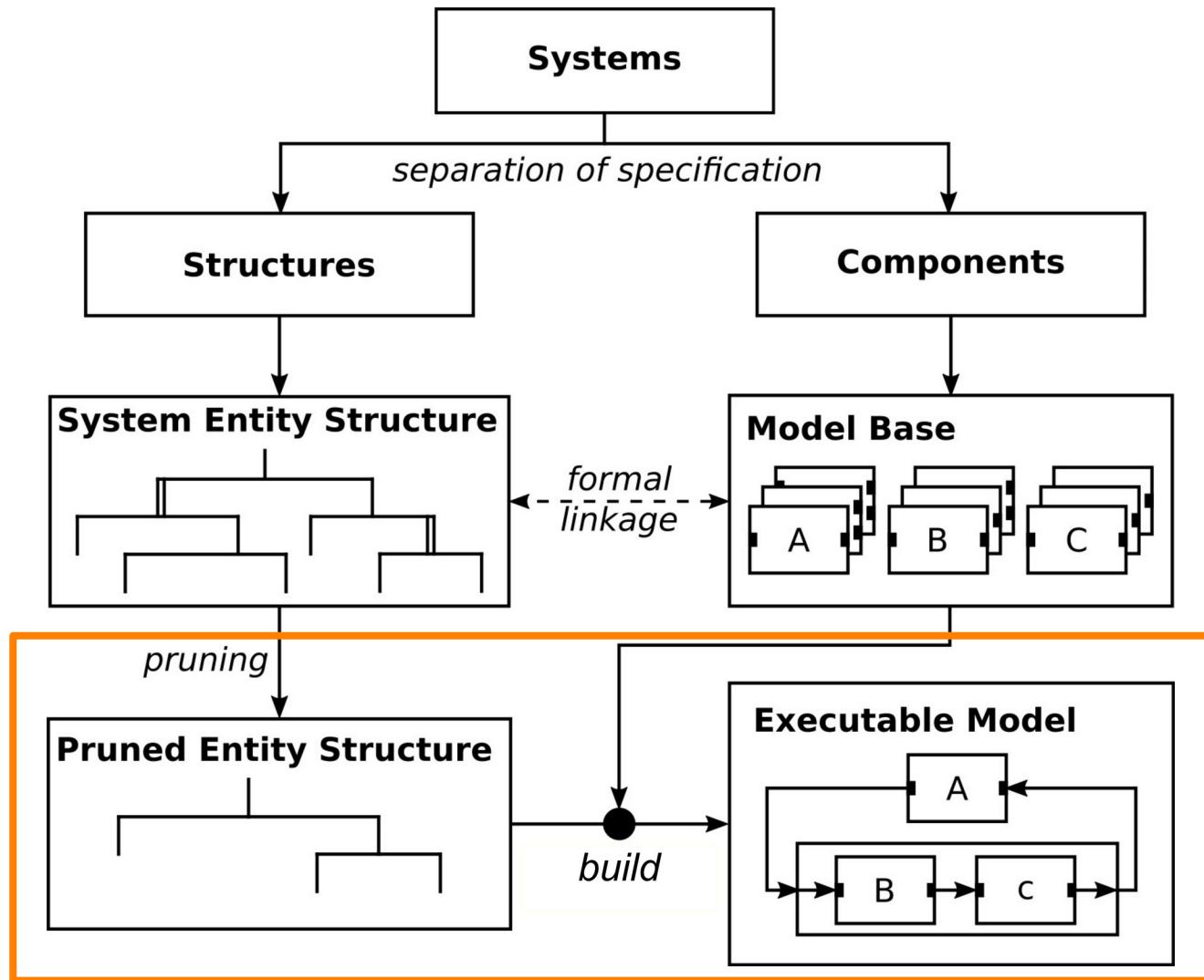


Outline

1. Introduction
2. The case study
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
- 5. Model selection and model generation** (H. Folkerts)
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
9. Conclusion



Model Selection and Generation





Model Selection and Generation of Variant #1

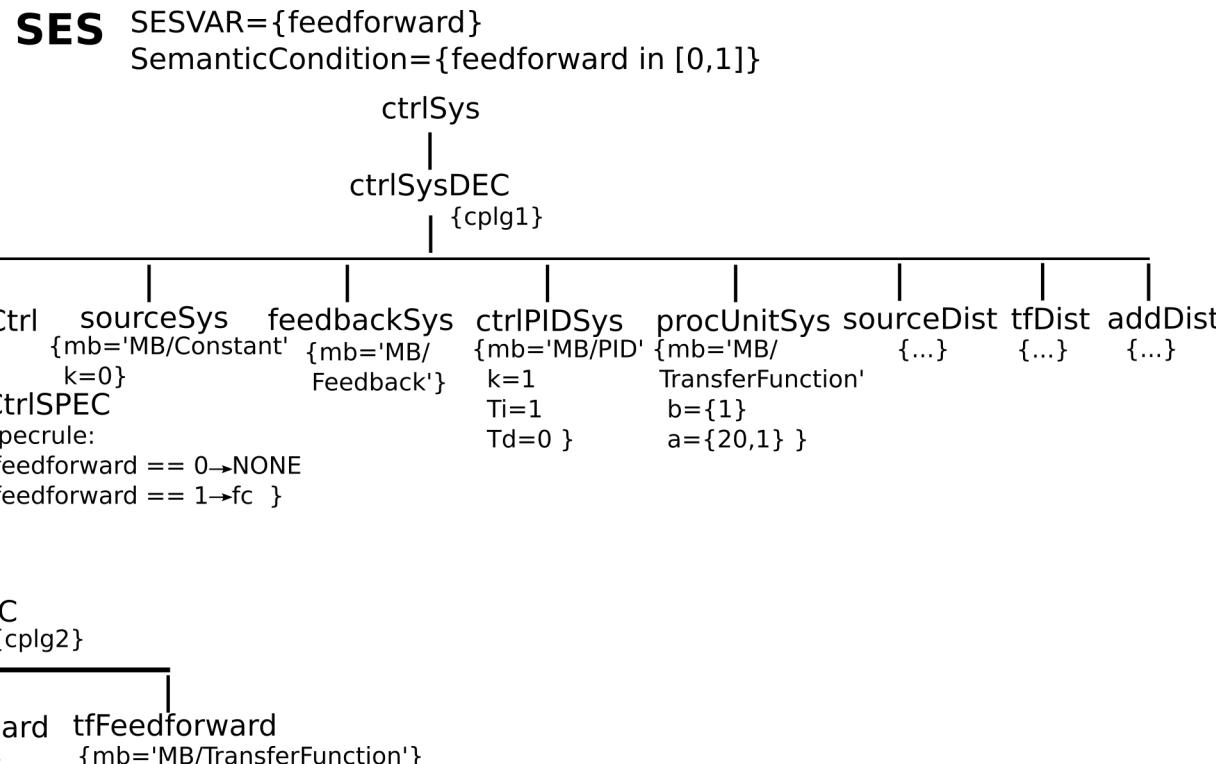
SES SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}

ctrlSys

ctrlSysDEC

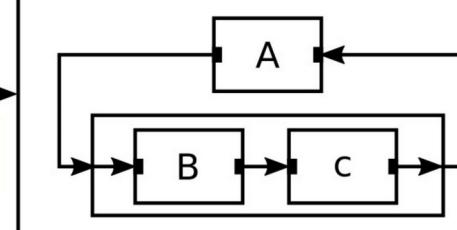
{cplg1}



Pruned Entity Structure

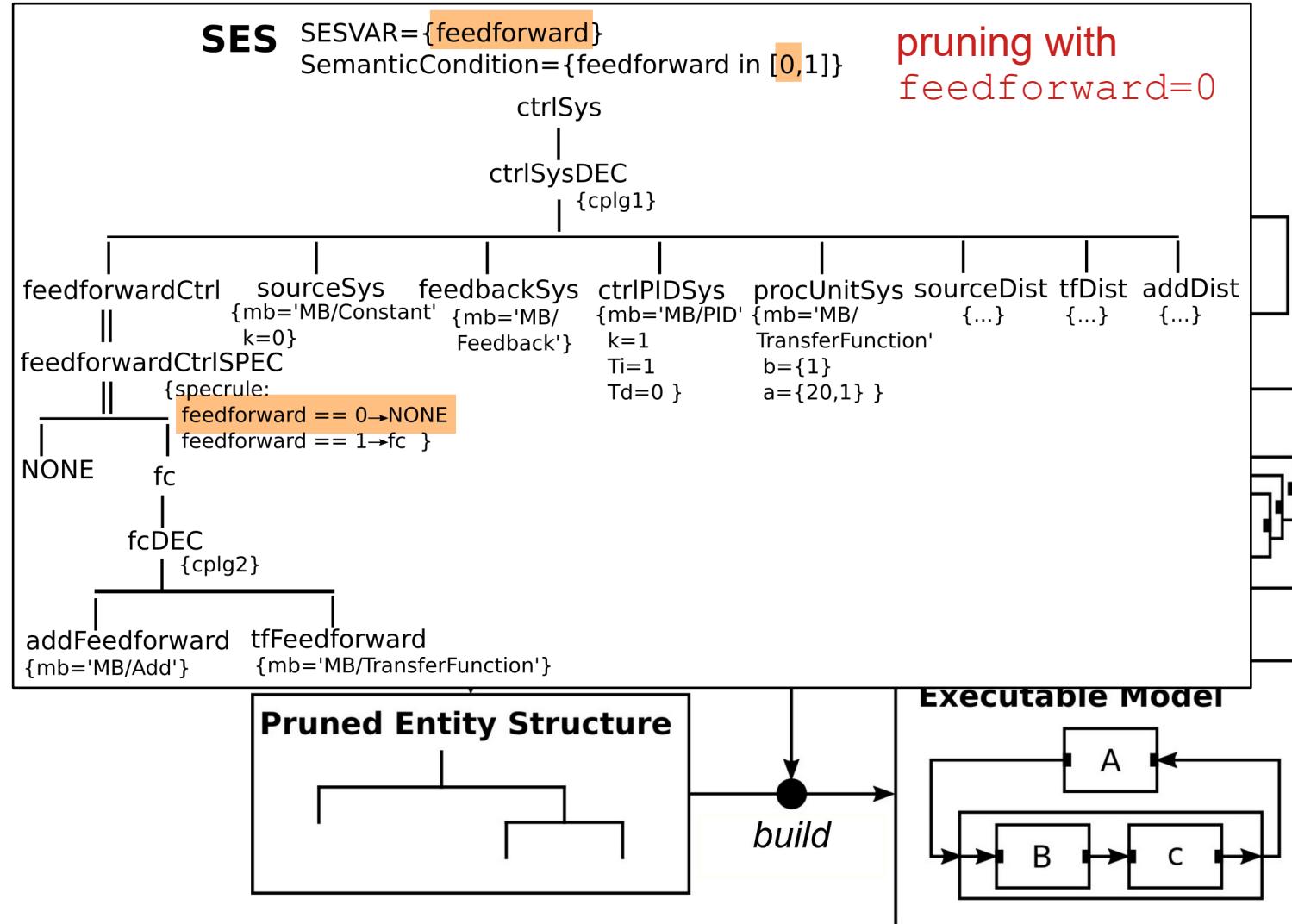
build

Executable Model



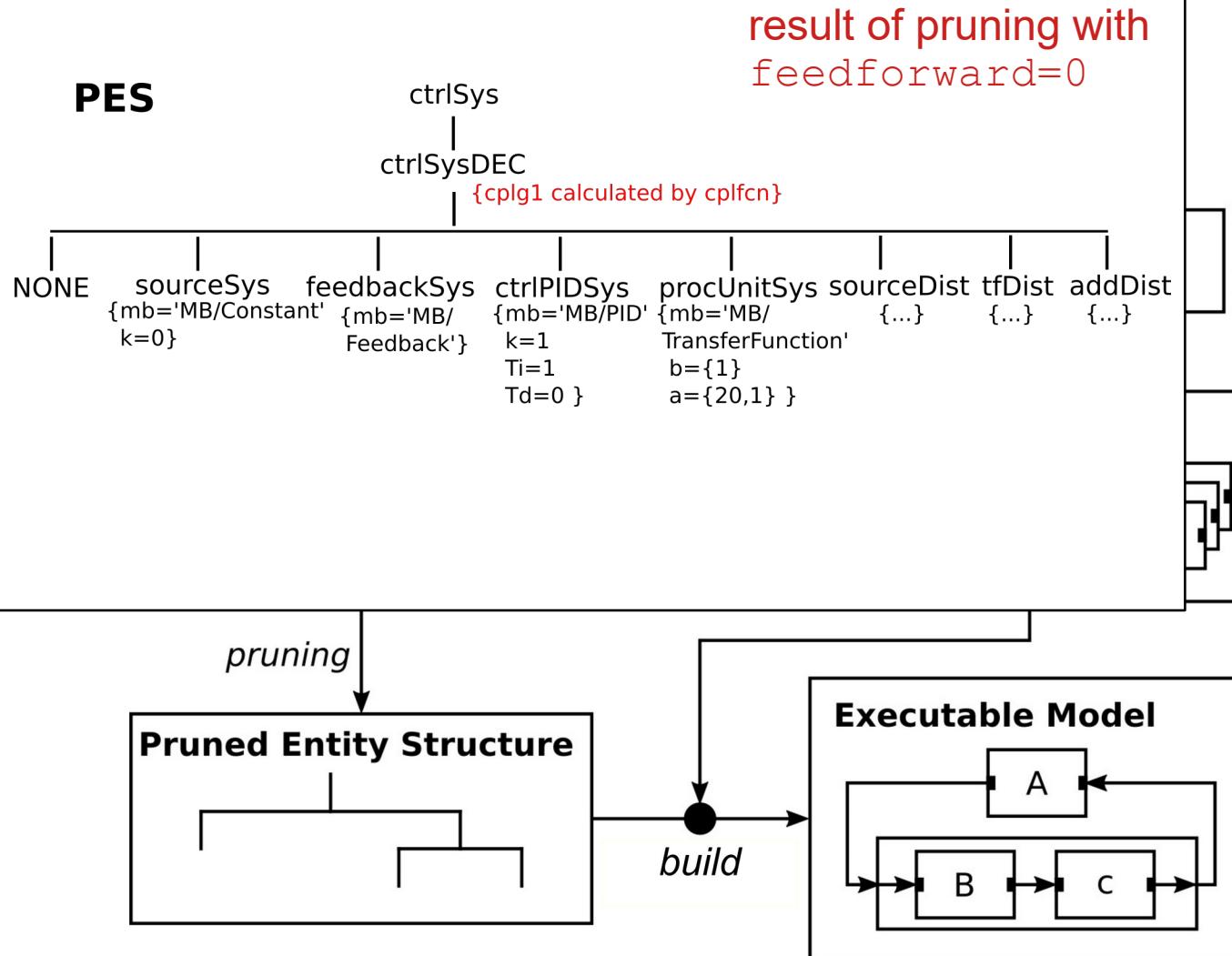


Model Selection and Generation of Variant #1



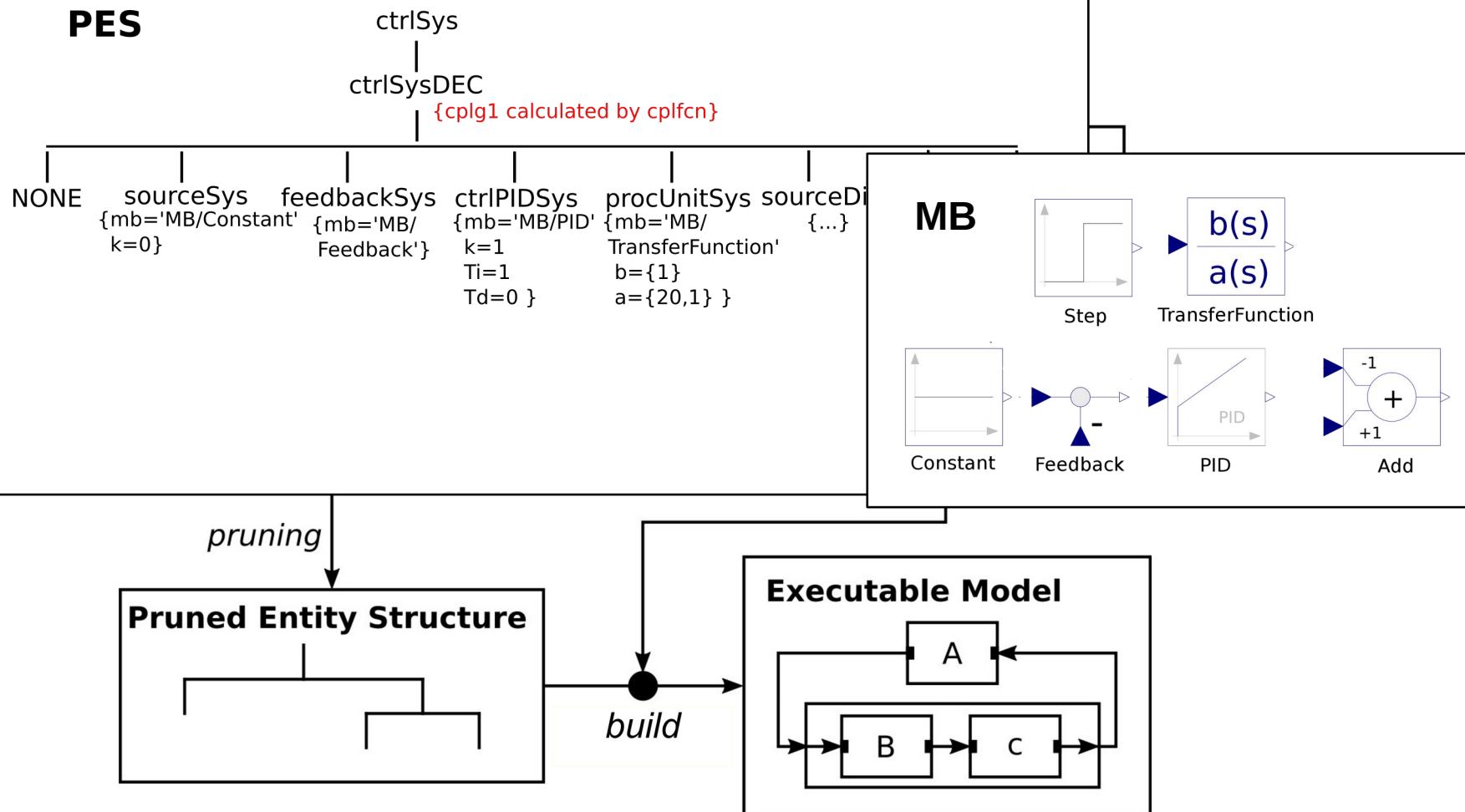


Model Selection and Generation of Variant #1 (2)



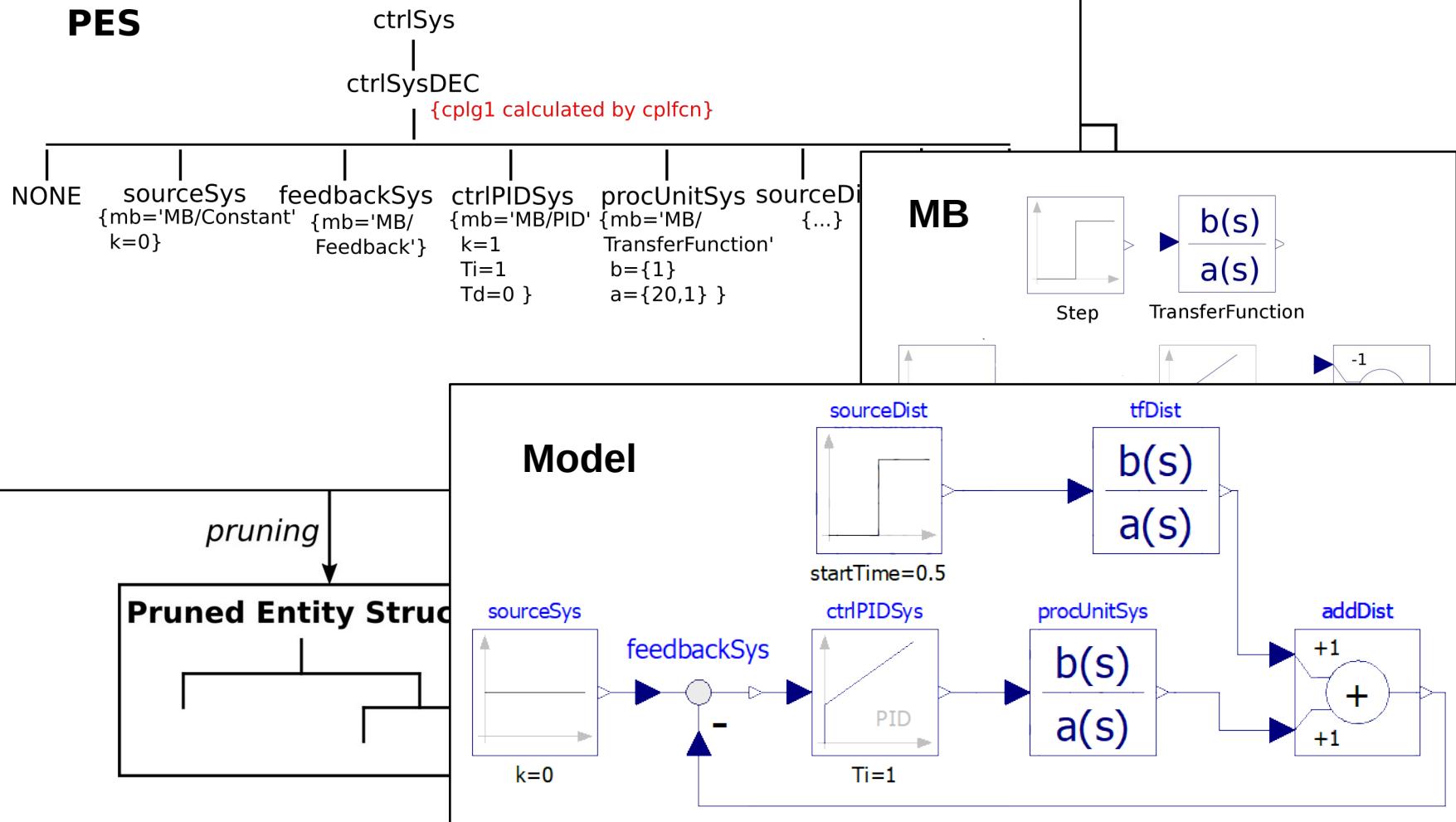


Model Selection and Generation of Variant #1 (3)





Model Selection and Generation of Variant #1 (4)





Model Selection and Generation of Variant #2

SES SESVAR={feedforward}

SemanticCondition={feedforward in [0,1]}

ctrlSys

ctrlSysDEC

{cplg1}

feedforwardCtrl sourceSys feedbackSys ctrlPIDSys procUnitSys sourceDist tfDist addDist

|| {mb='MB/Constant'

k=0}

{mb='MB/

Feedback'}

{mb='MB/PID'

k=1

{mb='MB/

TransferFunction'

{...}

{...}

{...}

feedforwardCtrlSPEC

{specrule:

feedforward == 0→NONE

feedforward == 1→fc }

NONE

fc

fcDEC

{cplg2}

addFeedforward

{mb='MB/Add'}

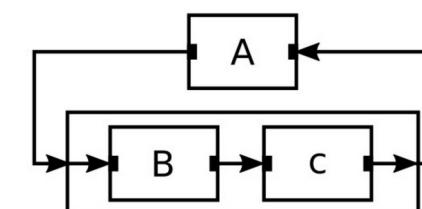
tfFeedforward

{mb='MB/TransferFunction'}

Pruned Entity Structure

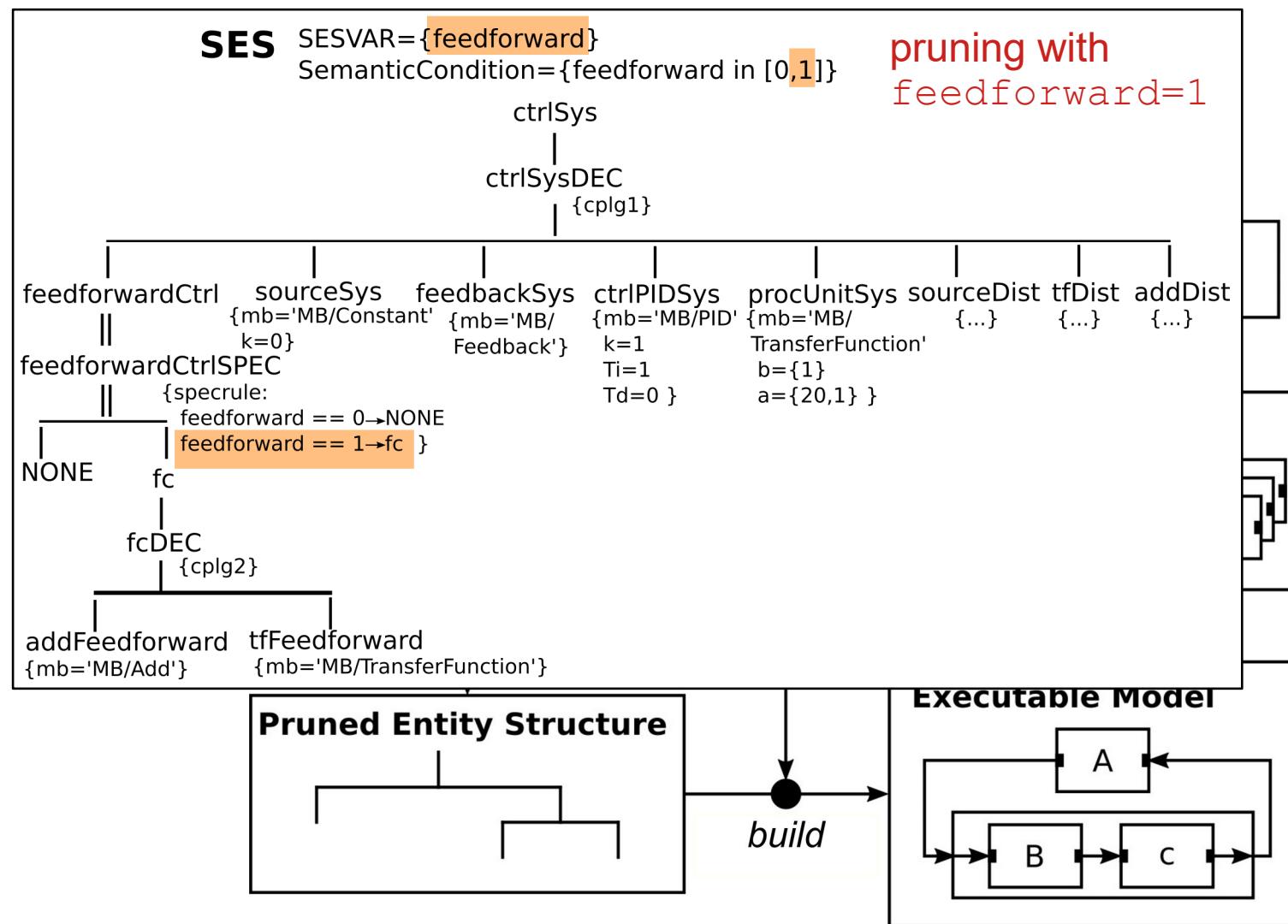
build

Executable Model



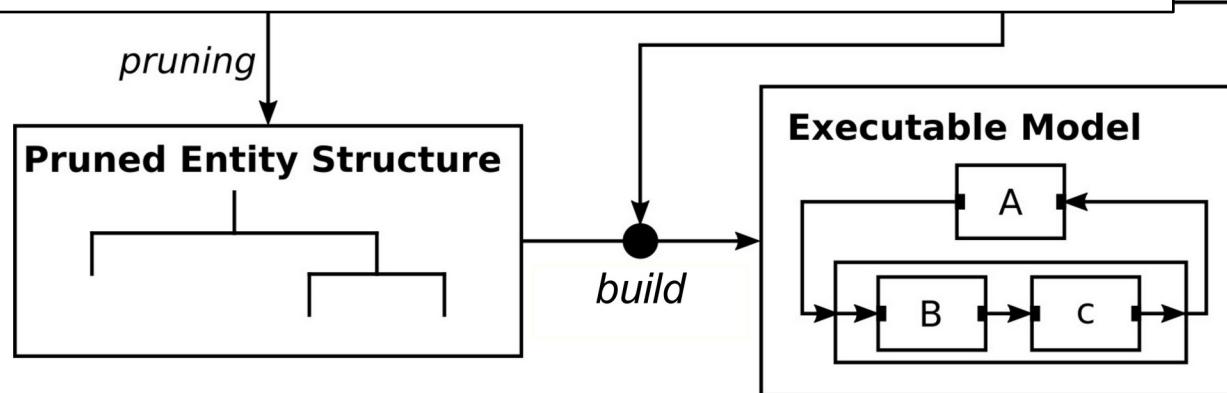
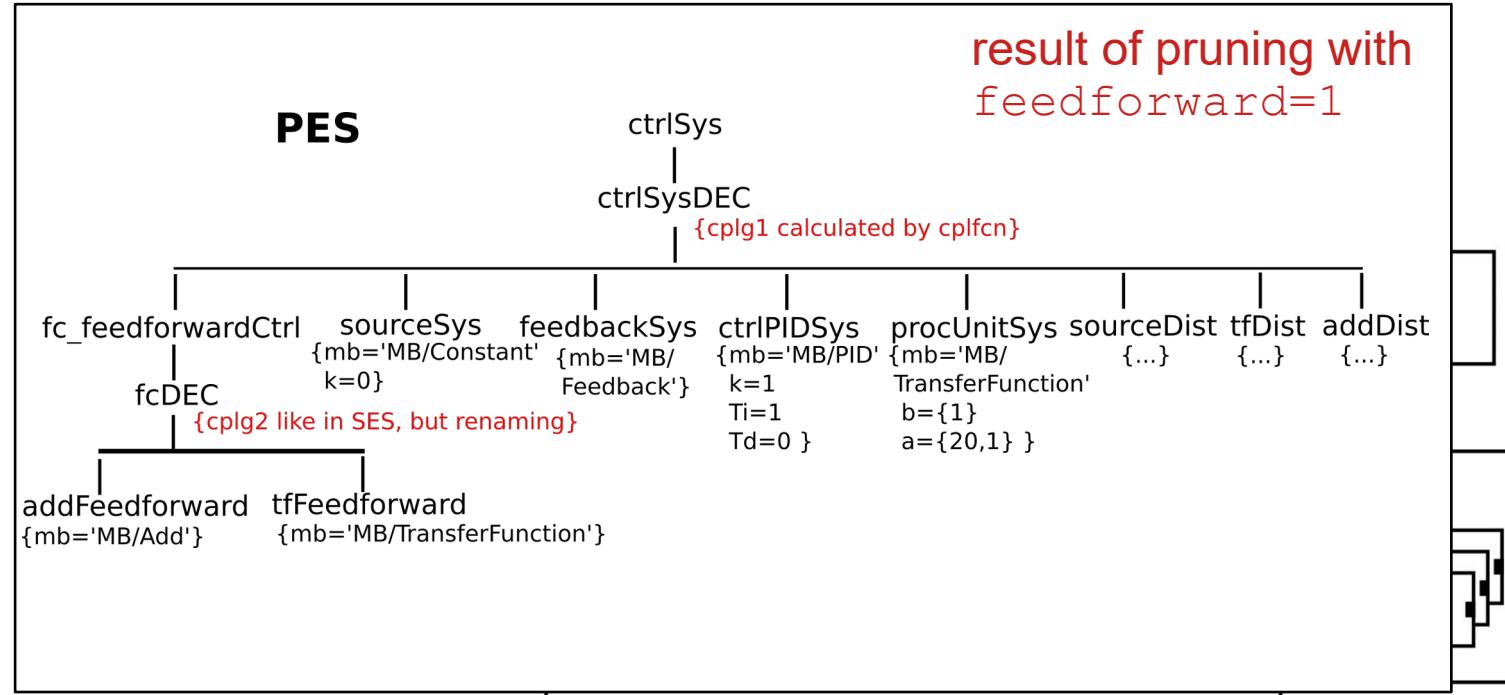


Model Selection and Generation of Variant #2



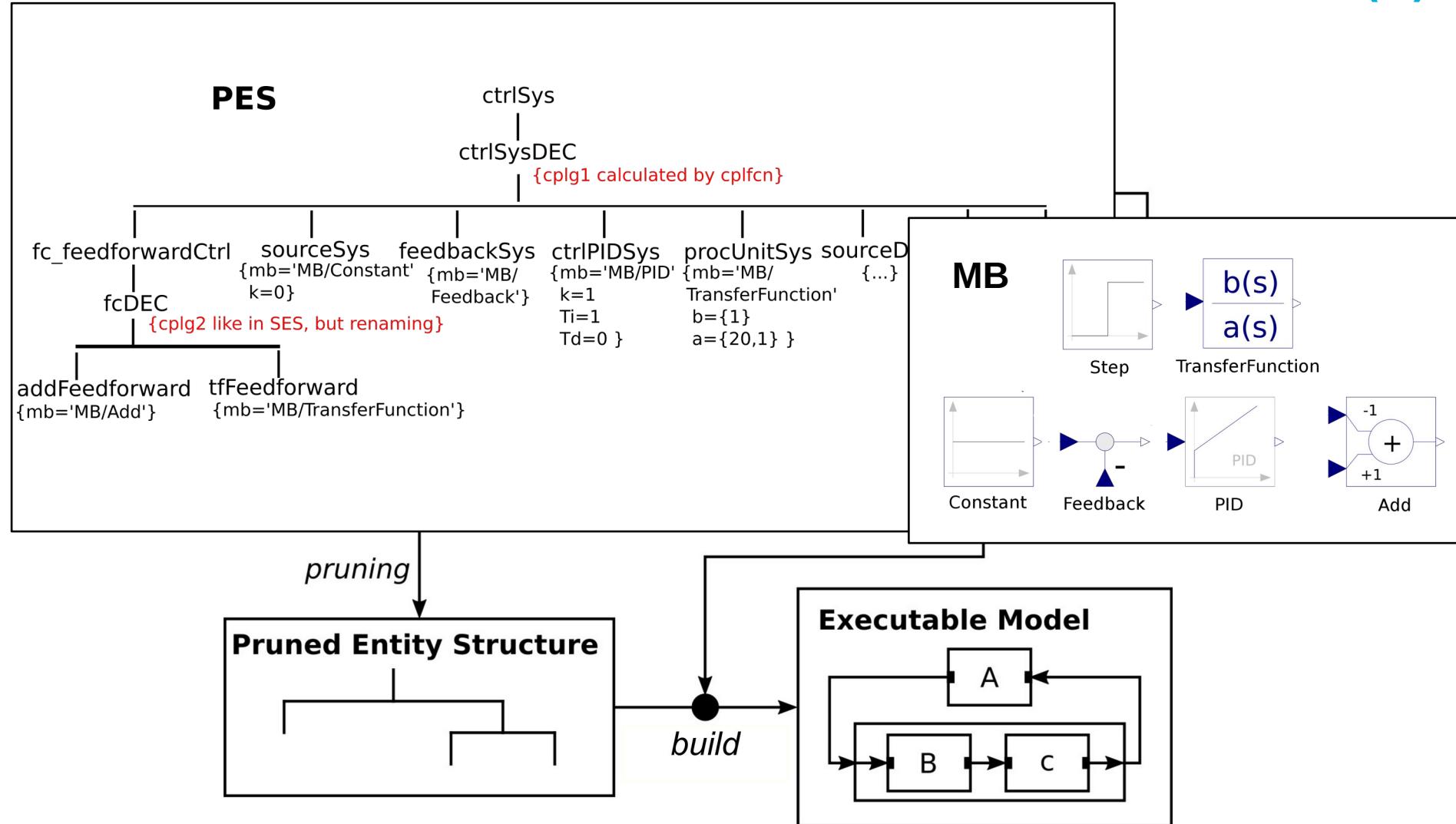


Model Selection and Generation of Variant #2 (2)



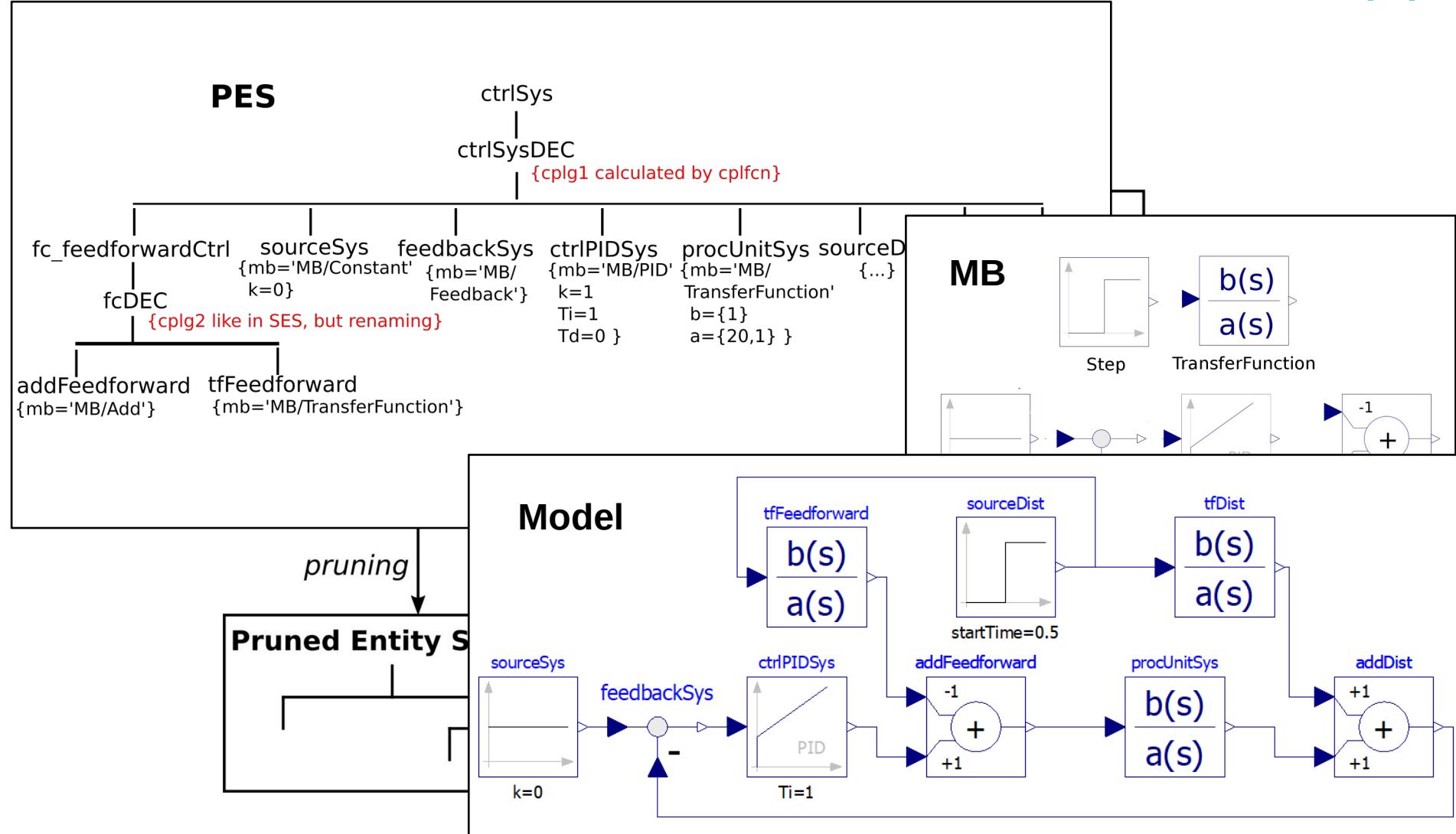


Model Selection and Generation of Variant #2 (3)



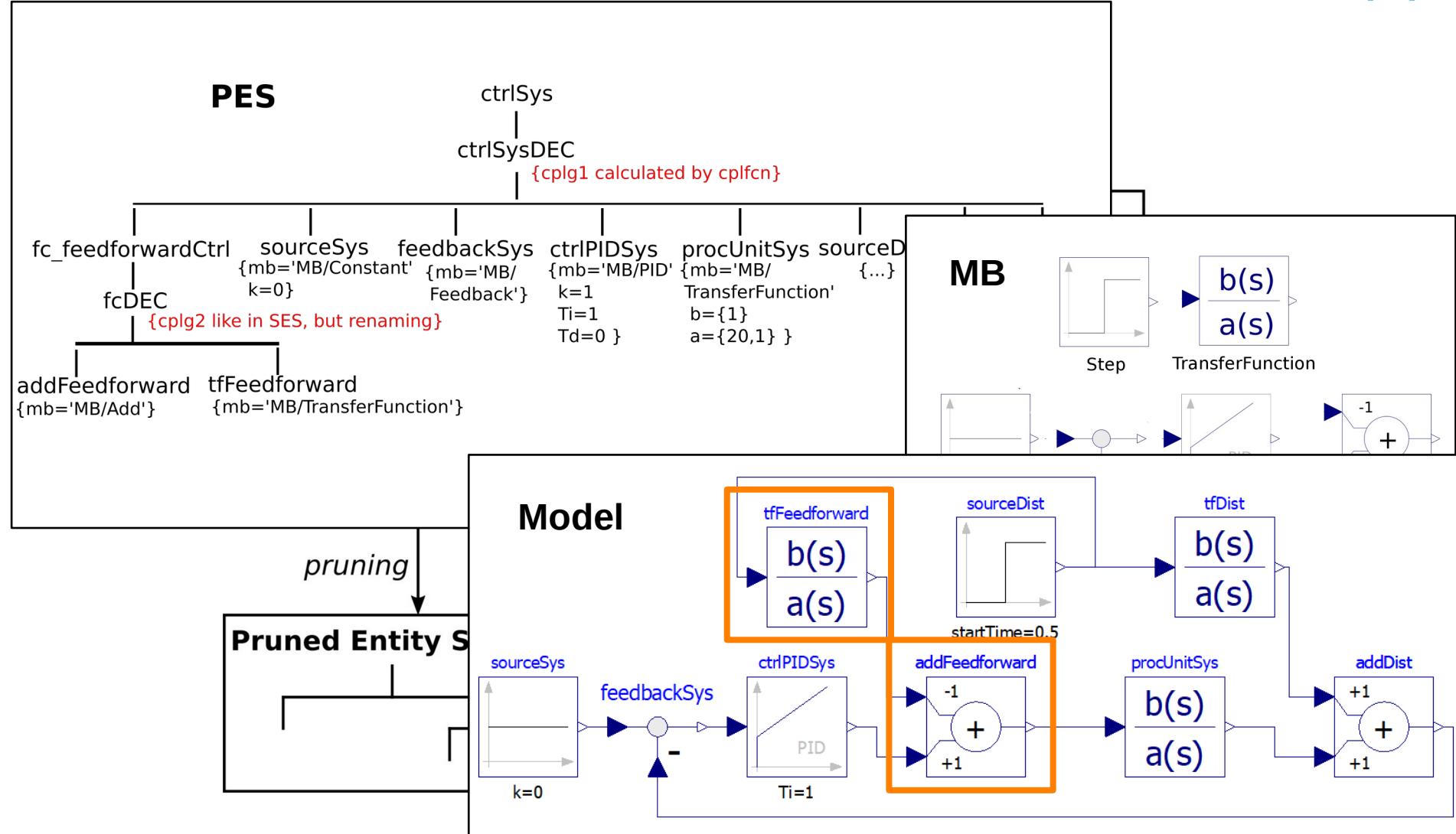


Model Selection and Generation of Variant #2 (4)





Model Selection and Generation of Variant #2 (4)





Python Toolset

- Available: <https://github.com/hendrikfolkerts>
- Tools
 - SESToPy → SES editor and IDE
 - SESViewEl → SES tree viewer
 - **SESMoPy** → Model builder
 - OpenModelica
 - Dymola
 - Simulink
 - SESEuPy
 - SESEcPy



Demonstration of SESMoPy (case study)

- Show provisional Experimental Frame from SESMoPy examples → `Template_for_SESMoPy`
- Show that different simulators can be set → `here` OpenModelica
- Show that two interfaces can be set → `here` native
- Merge Feedback SES from SESToPy examples to `simModel` → rename `simModel` to `ctrlSys` for merging.
- Show that configurations can be set in `expMethod`
- Prune for `feedforward=0` → dynamic couplings to static couplings
- Flatten for `feedforward=0` and save the FPES as file → explanation flattening: remove inner, coupled components → root node and leaves stay in tree → couplings recalculated
- (Prune for `feedforward=1` to show)
- Show the OpenModelica MB `MB.mo` and copy it in the same directory to the FPES file
- Open SESMoPy GUI → set FPES → create model → models for both configurations created
- Open one created model in OpenModelica and load MB file
- In OpenModelica open the model by double clicking
- Execute simulation → set simulation time to 50 seconds
 - Signals of interest: `sourceSys.y` `sourceDist.y` `addDist.y`
 - If the signals do not show up in plot: Click Auto Scale and Fit in View in plot
- If design objectives are not met with this structure and parameterization → later how to simulate automatically to find fitting structure and parameterization



Outline

1. Introduction
2. The case study
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. **The free Matlab SES toolbox** (C. Deacu)
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
9. Conclusion



The Free Matlab SES Toolbox

- Available: <https://github.com/cea-wismar>
- **SES Tbx Matlab
(available as App)**
 - SES editor
 - Function suite to work on SES
(merge, check, prune, flatten,...)
 - Model builder
- **SESViewEl**
 - SES tree viewer can be connected



The Free Matlab SES Toolbox – Model Builder

- To build & simulate models
 - is rather a build AND execution unit.
- Works with:
 - **Simulink** (internal calls or via script),
 - OpenModelica,
 - Dymola,
 - (soon MatlabDEVS).
- No graphical interface.



Demonstration of the SES Tbx Matlab with SESViewEI (case study)

- Basic functionality like in SESToPy.
- Two main differences compared to SESToPy:
 - SES wide settings right, node specific settings left side
 - Every change needs to be confirmed. ✓
- Load Feedback.mat from
SES-Tbx-root/more_examples/FeedbackControl
- Show SES in SESViewEI.
- Pruning for feedforward=0 and feedforward=1.
- Model building would work, but there is no accessible output.



Demonstration of the SES Tbx Matlab Model Generation (case study)

- For Simulink Out nodes are necessary:
 - add manually after model generation or
 - **add to the SES** before.
- Load adapted SES Feedback_with_outputs.mat
- Show the Simulink MB.
- Demonstration for Simulink with a function
 - Write a script or function where options for model and simulation are set (`build_simulate_feedback_with_outputs.m`).
 - File defines options for and then calls the model builder (`modelBuilder.m`).
 - Execute with `feedforward=1`, show model and output.

One of the challenges to keep the SES independent from the simulator!



The free Matlab SES Tbx – Conclusion

- Provides comprehensive functionality to design and manipulate SES.
- Model building works for Simulink, OpenModelica, Dymola, and (soon) MatlabDEVS.
- Model builder builds **and** executes model.
- Some challenges to keep the SES independent from the simulator.
- Automation of simulation experiments is possible!



Outline

1. Introduction
2. The case study
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. The free Matlab SES toolbox
- 7. Organization of a simulator-independent MB**
(H. Folkerts)
8. Full automation of simulation experiments
9. Conclusion



Model Building – Support Different Simulators

Facts / Challenges



Model Building – Support Different Simulators

Facts / Challenges

- System models should be tested with different simulators
 - Simulators are domain specific
 - Simulator correctness



Model Building – Support Different Simulators

Facts / Challenges

- System models should be tested with different simulators
 - Simulators are domain specific
 - Simulator correctness
- **SES is independent of simulator**



Model Building – Support Different Simulators

Facts / Challenges

- System models should be tested with different simulators
 - Simulators are domain specific
 - Simulator correctness
- **SES is independent of simulator**
- Native model building needs **one** MB for **each** simulator
 - Error prone and costly to maintain



Model Building – Support Different Simulators

Facts / Challenges

- System models should be tested with different simulators
 - Simulators are domain specific
 - Simulator correctness
- **SES is independent of simulator**
- Native model building needs **one MB** for **each** simulator
 - Error prone and costly to maintain
- Native model building needs **specific model builders**, because simulators use:
 - Different portnames / porttypes
 - Different (block) parameters / parameter names
 - Different parameter meaning
 - Different syntax



Model Building – Support Different Simulators

Facts / Challenges

- System models should be tested with different simulators
 - Simulators are domain specific
 - Simulator correctness
- **SES is independent of simulator**
- Native model building needs **one** MB for **each** simulator
 - Error prone and costly to maintain
- Native model building needs **specific model builders**, because simulators use:
 - Different portnames / porttypes
 - Different (block) parameters / parameter names
 - Different parameter meaning
 - Different syntax
- **Goal: One** (simple) MB and model builder for **all** simulators



Functional Mock-up Interface (FMI)^{1,2}

- ¹Blochwitz et al. (2011) „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“. Proc. of the 8th Modelica Conference, Dresden.
- ²Blochwitz et al. (2012) „Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. Proc. of the 9th Modelica Conference, Munich.



Functional Mock-up Interface (FMI) ^{1,2}

- FMI defines a standardized interface of components (models, blocks)

¹Blochwitz et al. (2011) „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“. Proc. of the 8th Modelica Conference, Dresden.

²Blochwitz et al. (2012) „Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. Proc. of the 9th Modelica Conference, Munich.



Functional Mock-up Interface (FMI) ^{1,2}

- FMI defines a standardized interface of components (models, blocks)
 - Interface is based on C code or binaries (know-how protection)

¹Blochwitz et al. (2011) „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“. Proc. of the 8th Modelica Conference, Dresden.

²Blochwitz et al. (2012) „Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. Proc. of the 9th Modelica Conference, Munich.



Functional Mock-up Interface (FMI) ^{1,2}

- FMI defines a standardized interface of components (models, blocks)
 - Interface is based on C code or binaries (know-how protection)
 - Many simulators support FMI

¹Blochwitz et al. (2011) „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“. Proc. of the 8th Modelica Conference, Dresden.

²Blochwitz et al. (2012) „Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. Proc. of the 9th Modelica Conference, Munich.



Functional Mock-up Interface (FMI)^{1,2}

- FMI defines a standardized interface of components (models, blocks)
 - Interface is based on C code or binaries (know-how protection)
 - Many simulators support FMI
 - Reuse of components
 - (i) For **model exchange**
 - (ii) For co-simulation

¹Blochwitz et al. (2011) „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“. Proc. of the 8th Modelica Conference, Dresden.

²Blochwitz et al. (2012) „Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. Proc. of the 9th Modelica Conference, Munich.



Functional Mock-up Interface (FMI)^{1,2}

- FMI defines a standardized interface of components (models, blocks)
 - Interface is based on C code or binaries (know-how protection)
 - Many simulators support FMI
 - Reuse of components
 - (i) For **model exchange**
 - (ii) For co-simulation
- **FMI for model exchange** defines an interface for the simulation environment to generate C code of a dynamic system model
(still problems for discrete event models)

¹Blochwitz et al. (2011) „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“. Proc. of the 8th Modelica Conference, Dresden.

²Blochwitz et al. (2012) „Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. Proc. of the 9th Modelica Conference, Munich.



Functional Mock-up Unit (FMU)



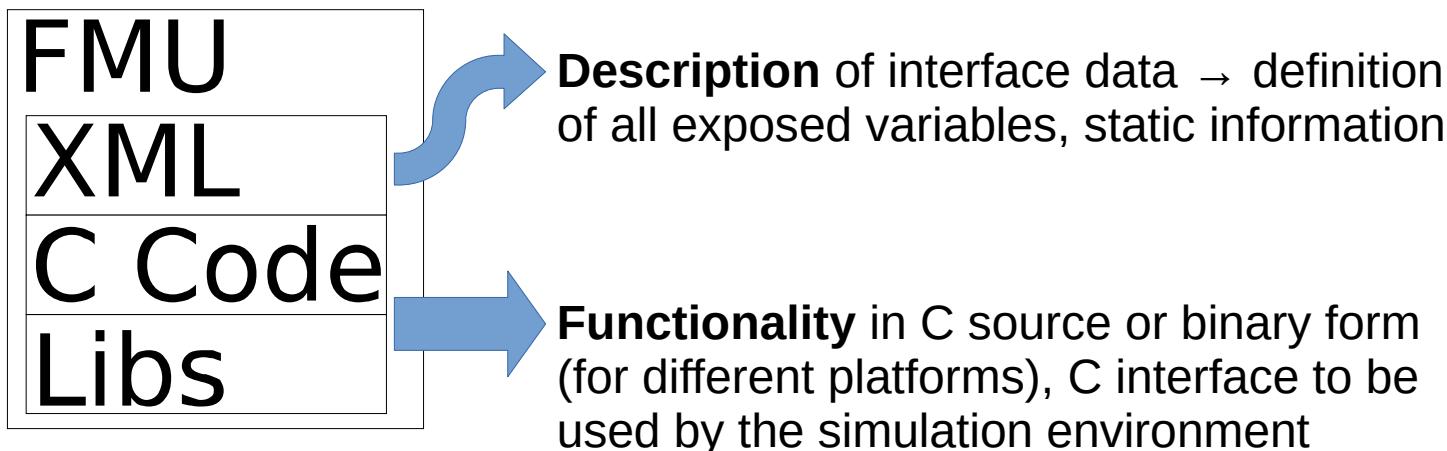
Functional Mock-up Unit (FMU)

- Block = Functional Mock-up Unit (FMU) → zipped file, file extension .fmu
- FMU implements the FMI



Functional Mock-up Unit (FMU)

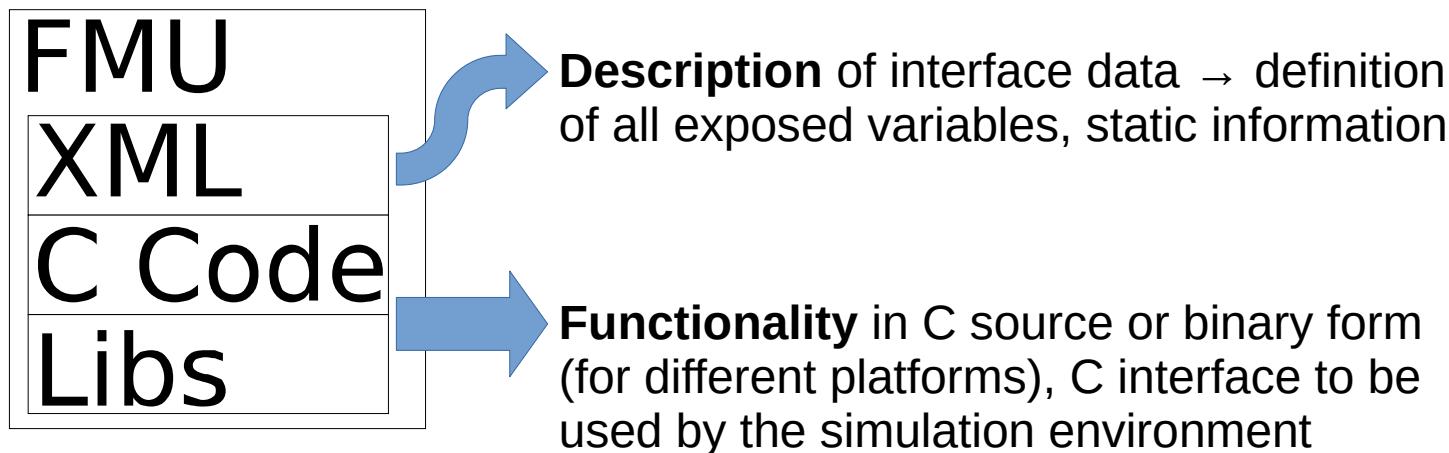
- Block = Functional Mock-up Unit (FMU) → zipped file, fileextension .fmu
- FMU implements the FMI



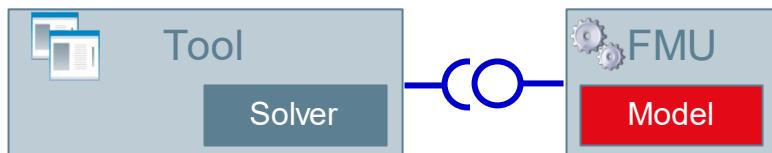


Functional Mock-up Unit (FMU)

- Block = Functional Mock-up Unit (FMU) → zipped file, fileextension .fmu
- FMU implements the FMI



Model Exchange





Model Building – Support Different Simulators



Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)



Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**



Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator



Model Building – Support Different Simulators

- **Using OpenModelica and FMI for model exchange (export whole model as FMU)**
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- **One MB for all simulators**





Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters





Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters



Current development state



Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters



Current development state

- Using OpenModelica and FMI for model exchange (**export block FMUs**)



Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters



Current development state

- Using OpenModelica and FMI for model exchange (**export block FMUs**)
 - Define an MB for OpenModelica and/or export FMUs from any simulator, import and configure blocks in OpenModelica, **export each configured block as FMU**



Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters



Current development state

- Using OpenModelica and FMI for model exchange (**export block FMUs**)
 - Define an MB for OpenModelica and/or export FMUs from any simulator, import and configure blocks in OpenModelica, **export each configured block as FMU**
 - Import and connect preconfigured FMUs in the target simulator



Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters



Current development state

- Using OpenModelica and FMI for model exchange (**export block FMUs**)
 - Define an MB for OpenModelica and/or export FMUs from any simulator, import and configure blocks in OpenModelica, **export each configured block as FMU**
 - Import and connect preconfigured FMUs in the target simulator
- One MB for all simulators





Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters



Current development state

- Using OpenModelica and FMI for model exchange (**export block FMUs**)
 - Define an MB for OpenModelica and/or export FMUs from any simulator, import and configure blocks in OpenModelica, **export each configured block as FMU**
 - Import and connect preconfigured FMUs in the target simulator
- One MB for all simulators
- Possibilty to change structure of the model





Model Building – Support Different Simulators

- Using OpenModelica and FMI for model exchange (**export whole model as FMU**)
 - Define an MB for OpenModelica and/or **export FMUs from any simulator**, build and configure OpenModelica model, finally **export the whole model as an FMU**
 - Import model FMU in the target simulator
- One MB for all simulators
- No possibility to change structure of model built as FMU, only parameters



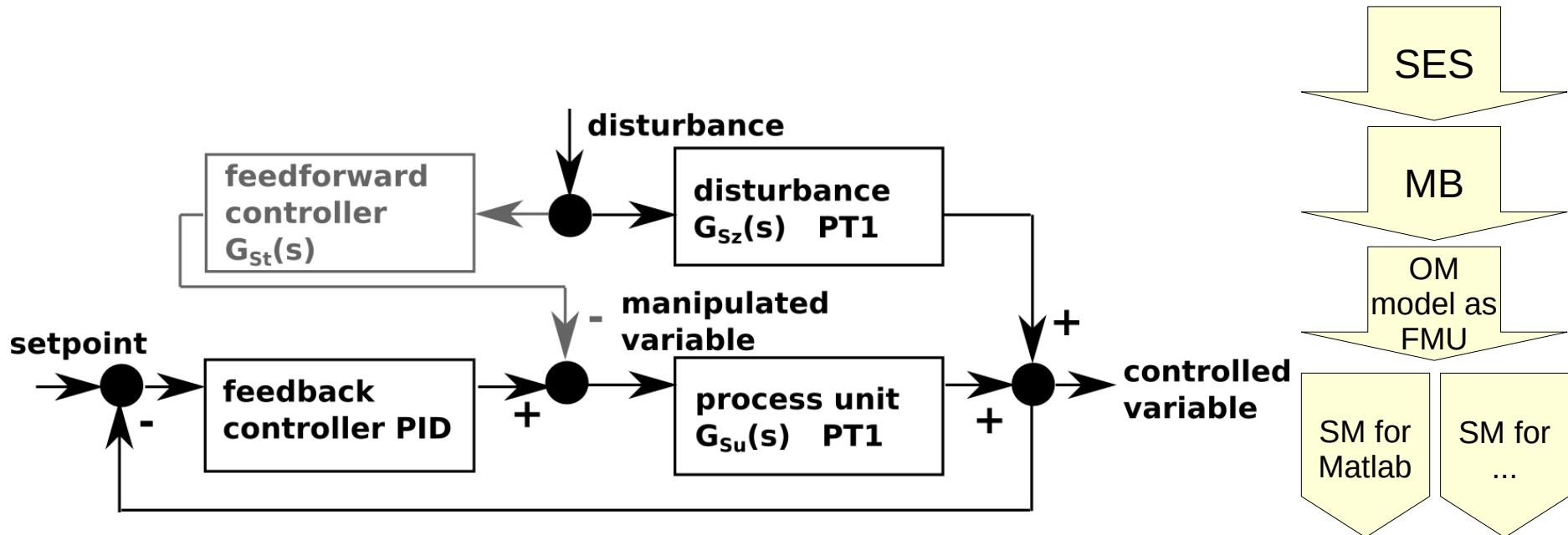
Current development state

- Using OpenModelica and FMI for model exchange (**export block FMUs**)
 - Define an MB for OpenModelica and/or export FMUs from any simulator, import and configure blocks in OpenModelica, **export each configured block as FMU**
 - Import and connect preconfigured FMUs in the target simulator
- One MB for all simulators
- Possibilty to change structure of the model
- Model building VERY slow → parallel processing?



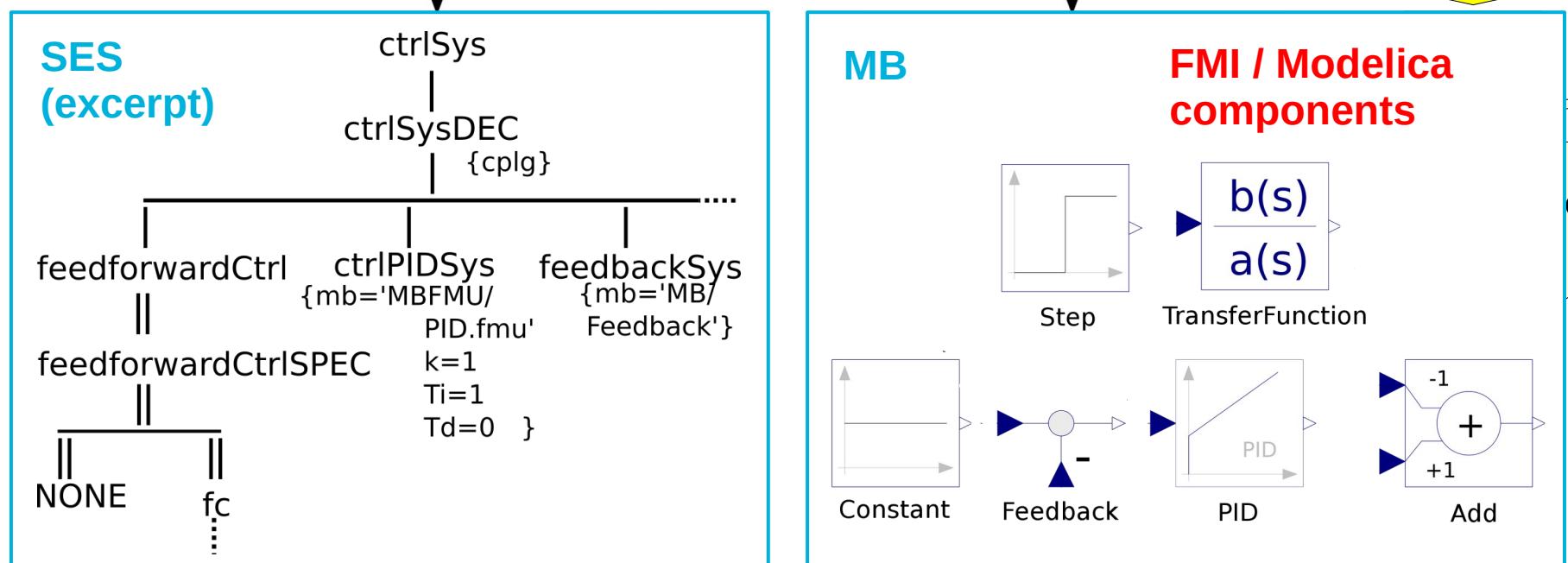
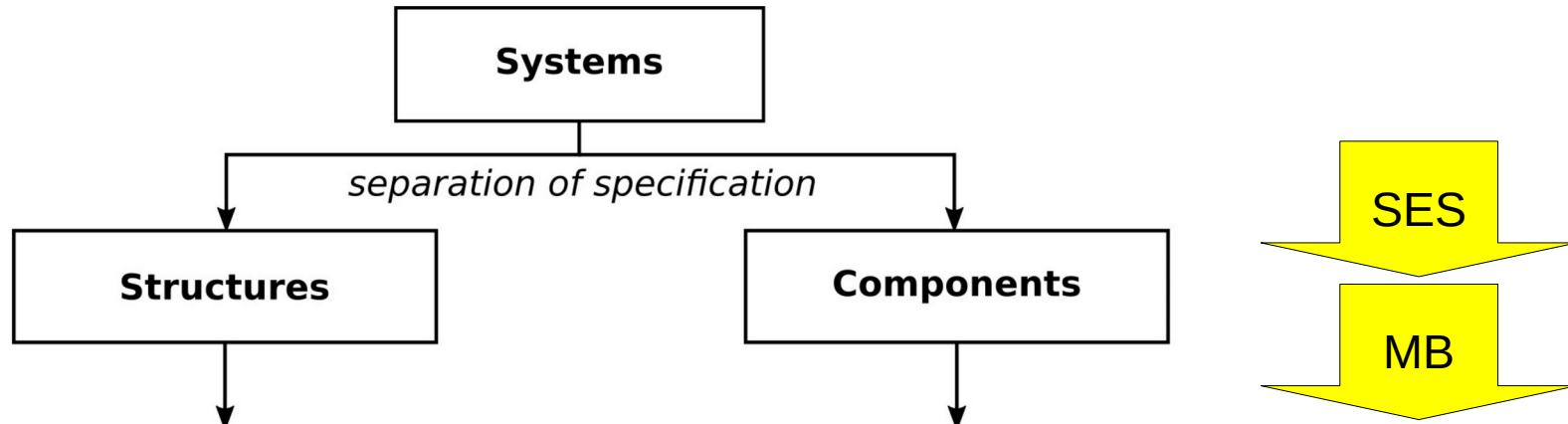


Case Study with FMI



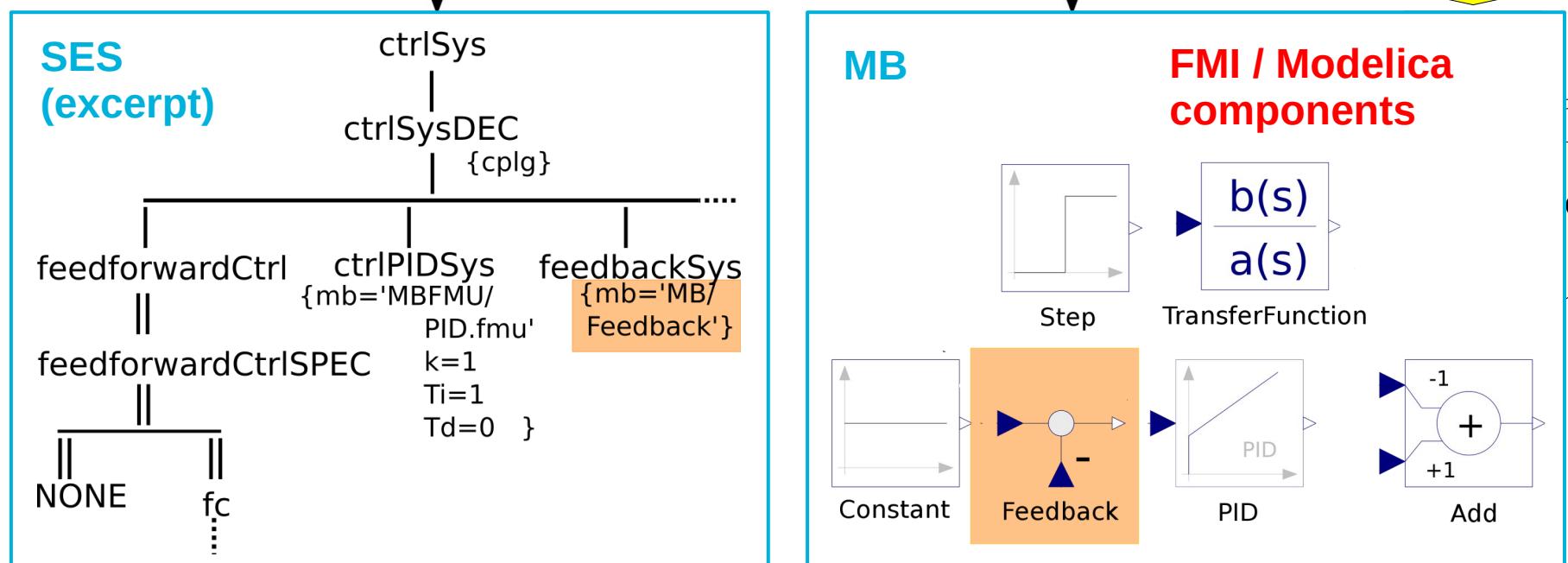
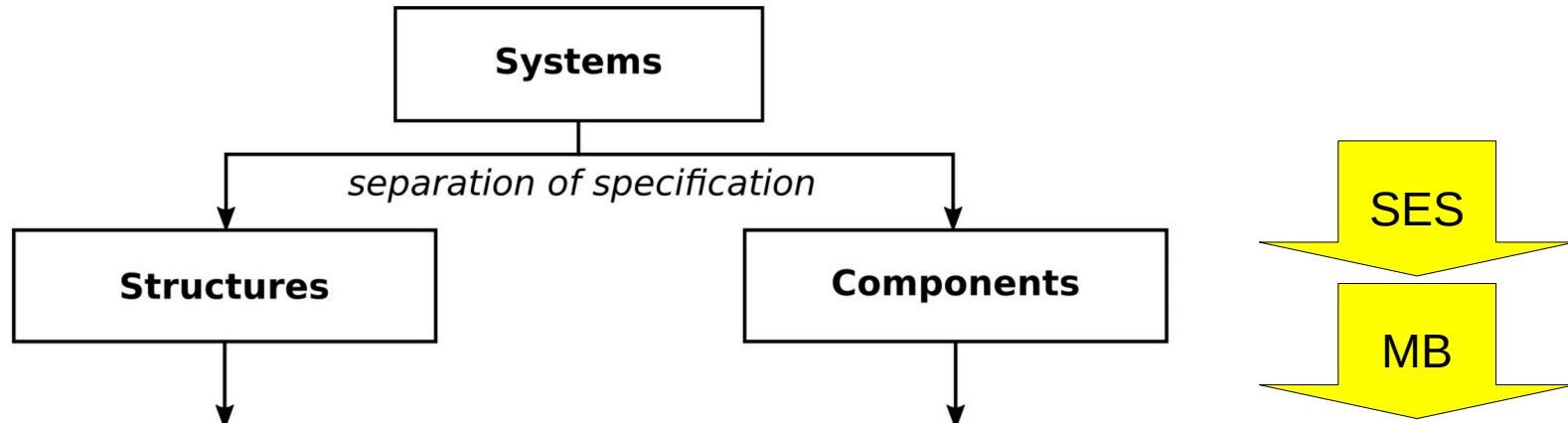


SES/MB Approach with FMI Components



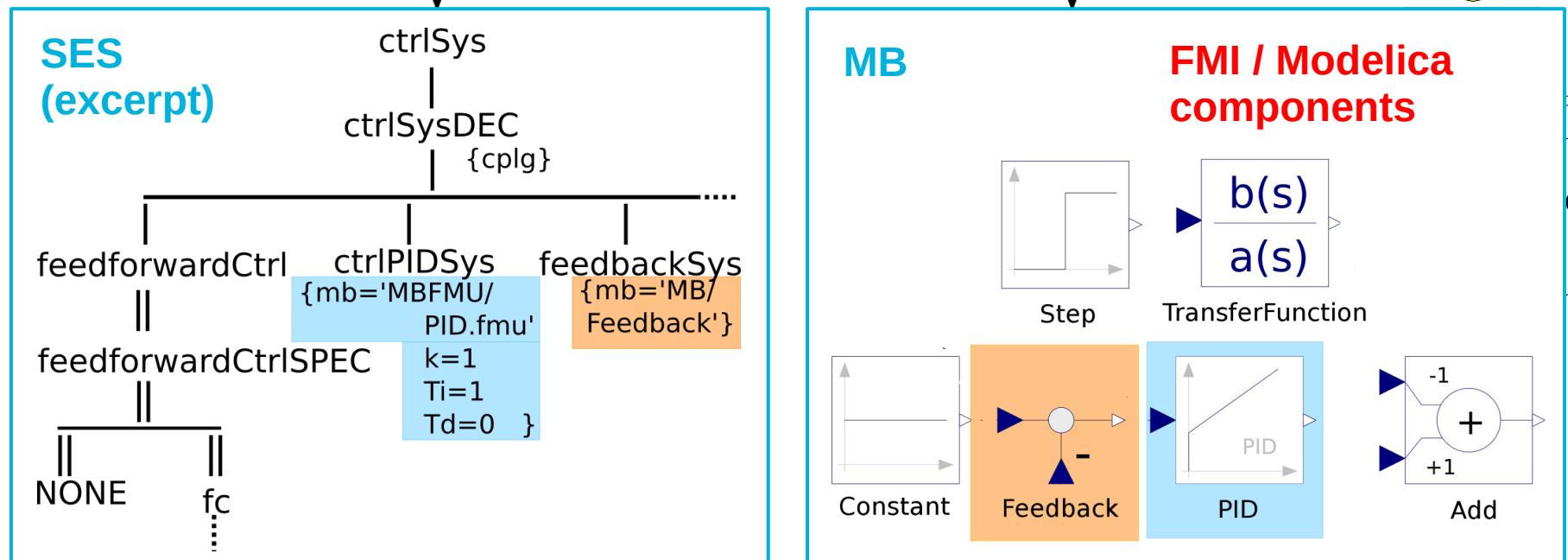
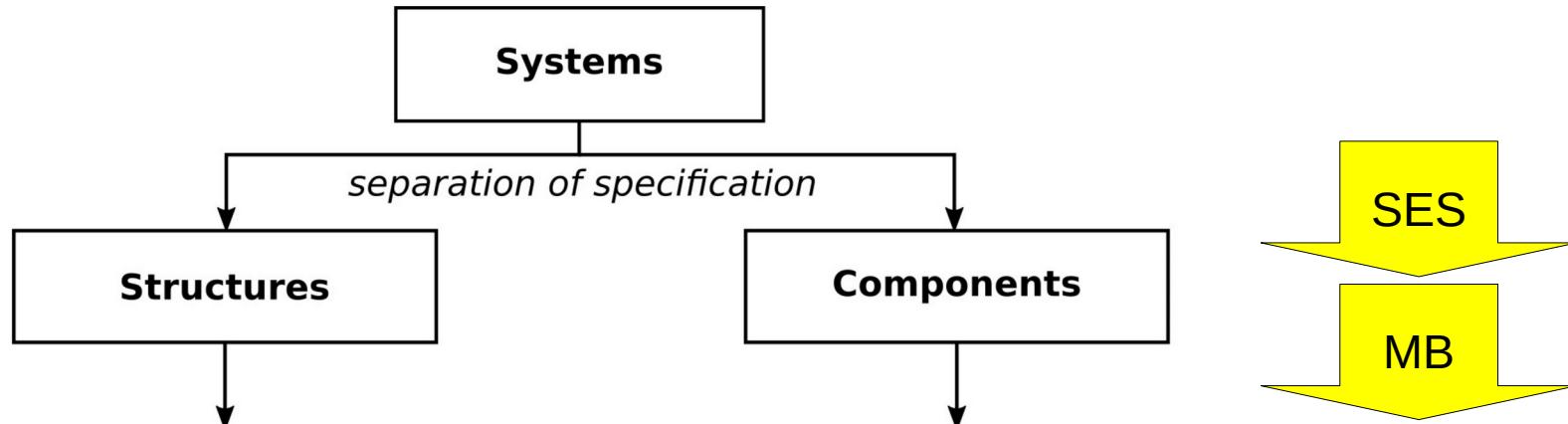


SES/MB Approach with FMI Components



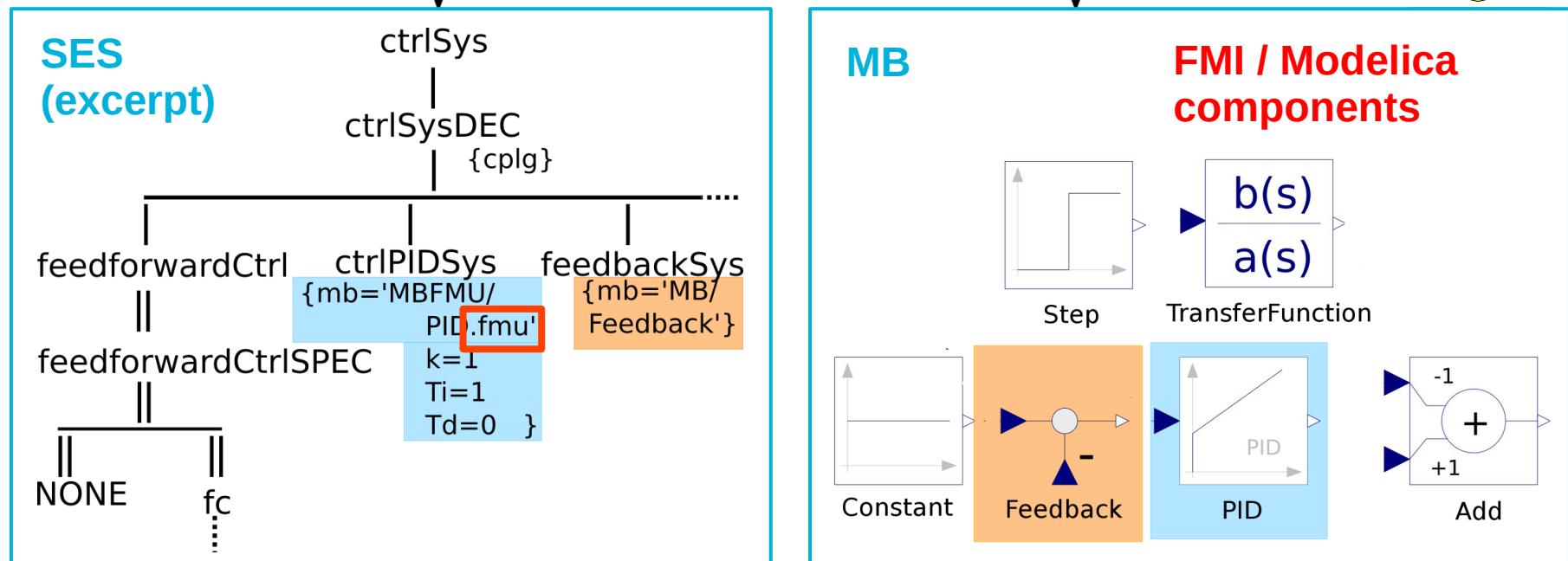
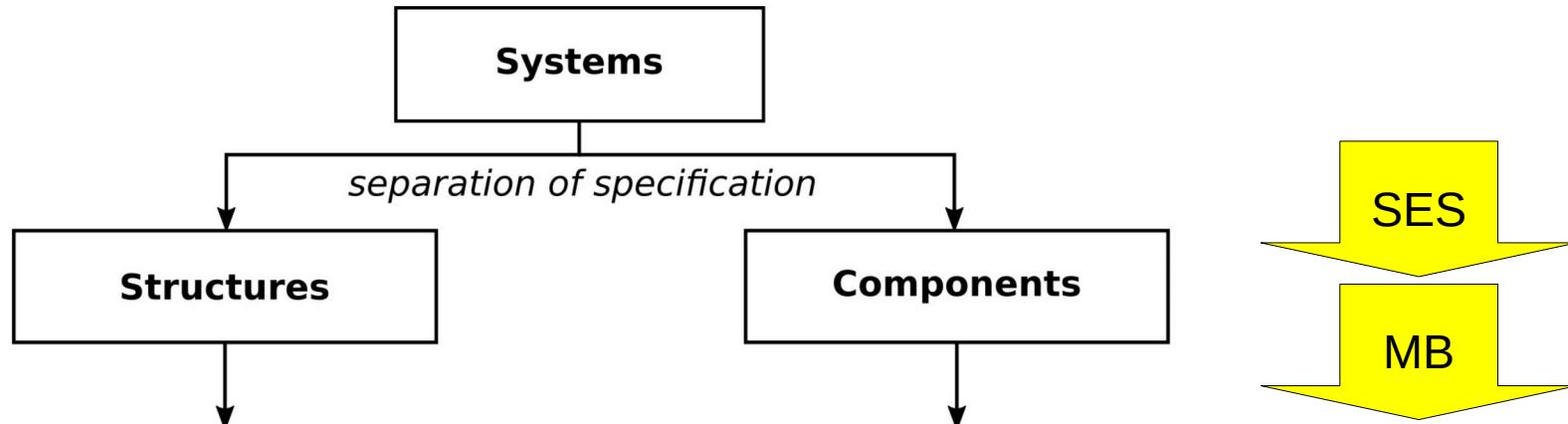


SES/MB Approach with FMI Components



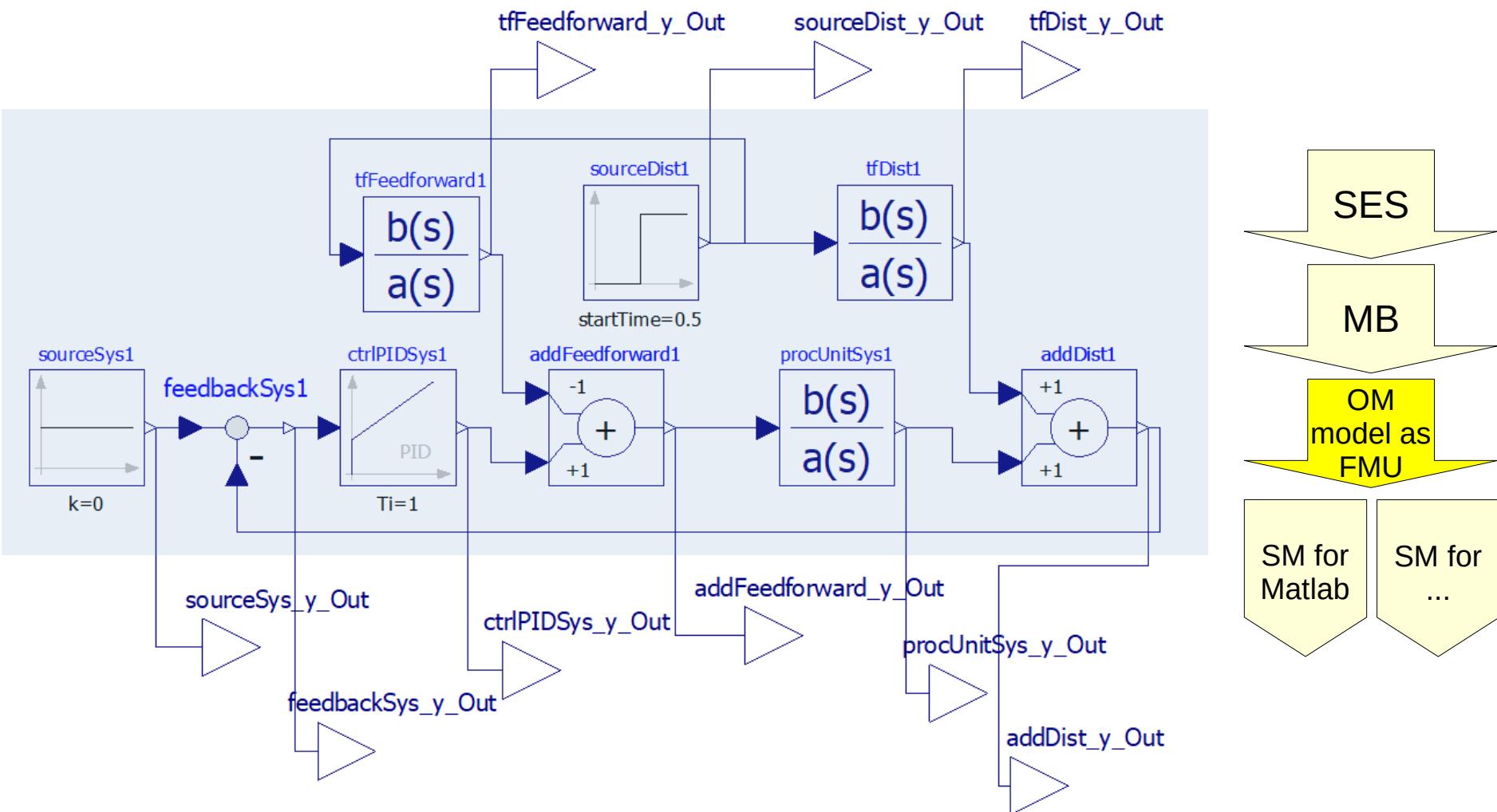


SES/MB Approach with FMI Components



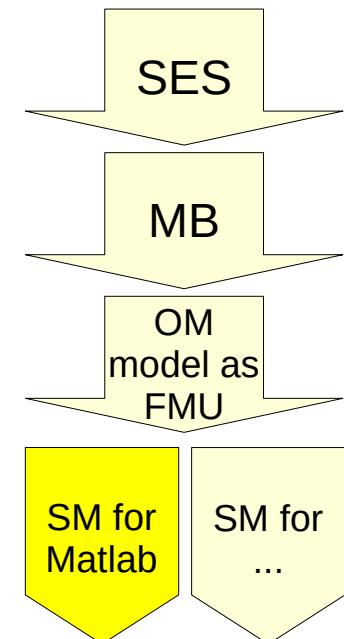


Generated & Configured OpenModelica Model / FMU





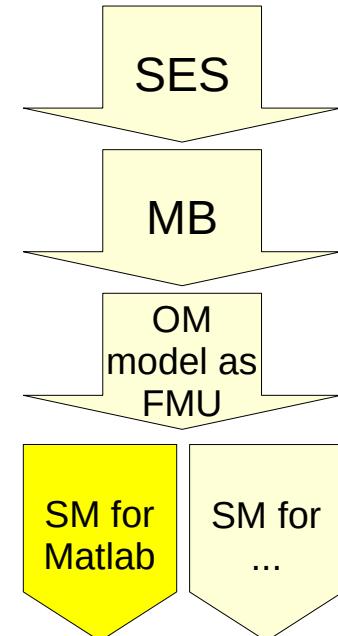
Case Study: FMU Imported in Matlab/Simulink





Case Study: FMU Imported in Matlab/Simulink

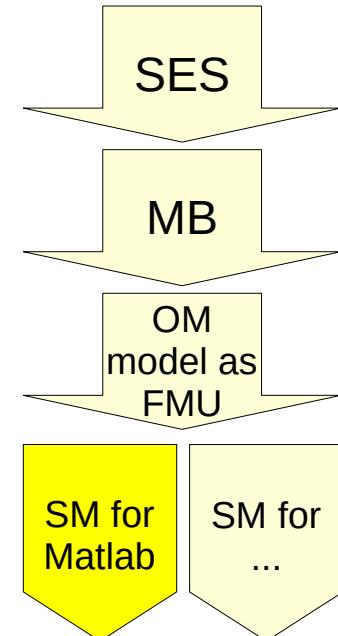
- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks





Case Study: FMU Imported in Matlab/Simulink

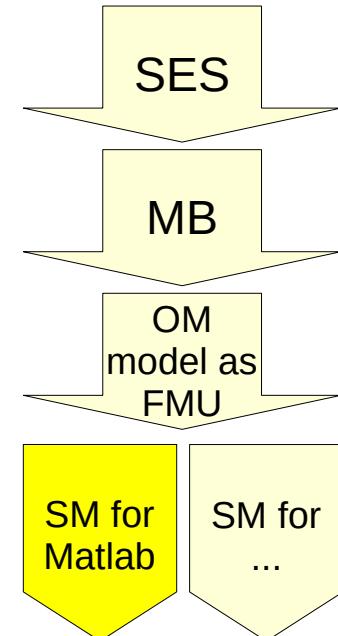
- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks
- A .m Matlab script is created with instructions on:





Case Study: FMU Imported in Matlab/Simulink

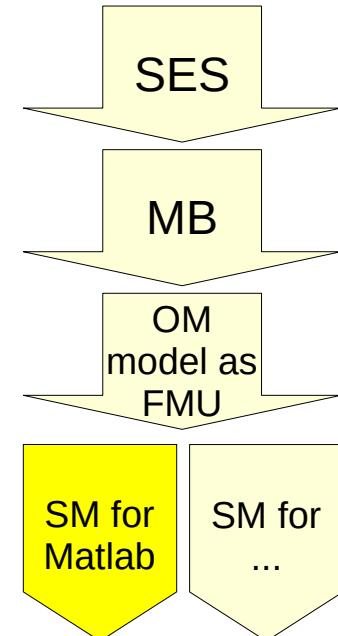
- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks
- A .m Matlab script is created with instructions on:
 - Create Simulink model





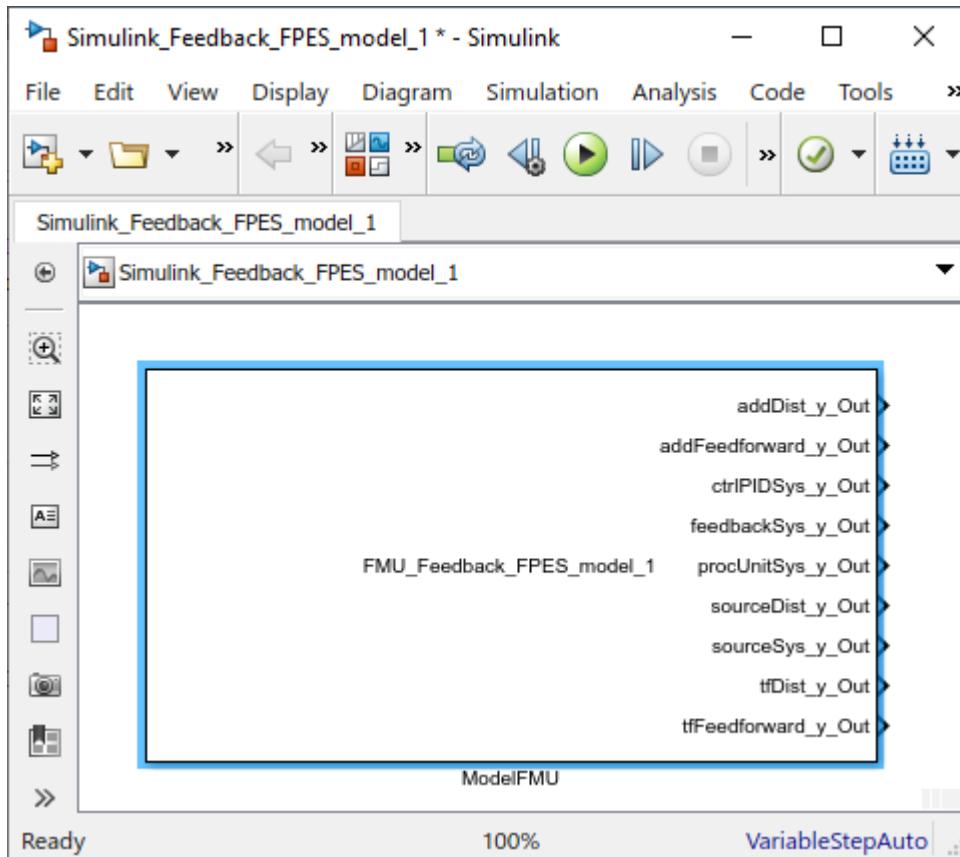
Case Study: FMU Imported in Matlab/Simulink

- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks
- A .m Matlab script is created with instructions on:
 - Create Simulink model
 - Add Simulink FMU block, configure with the model FMU
→ FMU is extracted



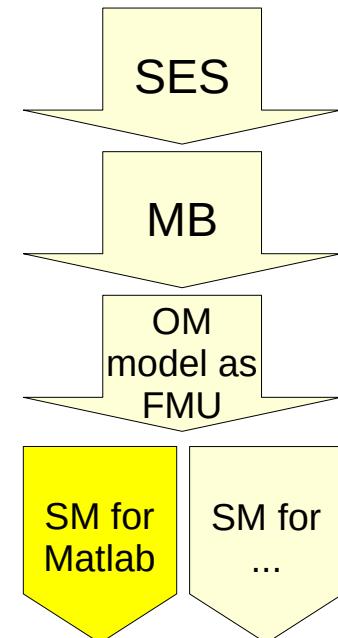


Case Study: FMU Imported in Matlab/Simulink



Simulink only
"t" block → "Out"
FMU, which in

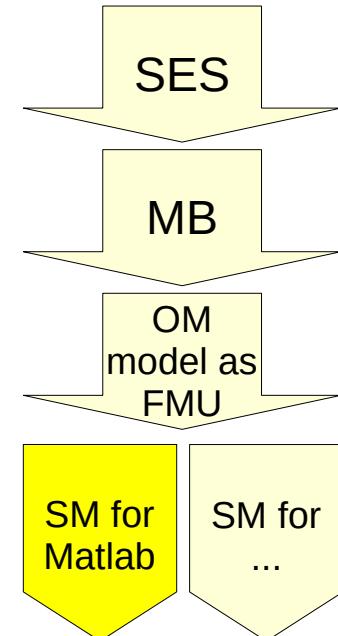
model FMU





Case Study: FMU Imported in Matlab/Simulink

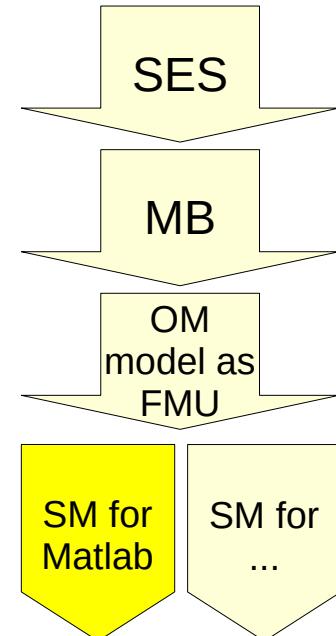
- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks
- A .m Matlab script is created with instructions on:
 - Create Simulink model
 - Add Simulink FMU block, configure with the model FMU
→ FMU is extracted





Case Study: FMU Imported in Matlab/Simulink

- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks
- A .m Matlab script is created with instructions on:
 - Create Simulink model
 - Add Simulink FMU block, configure with the model FMU
→ FMU is extracted
 - Create Submodel → Ports get names that can be searched for





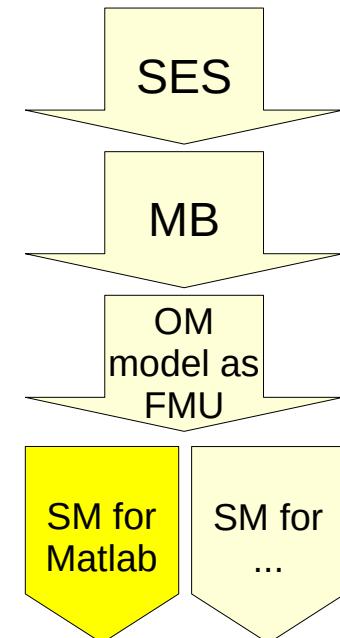
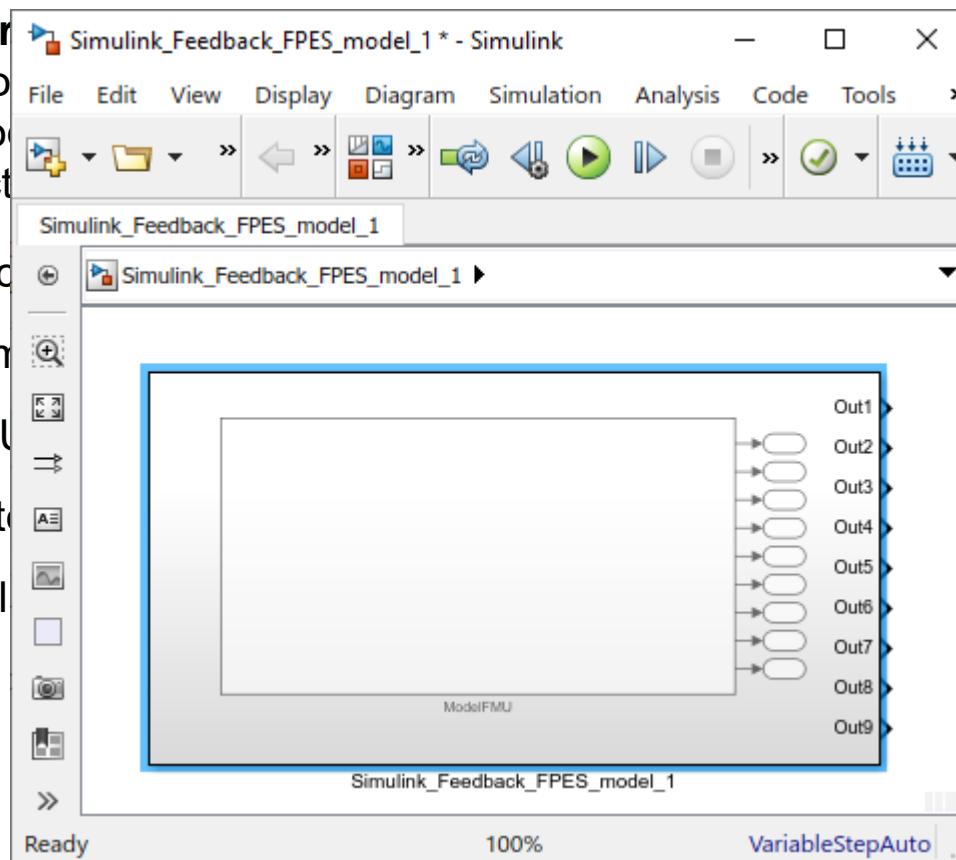
Case Study: FMU Imported in Matlab/Simulink

- **Why do we need port blocks?**

returns variables computed by the FMU
blocks added in Modelica
turn can be connected

- A .m Matlab script is created

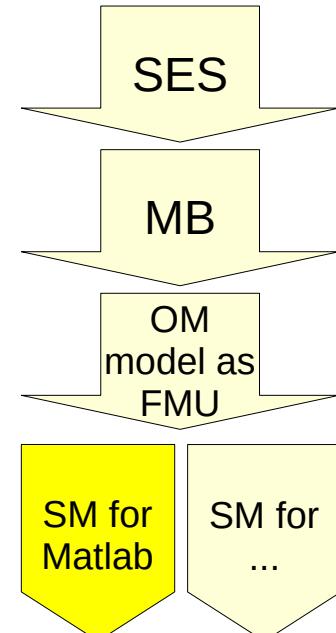
- Create Simulink model
- Add Simulink FMU block
- FMU is extracted
- Create Submodel for





Case Study: FMU Imported in Matlab/Simulink

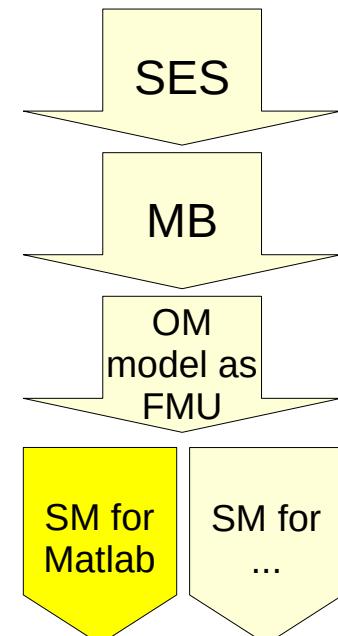
- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks
- A .m Matlab script is created with instructions on:
 - Create Simulink model
 - Add Simulink FMU block, configure with the model FMU
→ FMU is extracted
 - Create Submodel → Ports get names that can be searched for





Case Study: FMU Imported in Matlab/Simulink

- **Why do we need ports in the model FMU?** → Simulink only returns variables connected to an Simulink “Out” block → “Out” blocks added in Modelica before exporting the FMU, which in turn can be connected to Simulink “Out” blocks
- A .m Matlab script is created with instructions on:
 - Create Simulink model
 - Add Simulink FMU block, configure with the model FMU
→ FMU is extracted
 - Create Submodel → Ports get names that can be searched for
 - Rename the ports → There is an “outputs” section in the modelDescription.xml of the FMU





Case Study: FMU Imported in Matlab/Simulink

The screenshot shows the Simulink interface with the title bar "Simulink_Feedback_FPES_model_1 * - Simulink". The menu bar includes File, Edit, View, Display, Diagram, Simulation, Analysis, Code, Tools, and Help. The toolbar contains various icons for file operations, simulation, and analysis. The workspace window displays a "Simulink_Feedback_FPES_model_1" model. Inside the model, there is a block labeled "ModelFMU" which has several output ports: addDist_y_Out, addPst_y_Out, addFeedforward_y_Out, ctrlPIDSys_y_Out, feedbackSys_y_Out, procUnitSys_y_Out, sourceDist_y_Out, sourceSys_y_Out, tfDist_y_Out, and tfFeedforward_y_Out. To the right of the Simulink interface, a diagram illustrates the connection between the imported FMU and the host environment:

- A yellow box labeled "SES" is connected to a yellow box labeled "MB".
- "MB" is connected to a yellow box labeled "OM model as FMU".
- "OM model as FMU" is connected to two yellow hexagonal boxes labeled "SM for Matlab" and "SM for ...".

The status bar at the bottom indicates "Ready", "100%", and "VariableStepAuto".

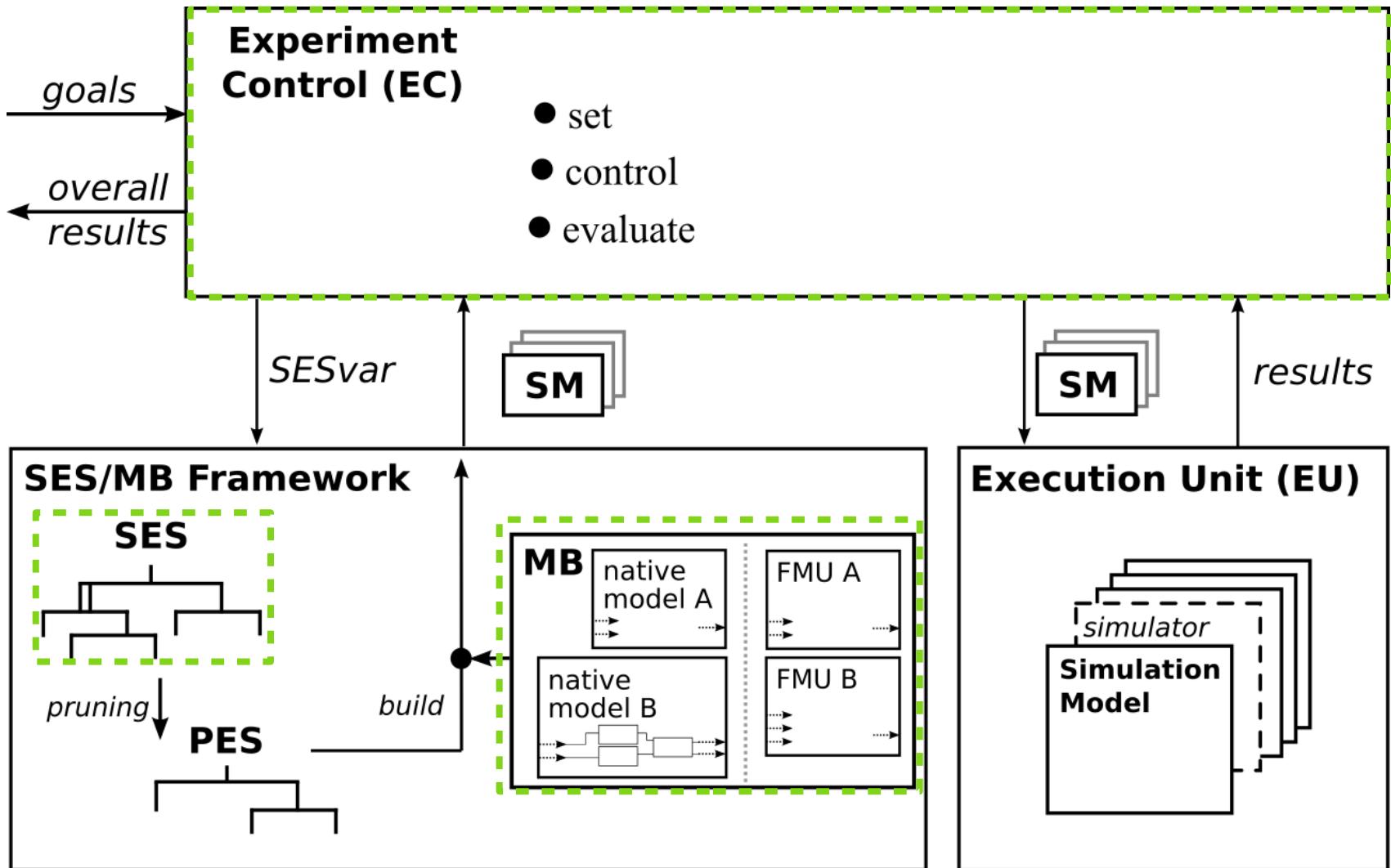


Outline

1. Introduction
2. The case study
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
- 8. Full automation of simulation experiments**
(H. Folkerts)
9. Conclusion

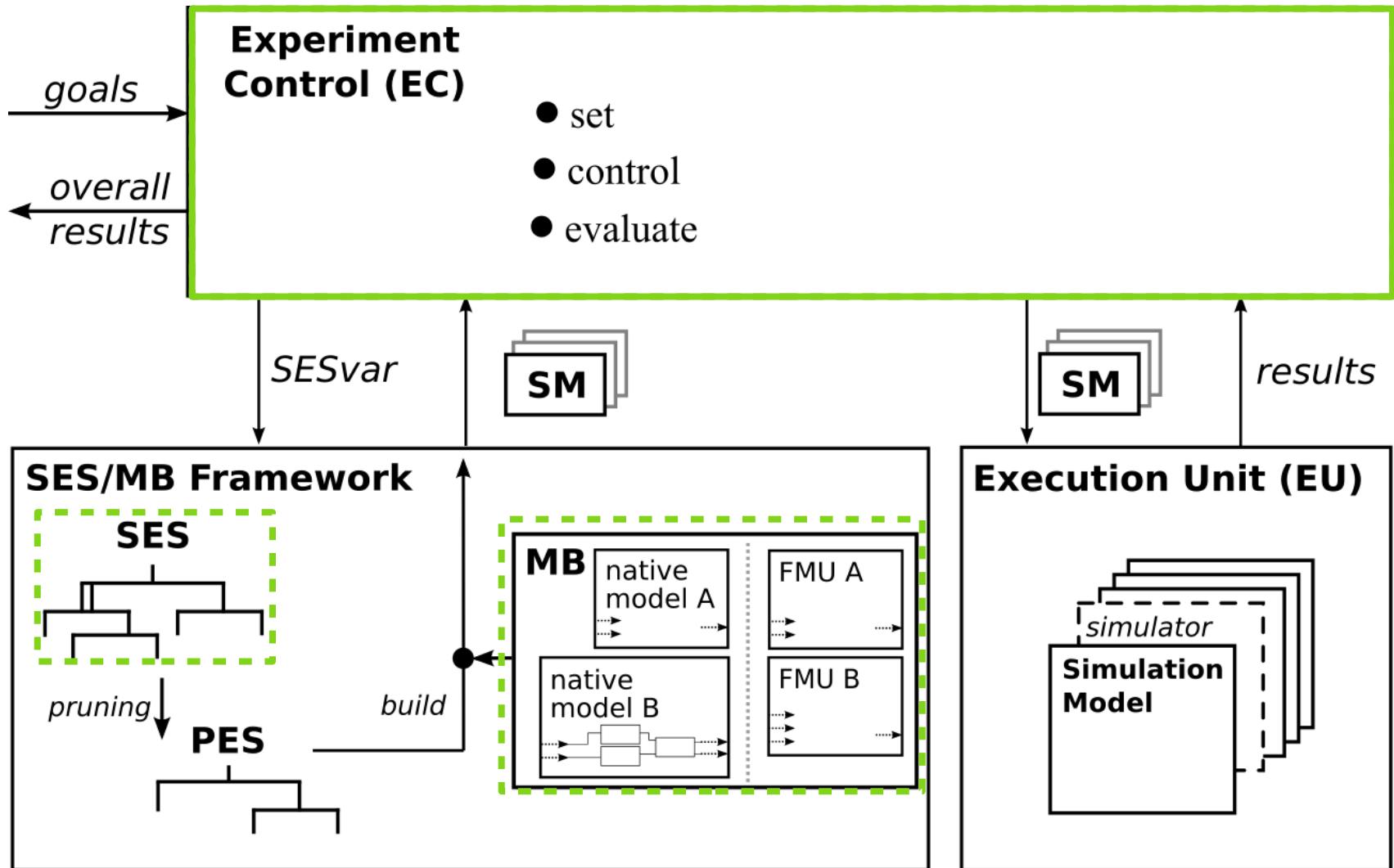


Extended SES/MB Architecture



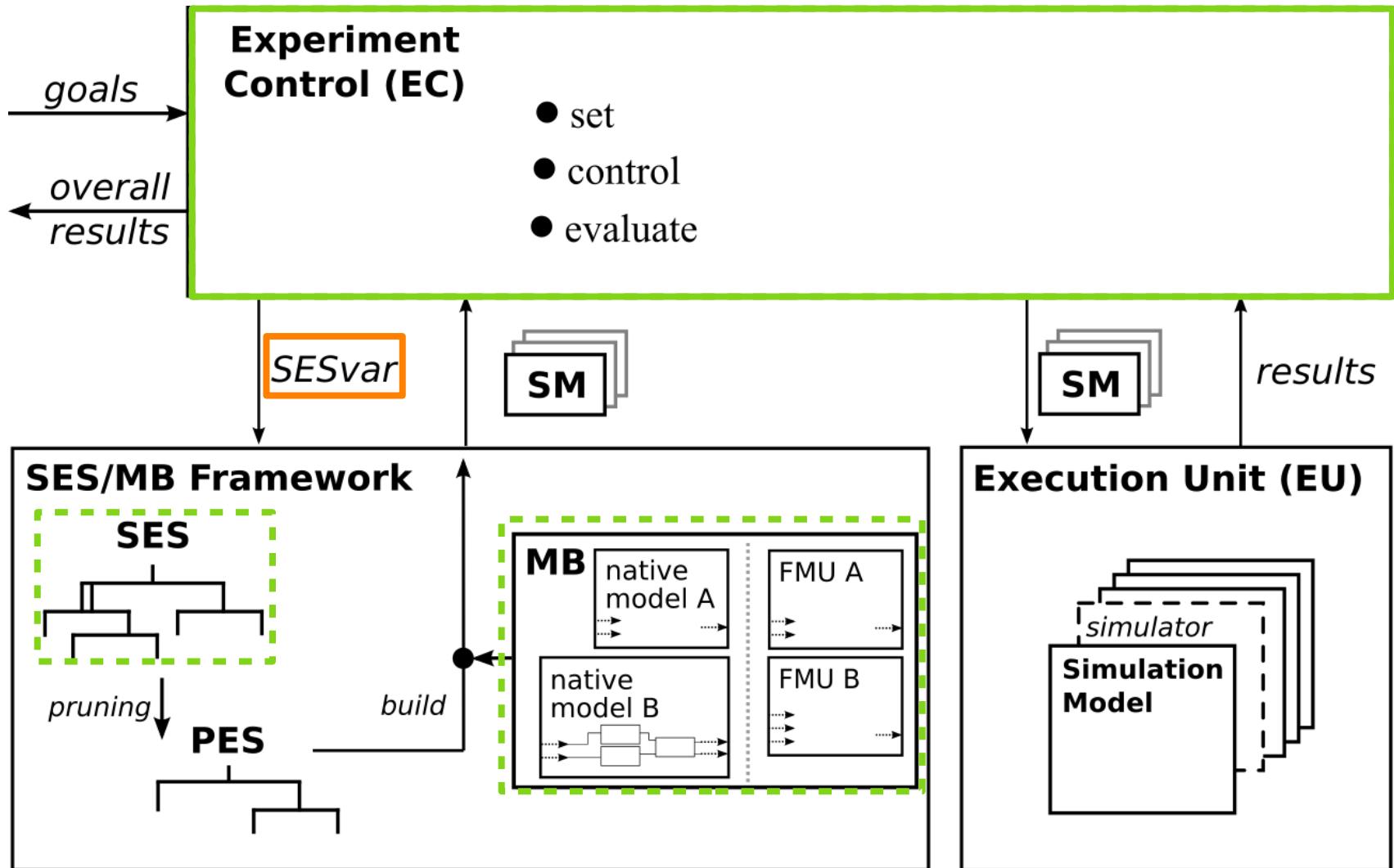


Extended SES/MB Architecture



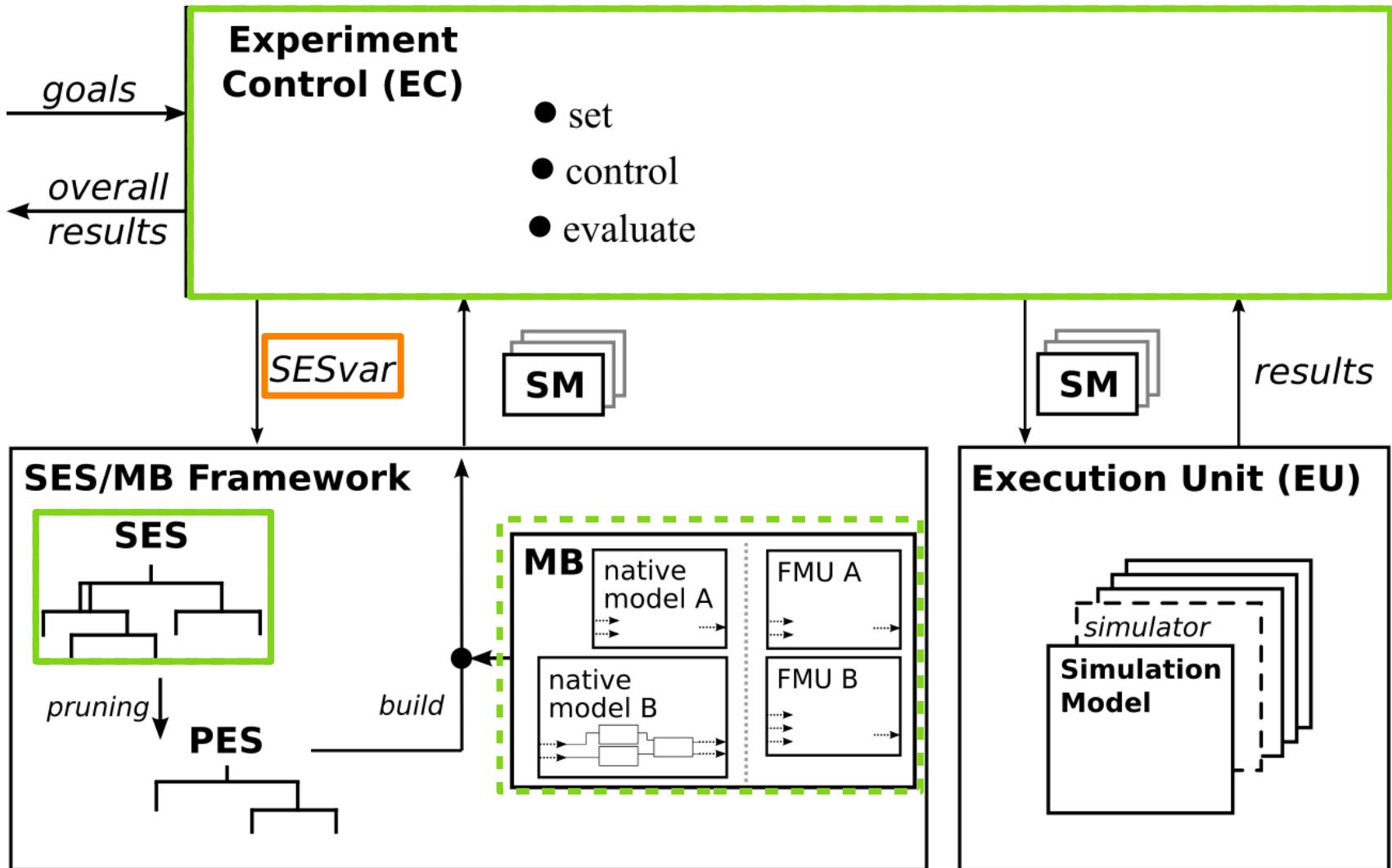


Extended SES/MB Architecture



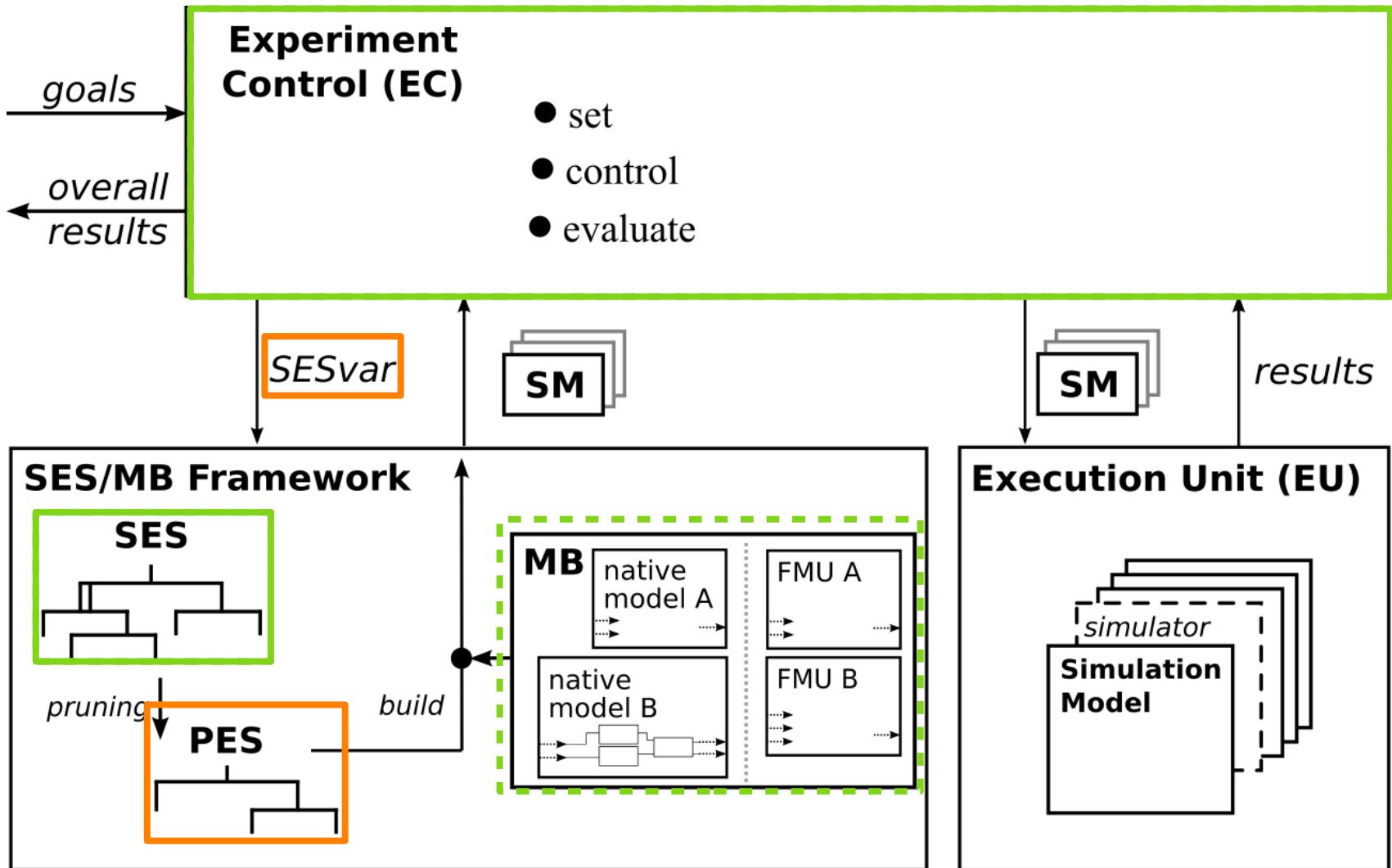


Extended SES/MB Architecture



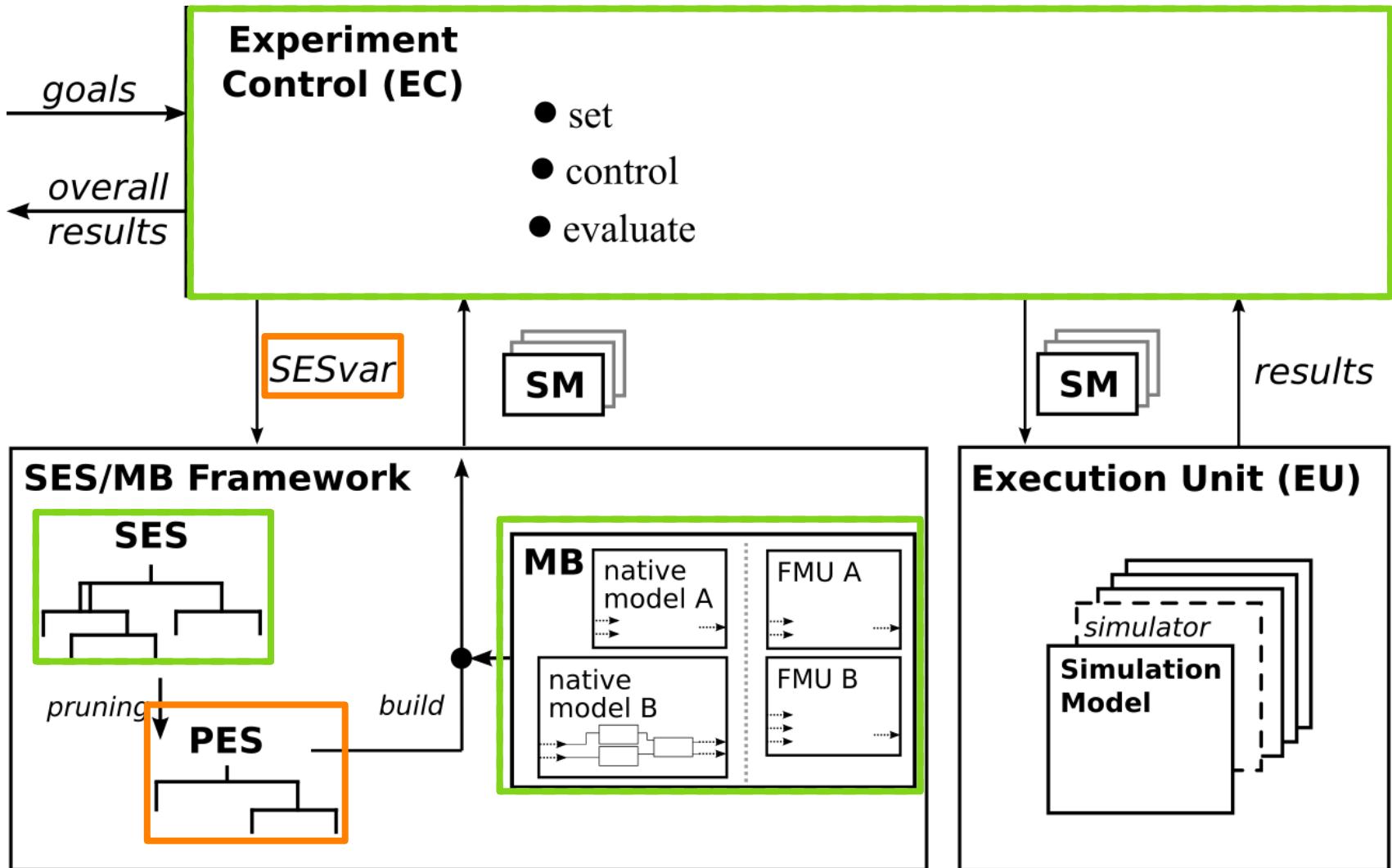


Extended SES/MB Architecture



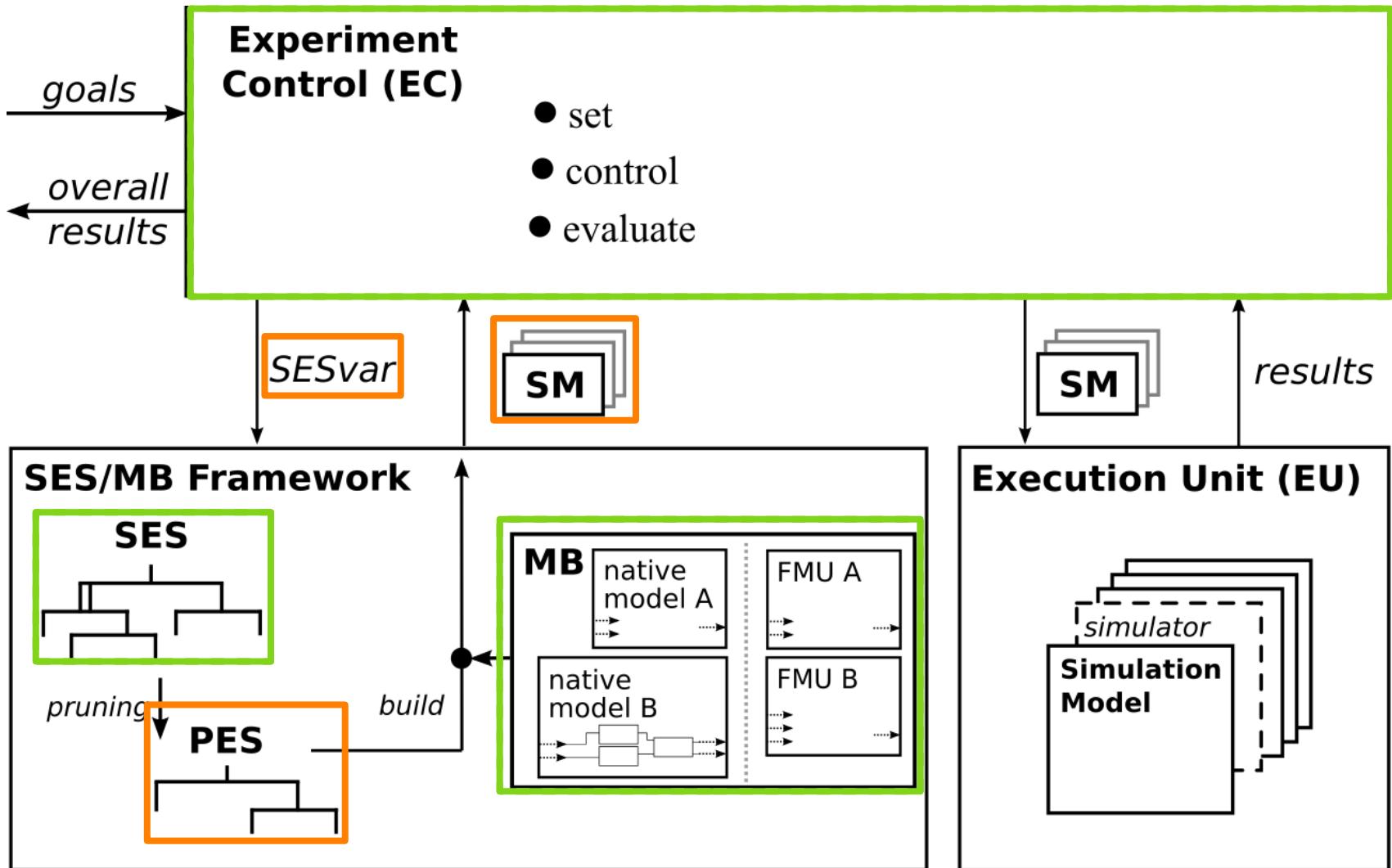


Extended SES/MB Architecture



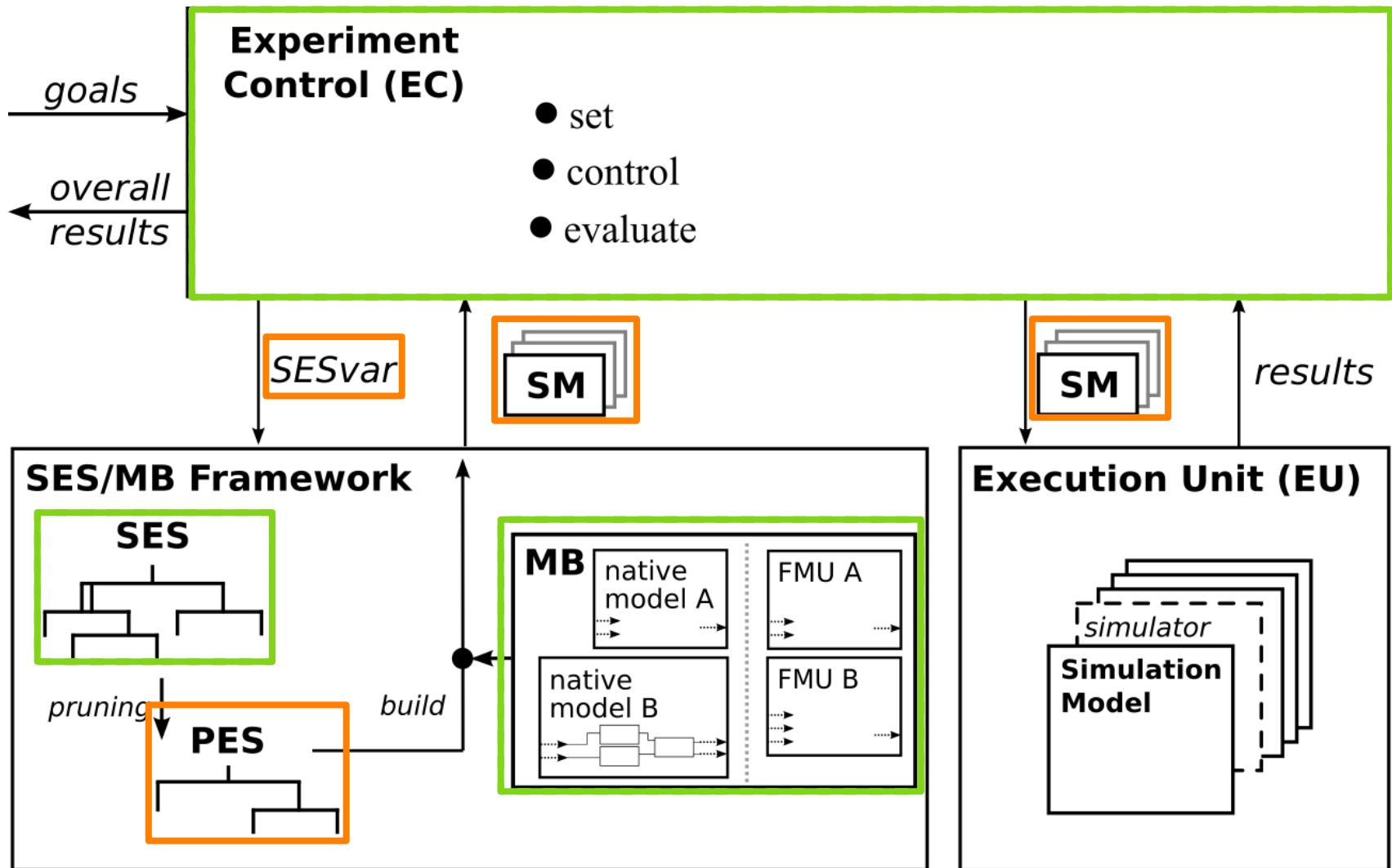


Extended SES/MB Architecture



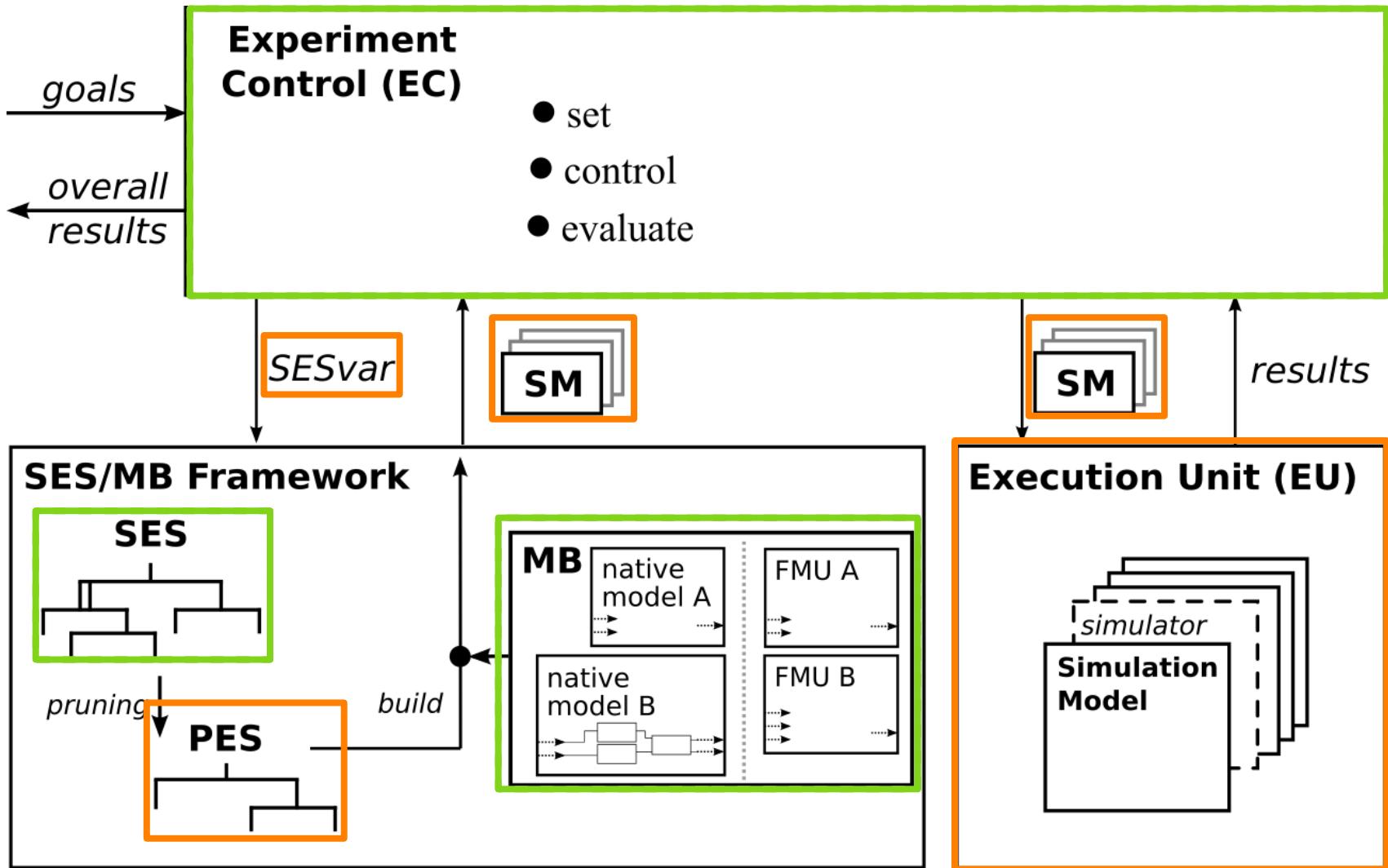


Extended SES/MB Architecture



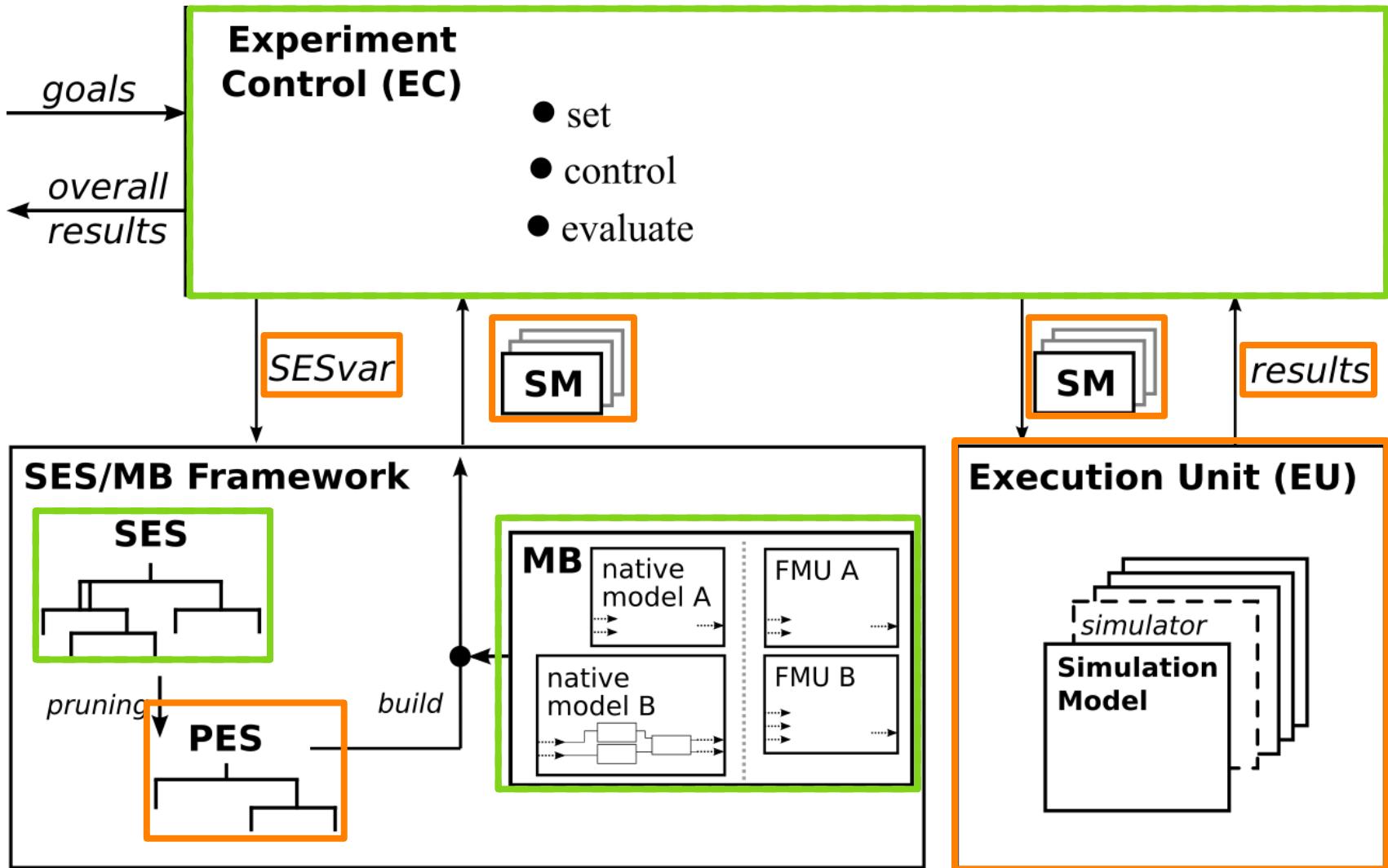


Extended SES/MB Architecture



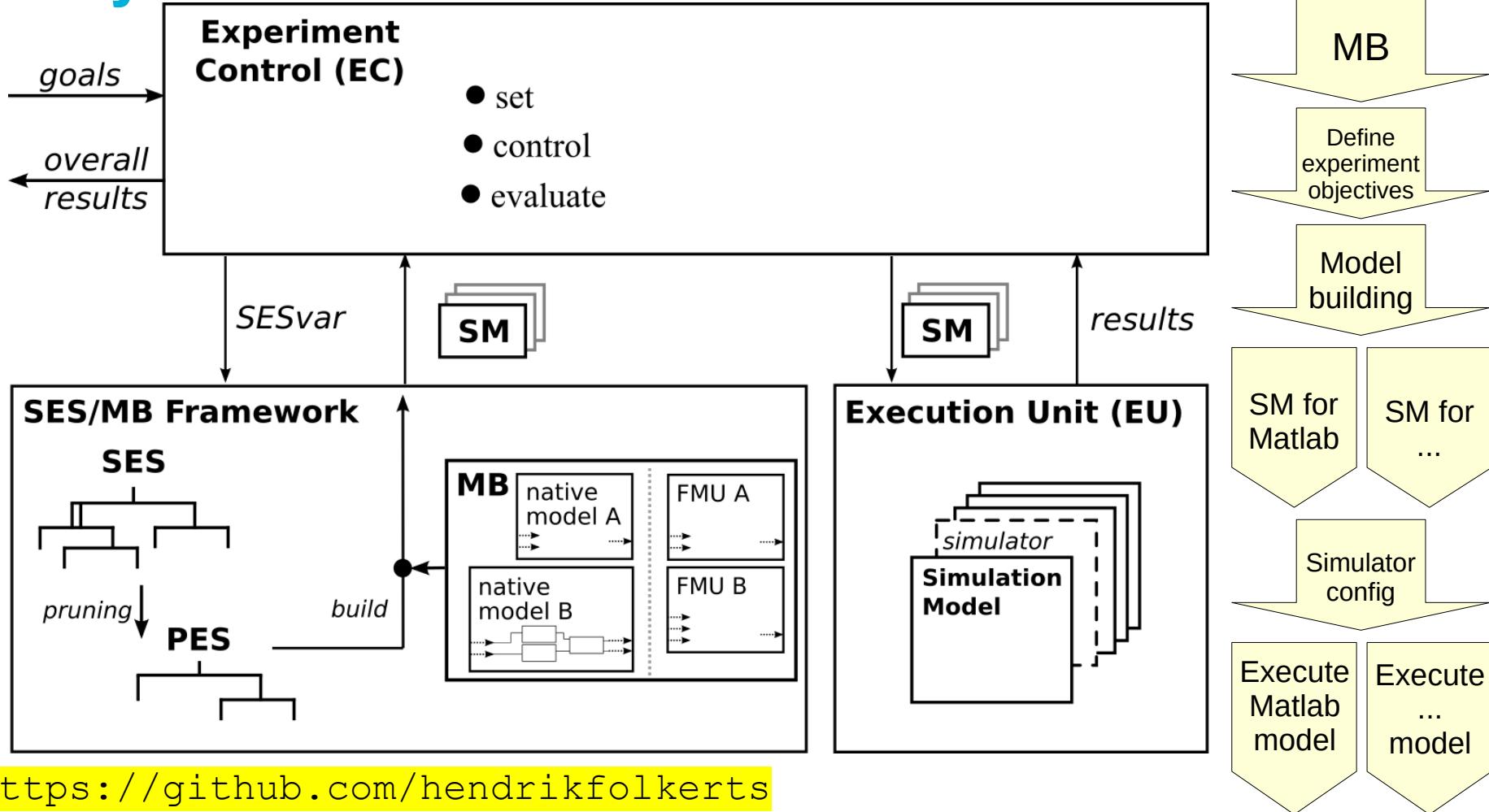


Extended SES/MB Architecture



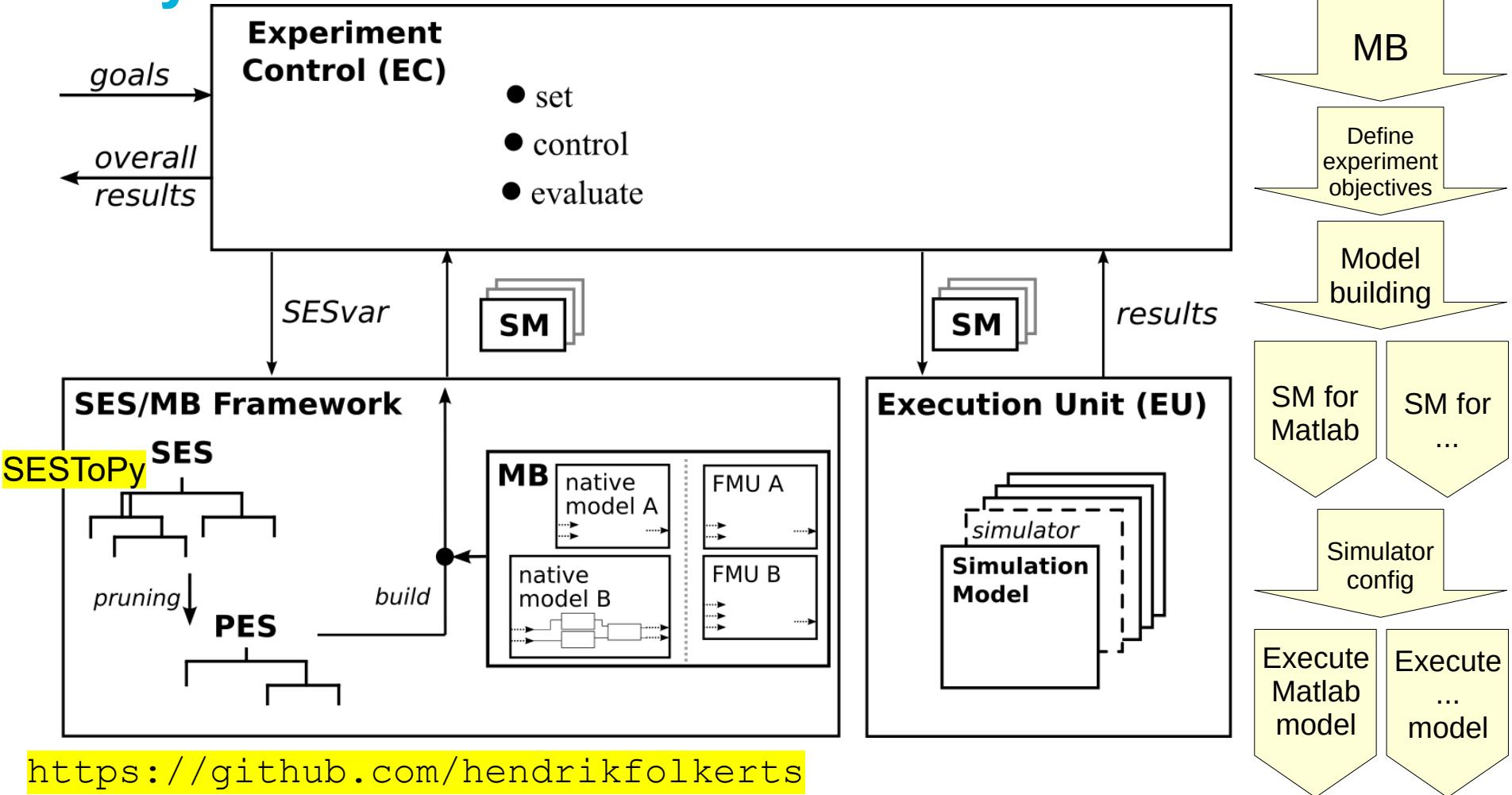


Extended SES/MB Architecture Python Tools



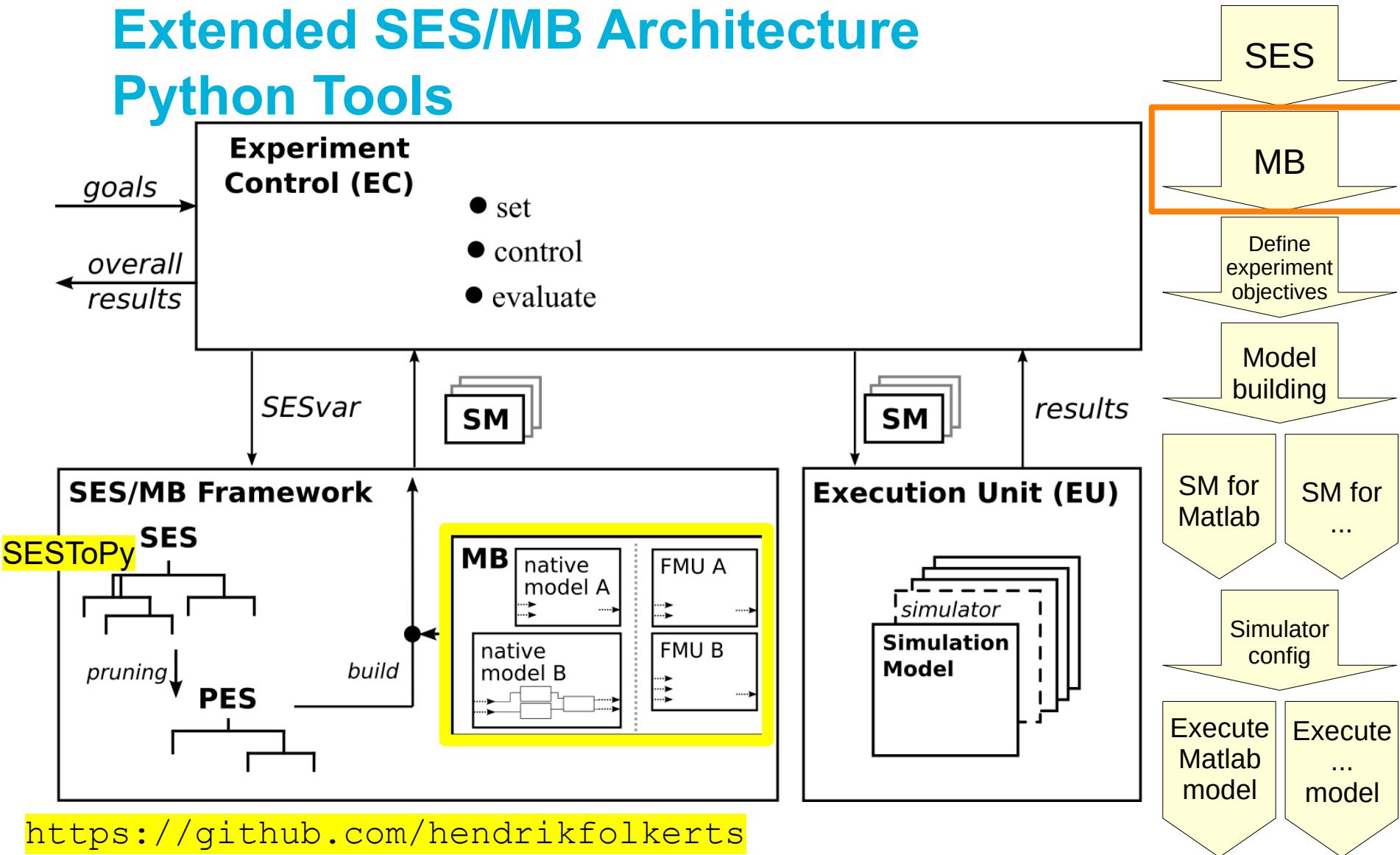


Extended SES/MB Architecture Python Tools



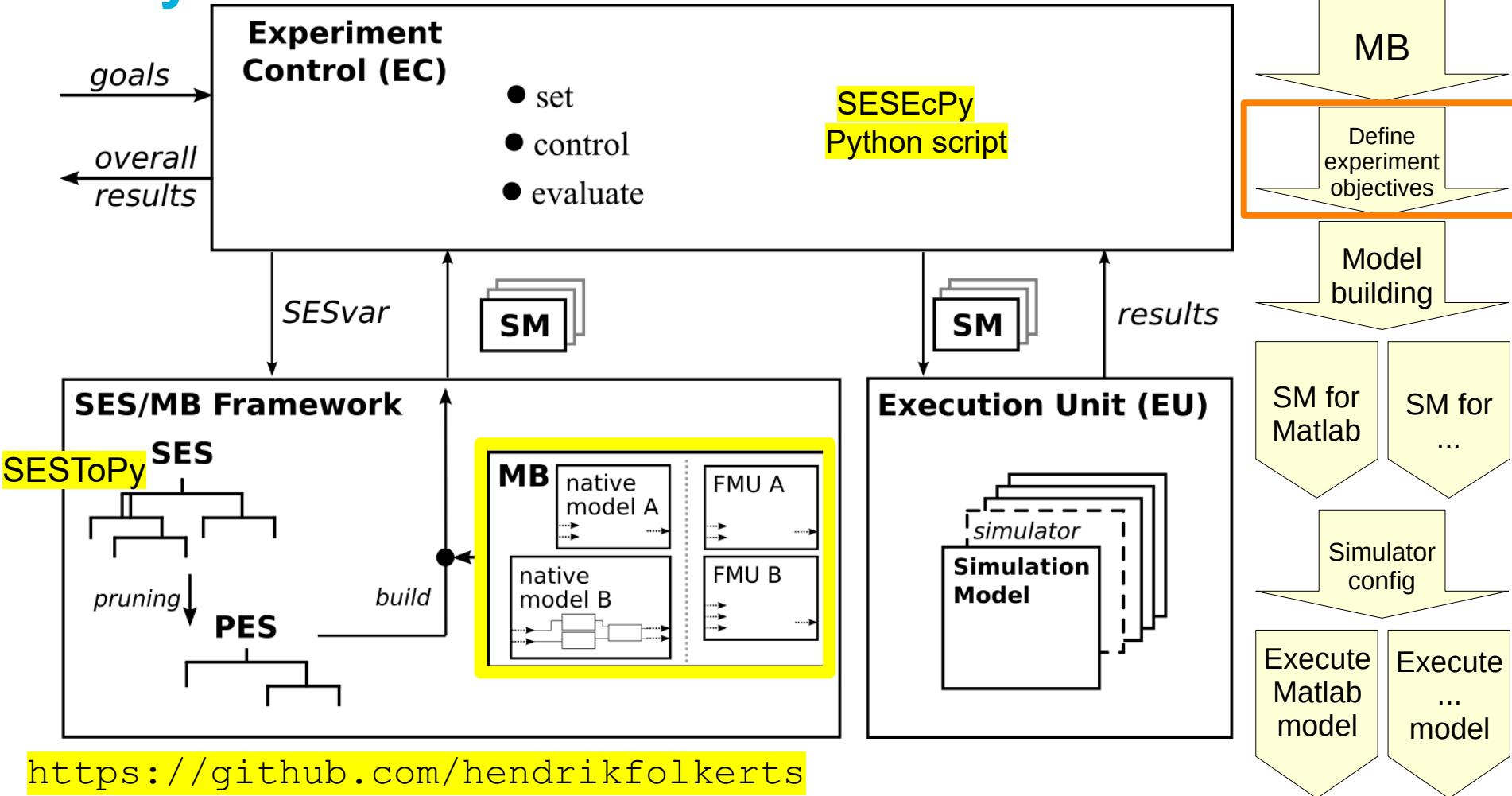


Extended SES/MB Architecture Python Tools



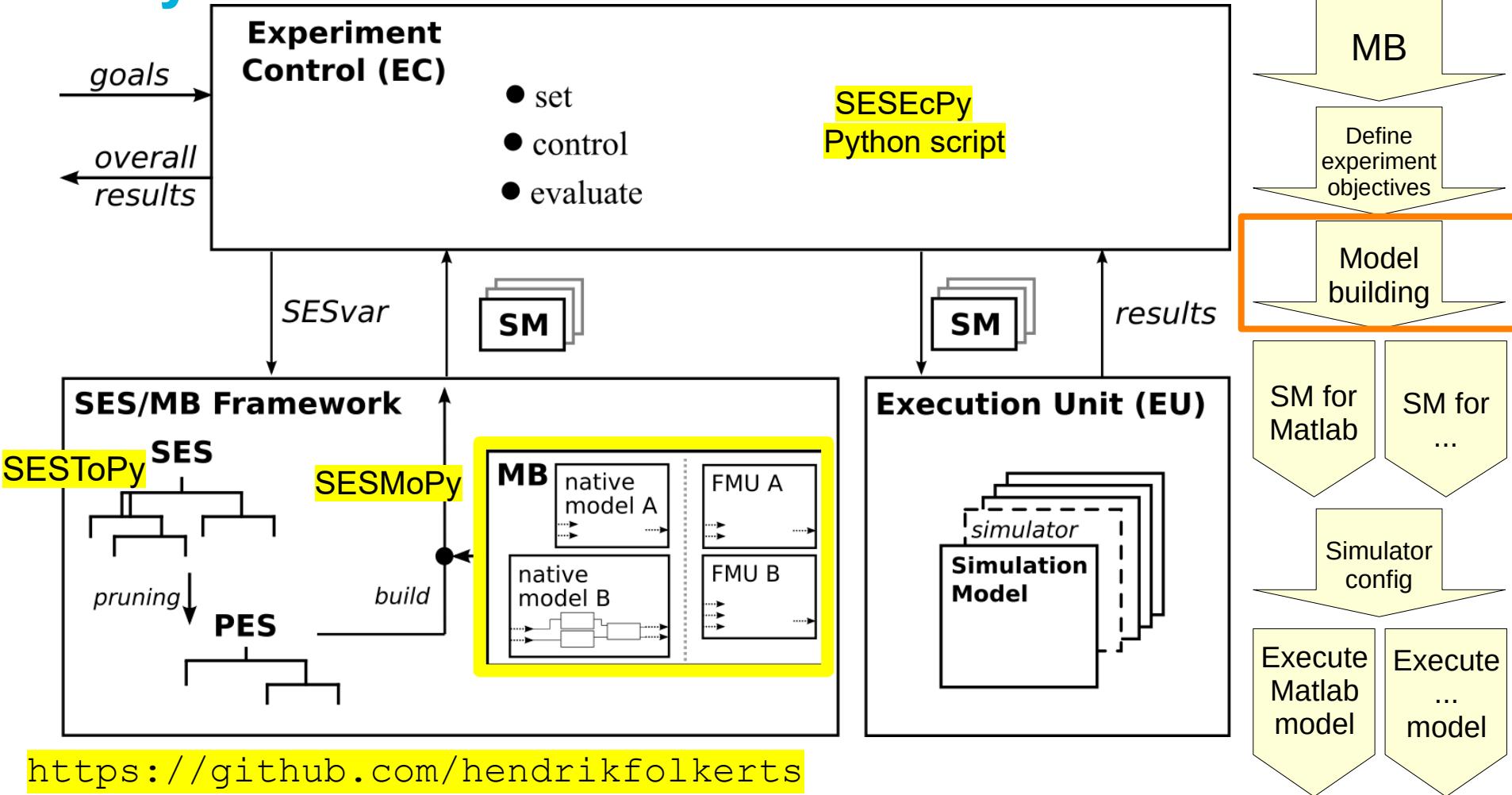


Extended SES/MB Architecture Python Tools



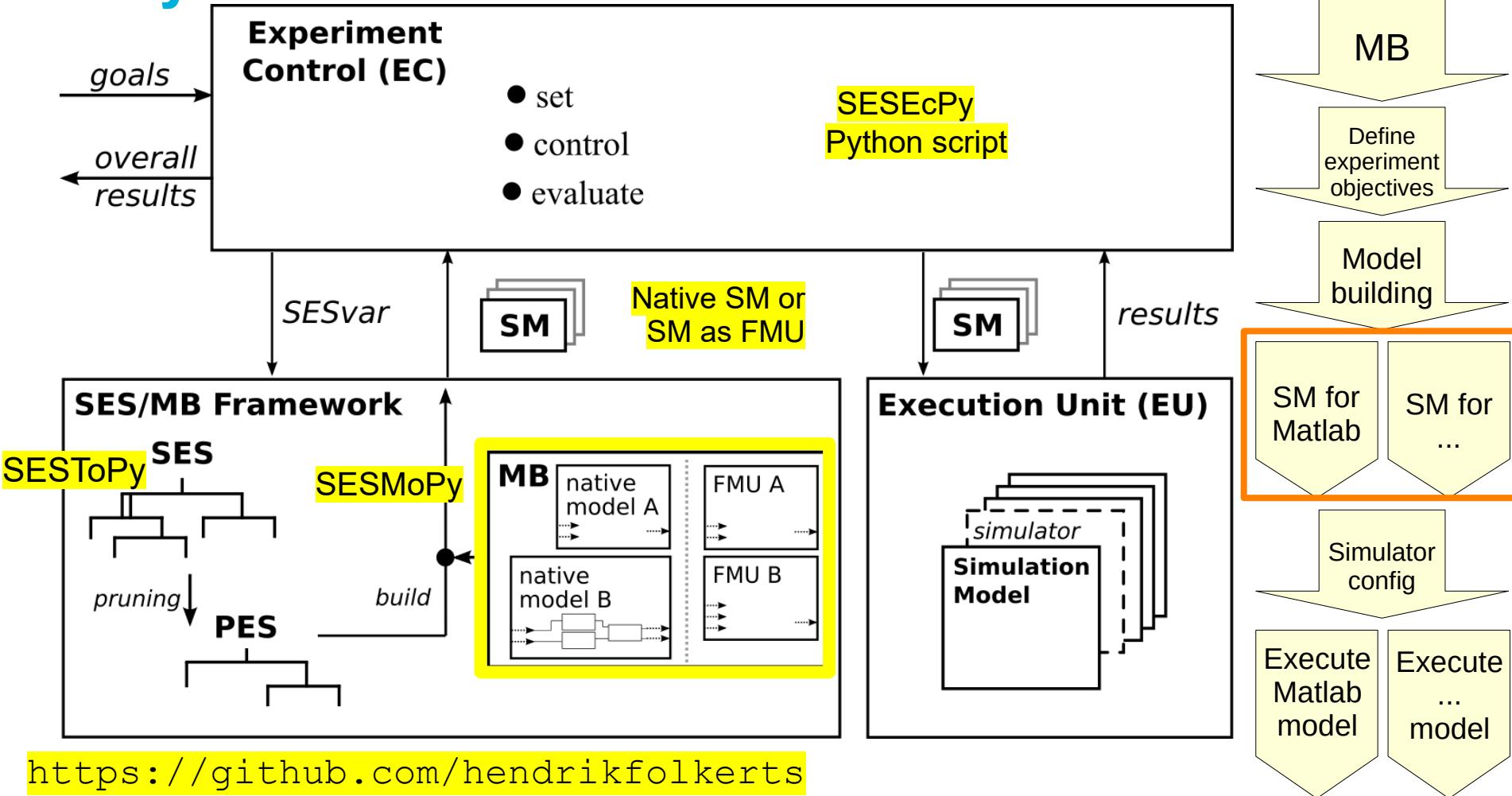


Extended SES/MB Architecture Python Tools



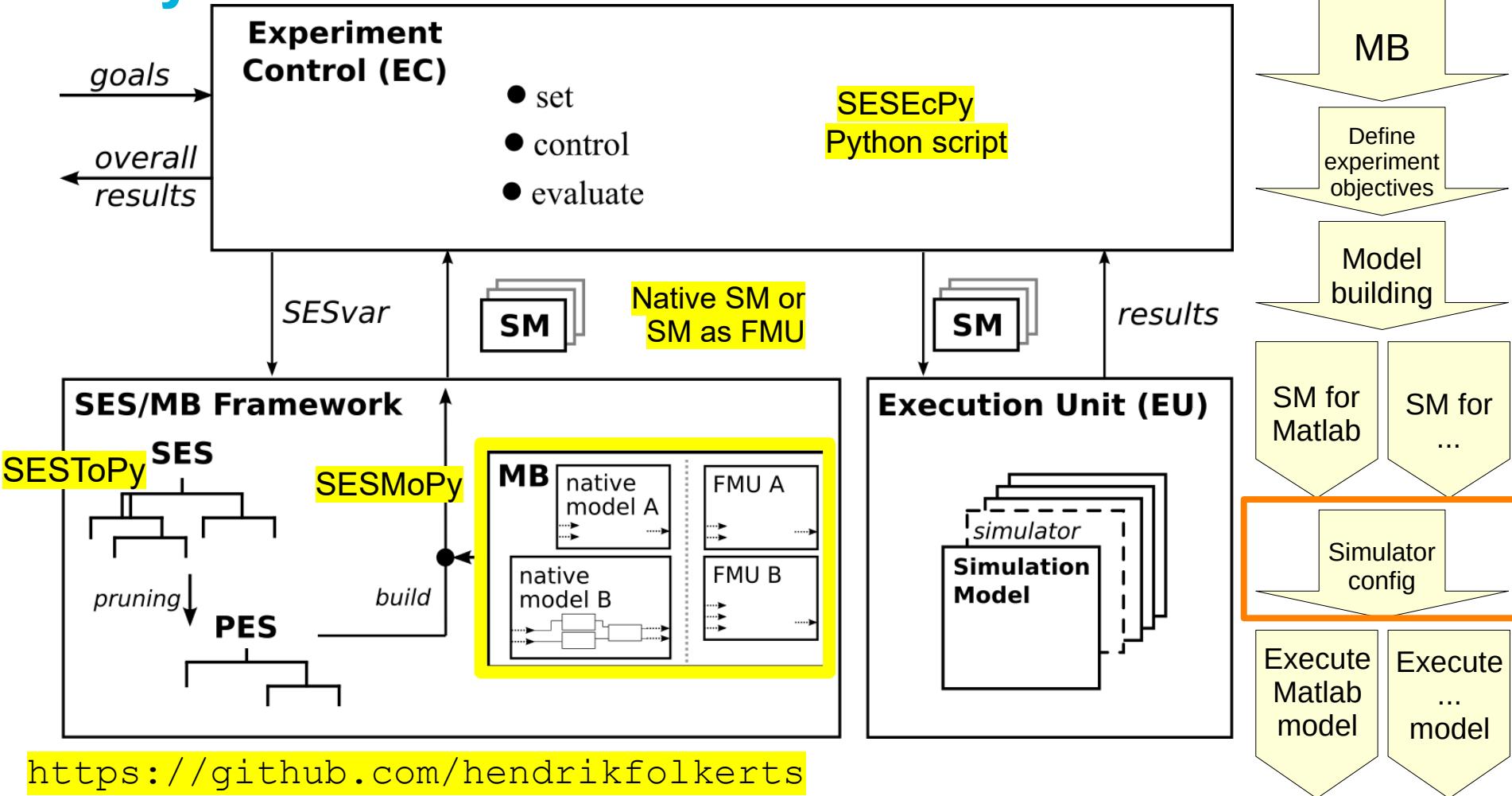


Extended SES/MB Architecture Python Tools



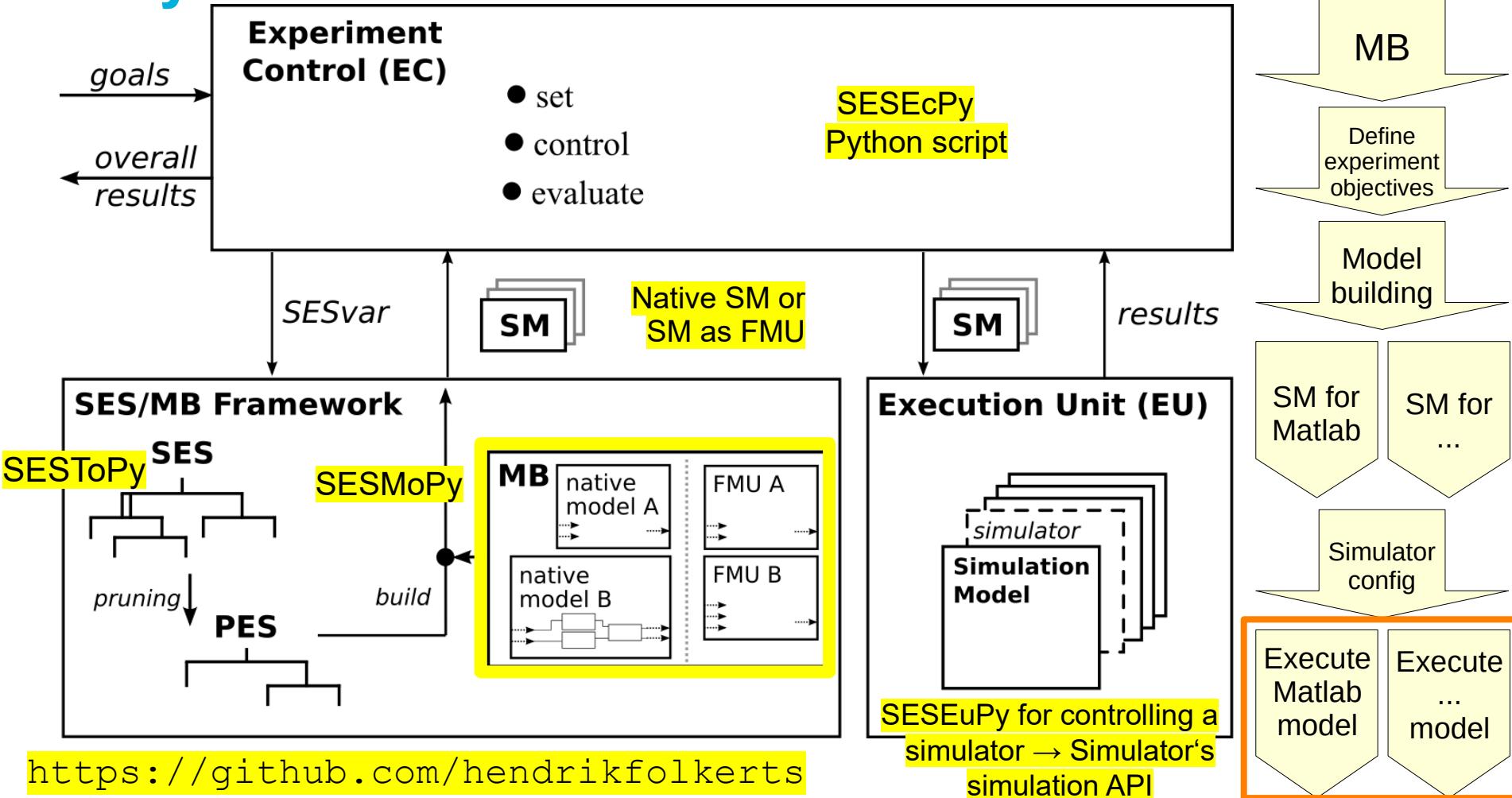


Extended SES/MB Architecture Python Tools





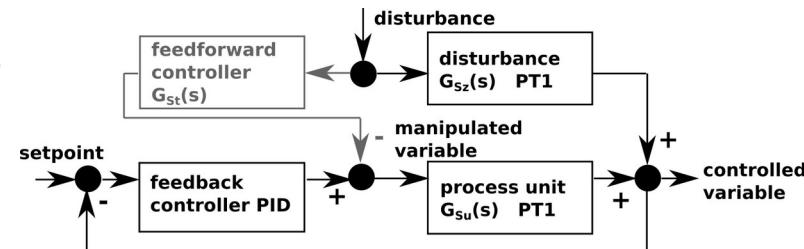
Extended SES/MB Architecture Python Tools



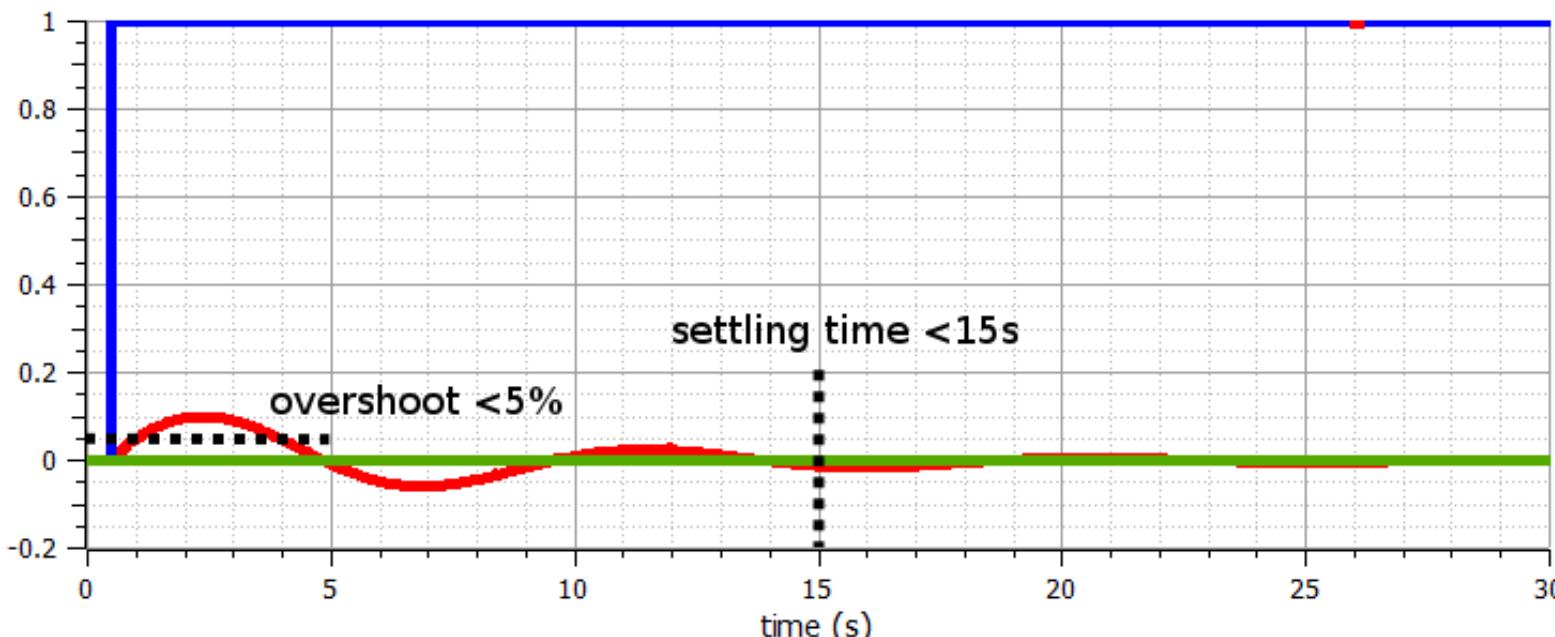


Case Study: Control Goals

- Goal for the control after a disturbance
 - Overshoot < 5%
 - Settling time < 15s



— controlled variable — disturbance — setpoint





Case Study: Experimentation Steps



Case Study: Experimentation Steps

- → **Code in Experiment Control ←**



Case Study: Experimentation Steps

- → **Code in Experiment Control ←**
- **Try without a feedforward control:**
 - `feedforward=0` simulate with PID: $k=1, Ti=1, Td=0$
 - `feedforward=0` simulate with PID: $k=5, Ti=0.5, Td=0$



Case Study: Experimentation Steps

- → **Code in Experiment Control ←**
- **Try without a feedforward control:**
 - feedforward=0 simulate with PID: $k=1, Ti=1, Td=0$
 - feedforward=0 simulate with PID: $k=5, Ti=0.5, Td=0$
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result



Case Study: Experimentation Steps

- → **Code in Experiment Control ←**
- **Try without a feedforward control:**
 - feedforward=0 simulate with PID: $k=1, Ti=1, Td=0$
 - feedforward=0 simulate with PID: $k=5, Ti=0.5, Td=0$
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result
- **Else try with a feedforward control:**
 - feedforward=1 simulate with both PID configurations



Case Study: Experimentation Steps

- → **Code in Experiment Control ←**
- **Try without a feedforward control:**
 - feedforward=0 simulate with PID: $k=1, Ti=1, Td=0$
 - feedforward=0 simulate with PID: $k=5, Ti=0.5, Td=0$
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result
- **Else try with a feedforward control:**
 - feedforward=1 simulate with both PID configurations
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result



Case Study: Experimentation Steps

- → **Code in Experiment Control ←**
- **Try without a feedforward control:**
 - feedforward=0 simulate with PID: $k=1, Ti=1, Td=0$
 - feedforward=0 simulate with PID: $k=5, Ti=0.5, Td=0$
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result
- **Else try with a feedforward control:**
 - feedforward=1 simulate with both PID configurations
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result
- **Else return: Goals cannot be reached with these configurations / parameters**

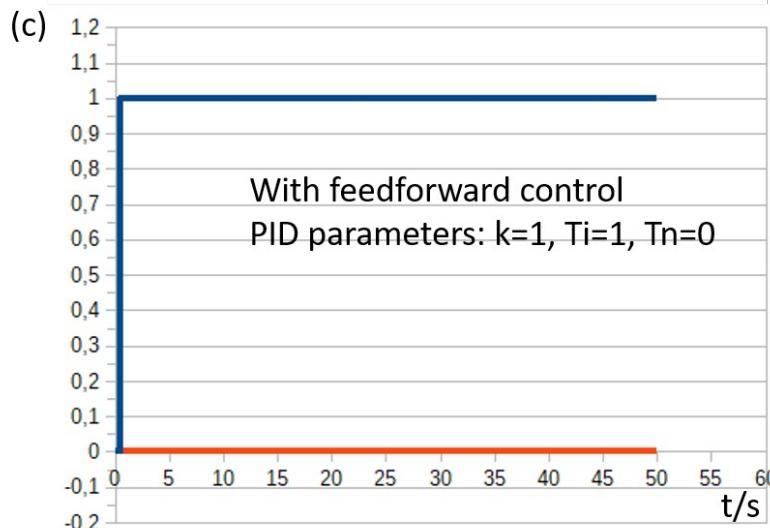
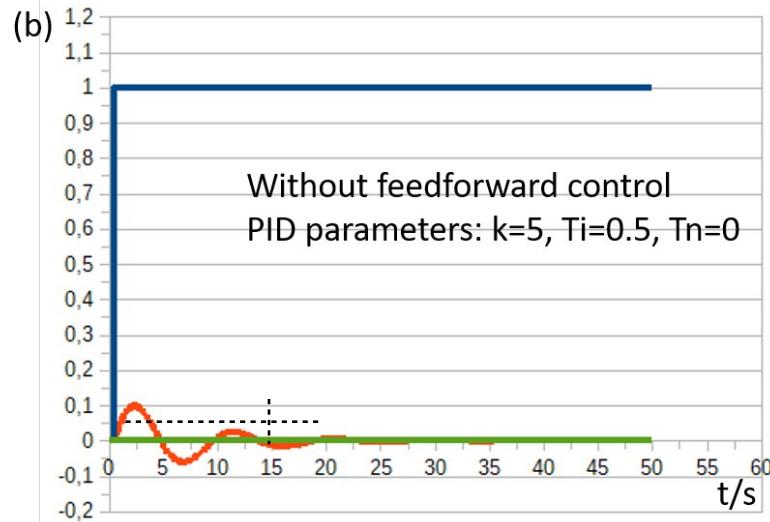
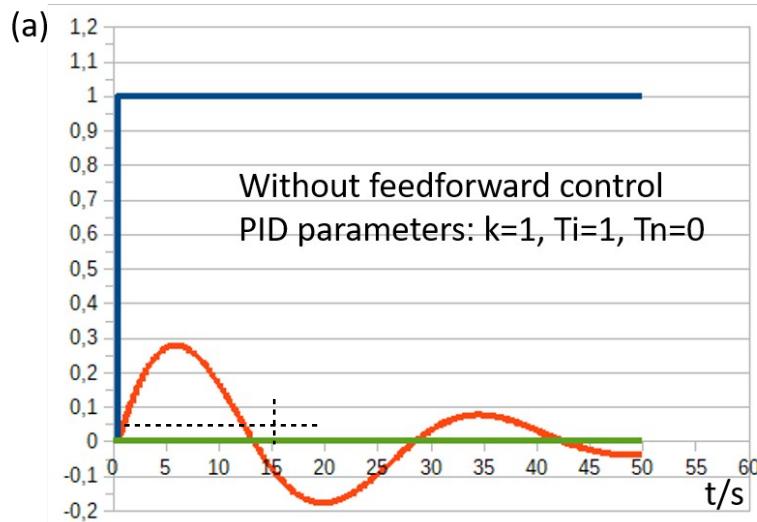


Case Study: Experimentation Steps

- → **Code in Experiment Control ←**
- **Try without a feedforward control:**
 - feedforward=0 simulate with PID: $k=1, Ti=1, Td=0$
 - feedforward=0 simulate with PID: $k=5, Ti=0.5, Td=0$
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result
- **Else try with a feedforward control:**
 - feedforward=1 simulate with both PID configurations
- **If the goals are reached with one of these configurations:**
 - Return PID configuration as overall result
- **Else return: Goals cannot be reached with these configurations / parameters**
- **Start over with another simulator → model by model validation**



Case Study: Simulation Results



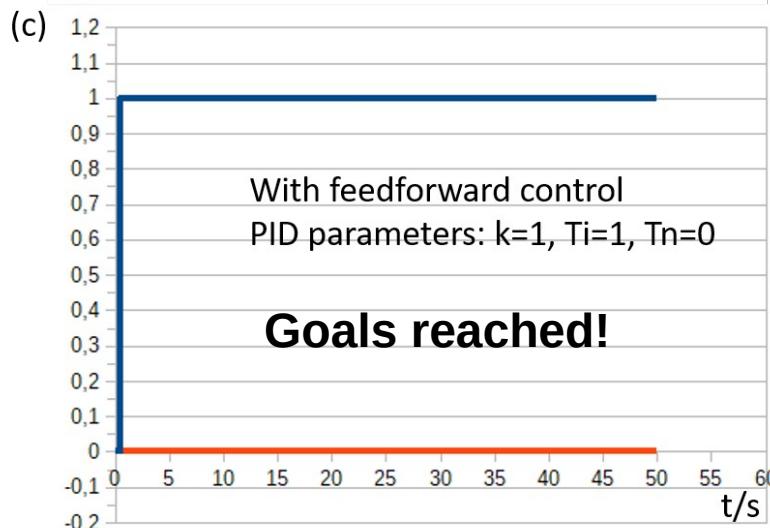
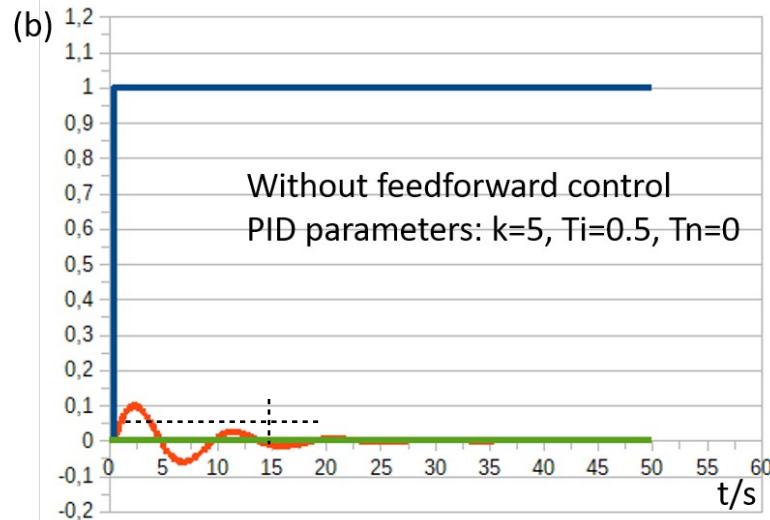
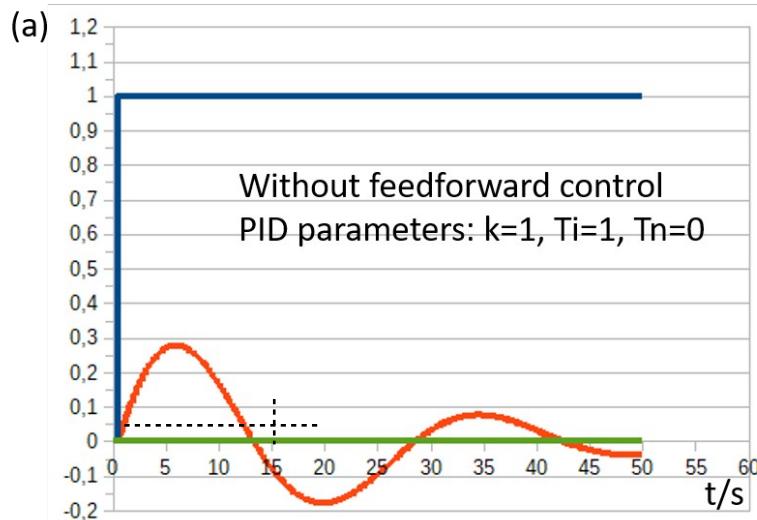
Key:

- green line: setpoint
- blue line: disturbance
- red line: controlled variable

Control goals:
overshoot < 5%
settling time < 15s



Case Study: Simulation Results



Key:

- setpoint
- disturbance
- controlled variable

Control goals:
overshoot < 5%
settling time < 15s



Outline

1. Introduction
2. The case study
3. Basics of SES/MB based modeling
4. Practical modeling: implementation of an SES
5. Model selection and model generation
6. The free Matlab SES toolbox
7. Organization of a simulator-independent MB
8. Full automation of simulation experiments
- 9. Conclusion** (T. Pawletta)



Conclusion

Sources for presented tools:

<https://github.com/cea-wismar>

<https://github.com/hendrikfolkerts>



Conclusion

- SES supports **simulator-independent modeling** of model configurations regarding model structures and parameter settings

Sources for presented tools:

<https://github.com/cea-wismar>

<https://github.com/hendrikfolkerts>



Conclusion

- SES supports **simulator-independent modeling** of model configurations regarding model structures and parameter settings
- MBs are usually simulator-specific (and system dynamic specific)
 - No problem, if working in only one M&S environment
 - Difficult maintenance, if working with multiple simulators

Sources for presented tools:

<https://github.com/cea-wismar>

<https://github.com/hendrikfolkerts>



Conclusion

- SES supports **simulator-independent modeling** of model configurations regarding model structures and parameter settings
- MBs are usually simulator-specific (and system dynamic specific)
 - No problem, if working in only one M&S environment
 - Difficult maintenance, if working with multiple simulators
- Using FMI a **simulator-independent MB** is possible
 - Support for efficient model building for multiple simulators (still problems for discrete event models)

Sources for presented tools:

<https://github.com/cea-wismar>

<https://github.com/hendrikfolkerts>



Conclusion

- SES supports **simulator-independent modeling** of model configurations regarding model structures and parameter settings
- MBs are usually simulator-specific (and system dynamic specific)
 - No problem, if working in only one M&S environment
 - Difficult maintenance, if working with multiple simulators
- Using FMI a **simulator-independent MB** is possible
 - Support for efficient model building for multiple simulators (still problems for discrete event models)
- The Extended SES/MB Architecture supports a **full experiment automation** regarding defined design objectives (using multiple simulators)

Sources for presented tools:

<https://github.com/cea-wismar>

<https://github.com/hendrikfolkerts>



Thank you!

Questions?



Backup



Chapter 4: Demonstration of SESToPy and SESViewEI Backup Slides



Connection of SESToPy and SESViewEI

The screenshot displays two windows illustrating the connection between SESToPy and SESViewEI.

SESToPy - SES Tools in Python3 / PyQt5 window:

- Toolbar:** Open, Save, Empty Current Model, Prune, Flatten.
- Menu Bar:** File, Edit, Merge, Transformation, ?
- Status Bar:** Model 1: not saved, Connection to server program taking the XML tree: Server IP 127.0.0.1, Server Port 5454, Connect (button highlighted with a red box), Disconnect.
- Global Settings:** SES / PES (radio buttons: SES (selected), incompletely pruned PES, PES, flattened PES). SES comment:
- Hierarchy Model:** Node dropdown set to Entity Node. Tree view showing a single node labeled "Tree".
- Node Specific Properties:** Default.
- Attributes:** Aspects, Number of Replications, Coupling, Specule.
- Bottom Buttons:** Help, SES Variables, Semantic Conditions (checkmark), Selection Constraints, SES Functions.

SESViewEI window:

- Title Bar:** SESViewEI - System Entity Structure View in Node.js / Electron.
- Text:** A socket server is started on port 54545. SESToPy can connect to this server now. Exit the program here: - Buttons:** Tree without Icons, Save Tree as SVG, Show/Hide Node Attributes.
- Text Fields:** Choose Font Size/Weight for the Tree: 10, normal. Nodename [left]/[right] of icon: Node [has] [has no] subtree.



Create SES Tree with SESToPy

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: not saved

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information

SES / PES

- SES
- incompletely pruned PES
- PES
- flattened PES

SES comment

Hierarchy Model

Node Entity Node

Tree

- EntityRoot Entity
- Descriptive Aspect
 - Object1 Entity
 - Object2 Entity

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
Name			
value			

Insert Delete Help

Aspectrule

Number of Replications

Coupling Specrule

SESViewEI

SESViewEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here: [Exit](#)

Tree without Icons? Save Tree as SVG The file 'tree.svg' was written in the program directory.

Choose Font Size/Weight for the Tree: 15 normal Nodename [left]/[right] of icon: Node [has] [has no] subtree.

SES variables semantic conditions Show/Hide Node Attributes

```
graph TD; EntityRoot[EntityRoot Entity] --- Descriptive[Descriptive Aspect]; Descriptive --- Object1[Object1 Entity]; Descriptive --- Object2[Object2 Entity]
```



Create SES Tree with SESToPy

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: not saved

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information

SES / PES

- SES
- incompletely pruned PES
- PES
- flattened PES

SES comment

Hierarchy Model

Node Entity Node

Tree

- EntityRoot Entity
- Descriptive Aspect
 - Object1 Entity
 - Object2 Entity

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
Name			
value			

Insert Delete Help

Aspectrule

Number of Replications

Coupling Specrule

SESViewEI

SESViewEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here: [Exit](#)

Tree without Icons? Save Tree as SVG The file 'tree.svg' was written in the program directory.

Choose Font Size/Weight for the Tree: 15 normal Nodename [left]/[right] of icon: Node [has] [has no] subtree.

SES variables semantic conditions Show/Hide Node Attributes

```
graph TD; EntityRoot[EntityRoot Entity] --- Descriptive[Descriptive Aspect]; Descriptive --- Object1[Object1 Entity]; Descriptive --- Object2[Object2 Entity]
```



Create SES Tree with SESToPy

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: not saved

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information

SES / PES

- SES
- incompletely pruned PES
- PES
- flattened PES

SES comment

Hierarchy Model

Node Entity Node

Tree	Type	MB	atr
EntityRoot	Entity		
Descriptive	Aspect		
Object1	Entity		
Object2	Entity		

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment

Name _____
value _____
Insert Delete Help

Aspectrule
Number of Replications
Coupling
Specrule

SESViewEI

SESViewEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here: [Exit](#)

Tree without Icons? Save Tree as SVG The file 'tree.svg' was written in the program directory.

Choose Font Size/Weight for the Tree: 15 normal Nodename [left]/[right] of icon: Node [has] [has no] subtree.

SES variables semantic conditions Show/Hide Node Attributes

```
graph TD; EntityRoot[EntityRoot] --> Descriptive[Descriptive]; Descriptive --> Object1[Object1]; Descriptive --> Object2[Object2]
```



Create SES Tree with SESToPy

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: not saved

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Model 1 Model 2 Model 3 Model 4 Model 5 Model 6 Model 7 Model 8 Model 9 Model 10

Global Settings Information

SES / PES

- (radio) SES
- (radio) incompletely pruned PES
- (radio) PES
- (radio) flattened PES

SES comment

Hierarchy Model

Node Entity Node

Tree

- EntityRoot
 - Descriptive
 - Object1
 - Object2

Press F2 key to rename

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
Name			
value			

Insert Delete Help

Aspectrule

Number of Replications

Coupling Specrule

SESViewEI

SESViewEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here:

Tree without Icons? Save Tree as SVG The file 'tree.svg' was written in the program directory.

Choose Font Size/Weight for the Tree: 15 Nodename [left]/[right] of icon: Node [has] [has no] subtree.

SES variables semantic conditions Show/Hide Node Attributes

```
graph TD; EntityRoot --> Descriptive; Descriptive --> Object1; Descriptive --> Object2;
```



Create SES Tree with SESToPy

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: not saved

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information

SES / PES

- (radio) SES
- (checkbox) incompletely pruned PES
- (checkbox) PES
- (checkbox) flattened PES

SES comment

Hierarchy Model

Node Entity Node

Tree EntityRoot Entity

Type MB atr

EntityAspect Entity

Object1 Entity

Object2 Entity

Press F2 key to rename

Node Specific Properties Default Attributes

Name Value var/fun comment

Name _____

value _____

Insert Delete Help

Aspectrule Number of Replications Coupling Specrule

SESViewEI

SESViewEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here:

Tree without Icons? Save Tree as SVG The file 'tree.svg' was written in the program directory.

Choose Font Size/Weight for the Tree: 15 normal Nodename [left]/[right] of icon: Node [has] [has no] subtree.

SES variables semantic conditions Show/Hide Node Attributes

EntityRoot Entity

Descriptive Entity

Object1 Entity

Object2 Entity

2021-02-02 / 19:00:28 / Version 2021.02.02



Create SES Tree with SESToPy

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: not saved

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information

SES / PES

- (radio) SES
- (checkbox) incompletely pruned PES
- (checkbox) PES
- (checkbox) flattened PES

SES comment

Hierarchy Model

Node Entity Node

Tree EntityRoot Entity Object1 Entity Object2 Entity

Press F2 key to rename

Node Specific Properties Default Attributes

Name	Value	var/fun	comment
Name			
value			

Insert Delete Help

Aspectrule Number of Replications Coupling Specrule

SESViewEI

SESViewEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here:

Tree without Icons? Save Tree as SVG The file 'tree.svg' was written in the program directory.

Choose Font Size/Weight for the Tree: 15 normal Nodename [left]/[right] of icon: Node [has] [has no] subtree.

SES variables semantic conditions Show/Hide Node Attributes

```
graph TD; EntityRoot --> Descriptive; Descriptive --> Object1; Descriptive --> Object2;
```



Create SES Tree with SESToPy

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: not saved

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information

SES / PES

- (radio) SES
- (checkbox) incompletely pruned PES
- (checkbox) PES
- (checkbox) flattened PES

SES comment

Hierarchy Model

Node Entity Node

Tree EntityRoot EntityAspect Object1 Object2

Press F2 key to rename

Node Specific Properties Default Attributes

Name	Value	var/fun	comment

Name _____ Value _____

Insert Delete Help

Aspectrule Number of Replications Coupling Specrule

SESViewEI

SESVIEWEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here: [Exit](#)

Tree without Icons? Save Tree as SVG The file 'tree.svg' was written in the program directory.

Choose Font Size/Weight for the Tree: 15 normal Nodename [left]/[right] of icon: Node [has] [has no] subtree.

SES variables semantic conditions Show/Hide Node Attributes

```
graph TD; EntityRoot --> Descriptive; Descriptive --> Object1; Descriptive --> Object2;
```



Create SES Tree with SESToPy

The image shows two windows demonstrating the creation of an SES tree using SESToPy.

SESToPy - SES Tools in Python3 / PyQt5 (Left Window):

- Toolbar:** File, Edit, Merge, Transformation, ?; Open, Save, Empty Current Model, Prune, Flatten.
- Model Status:** Model 1: not saved.
- Connection:** Connection to server program taking the XML tree: Server IP 127.0.0.1, Server Port 54545, Connect, Disconnect.
- Hierarchy Model:** Shows a tree structure under "Tree". The root node is "Entity Node". It has a child "EntityRoot" which further has children "Descriptive" and "Object1". "Object1" has a child "Object2". A tooltip "Press F2 key to rename" points to the "Entity Node" label.
- Node Specific Properties:** Default, Attributes.
- Table:** Name | Value | var/fun | comment.
- Buttons:** Insert, Delete, Help.
- Form Fields:** Name, value, Aspectsrule, Number of Replications, Coupling, Specrule.
- Left Panel:** Global Settings, Information, SES / PES (radio buttons for SES, incompletely pruned PES, PES, flattened PES), SES comment, Help.
- Bottom:** 2021-02-02 / 19:00:28 / Version 2021.02.02.

SESViewEI (Right Window):

- Title:** SESViewEI - System Entity Structure View in Node.js / Electron.
- Message:** A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here: - Controls:** Tree without Icons? , Save Tree as SVG, Choose Font Size/Weight for the Tree: 15, normal, Show/Hide Node Attributes.
- Diagram:** Displays the SES tree structure. The root node is "EntityRoot". It has a child "Descriptive". "Descriptive" has two children: "Object1" and "Object2".



Create SES Tree with SESToPy

The image shows two windows related to system entity structure (SES) modeling:

SESToPy - SES Tools in Python3 / PyQt5 (Left Window):

- Toolbar:** File, Edit, Merge, Transformation, ?; Open, Save, Empty Current Model, Prune, Flatten.
- Model Status:** Model 1: not saved.
- Connection:** Connection to server program taking the XML tree: Server IP 127.0.0.1, Server Port 54545, Connect, Disconnect.
- Hierarchy Model:** A tree view showing the structure: Entity Node → EntityRoot (Entity) → Descriptive (Aspect) → Object1 (Entity) → Object2 (Entity). The "Entity Node" entry in the list is highlighted with a red box. An arrow points from the text "Press F2 key to rename" to the "Entity Node" entry.
- Node Specific Properties:** Default, Attributes.
- Node List:** Name, Value, var/fun, comment.
- Buttons:** Insert, Delete, Help.
- Bottom:** Aspects rule, Number of Replications, Coupling, Specrule.

SESViewEI (Right Window):

- Title:** SESViewEI - System Entity Structure View in Node.js / Electron.
- Message:** A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here: **Exit**.
- Settings:** Tree without Icons? , Save Tree as SVG, Choose Font Size/Weight for the Tree: 15, normal, Nodename [left]/[right] of icon: Node [has] [has no] subtree, SES variables, semantic conditions, Show/Hide Node Attributes.
- Diagram:** A visual representation of the SES tree structure. The root node is EntityRoot, which has a child Descriptive. Descriptive has two children: Object1 and Object2.



Edit Entity Node with SESToPy

The screenshot shows the SESToPy application window. The title bar reads "SESToPy - SES Tools in Python3 / PyQt5". The menu bar includes File, Edit, Merge, Transformation, and Help. The toolbar contains icons for Open, Save, Empty Current Model, Prune, and Flatten.

The main interface consists of several panels:

- Model 1: not saved**: A tab bar with tabs for Model 1 through Model 10.
- Global Settings**: A panel containing "Information" and "SES / PES" sections. The "SES" radio button is selected. Other options include "incompletely pruned PES", "PES", and "flattened PES". A "SES comment" text area is also present.
- Hierarchy Model**: A tree view titled "Entity Node". The tree structure is as follows:

```
Tree          Type   MB   atr   ars   cpl   srs   uid
EntityRoot    Entity
Descriptive   Aspect
Object1       Entity
Object2       Entity
```

The "Object2" node is highlighted with a red border.
- Node Specific Properties**: A panel for editing properties of the selected node ("Object2"). It includes a "Default" section and an "Attributes" table. The "Attributes" table has columns: Name, Value, var/fun, and comment. One row is present: Name "var1" with Value "1". Below the table are fields for "Name" (var2) and "Value" (3), along with Insert, Delete, and Help buttons.
- Bottom Panels**: Includes sections for Aspectrule, Number of Replications, Coupling, and Specrule.

At the bottom left, the status bar displays "2021-02-02 / 19:04:14 / Version 2021.02.02".



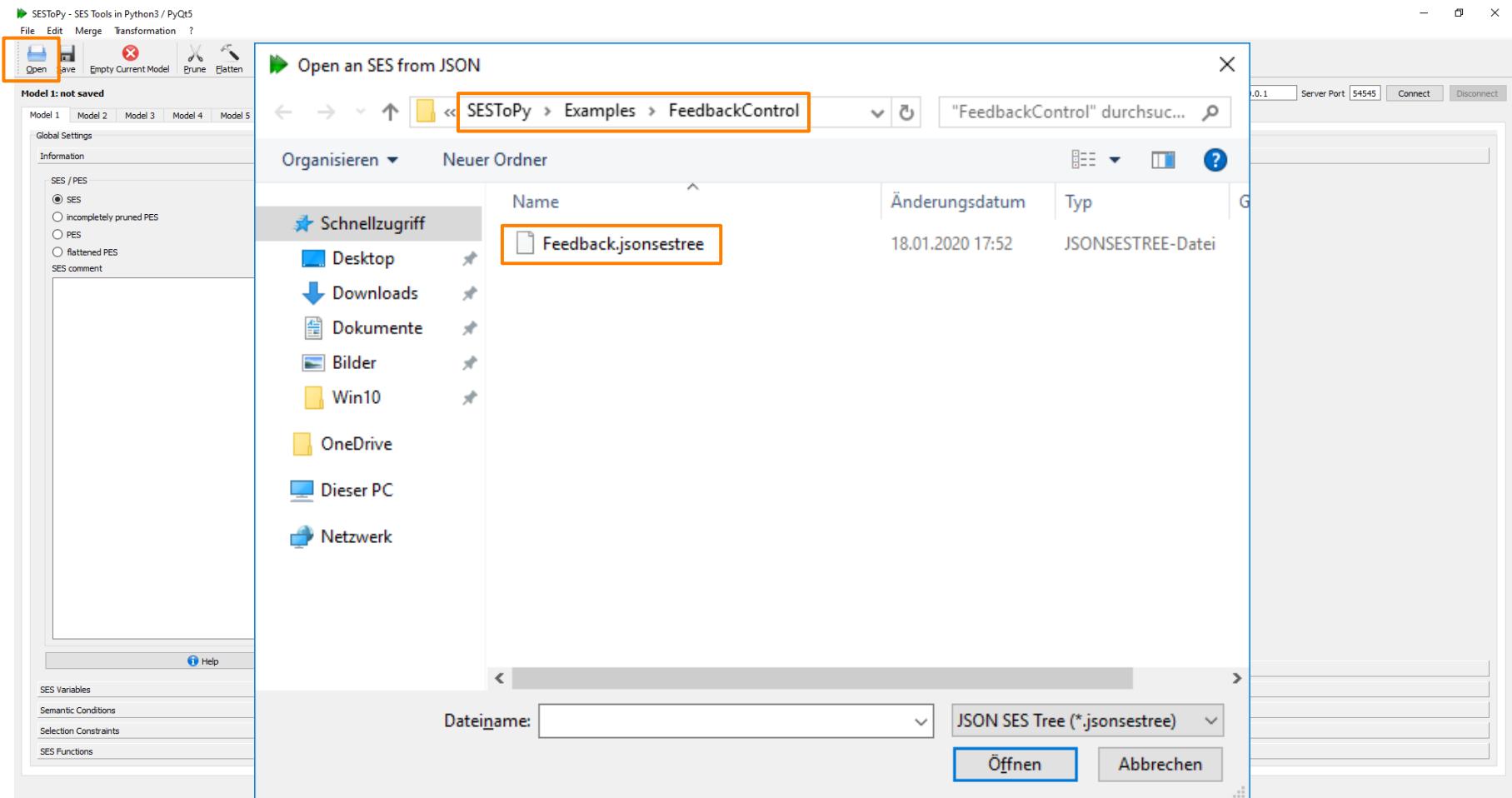
Edit Specialization Node with SESToPy

The screenshot shows the SESToPy application window with the title "SESToPy - SES Tools in Python3 / PyQt5". The main interface is divided into several sections:

- File Bar:** File, Edit, Merge, Transformation, ?
- Toolbar:** Open, Save, Empty Current Model, Prune, Flatten.
- Model Status:** Model 1: not saved
- Connection:** Connection to server program taking the XML tree: Server IP 127.0.0.1, Server Port 54545, Connect, Disconnect.
- Hierarchy Model:** A tree view showing the structure:
 - EntityRoot (Entity, uid: 1)
 - Descriptive (Spec, uid: 2)
 - Object1 (Entity, uid: 3)
 - Object2 (Entity, uid: 4)
- Global Settings:** Information, SES Variables, a table with columns Name, Value, and Comment. One row (var1, 1) is highlighted with a red border.
- Node Specific Properties:** A large panel on the right containing tabs for Default, Attributes, Aspectrule, Number of Replications, and Coupling. A specific section labeled "Specrule" is highlighted with an orange border, containing a table with columns Node, uid, Condition, result, and comment. Two rows are listed:
 - Object1 (uid: 3), Condition: var1==1, result: T
 - Object2 (uid: 4), Condition: var1==2, result: F
- Bottom Status:** 2021-02-02 / 19:05:48 / Version 2021.02.02



SESToPy Open the Feedback Example





SESToPy Feedback Example

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Feedback

Connection to server program taking the XML tree: Server IP: 127.0.0.1 Server Port: 54545 Connect Disconnect

Global Settings

Hierarchy Model

Node Entity Node

Tree

- ctrlSys
 - ctrlSysDEC
 - feedforwardCtrl
 - feedforwardCtrlSPEC
 - fc
 - fcDEC
 - tffFeedforward
 - addFeedforward
 - NONE
 - sourceSys
 - feedbackSys
 - ctrlPIDSys
 - procUnitSys
 - sourceDist
 - tDist
 - addDist

Name Value Comm

Name	Value	Comm
1	feedforward	1

Name

Value

Insert Delete Help

Semantic Conditions

Selection Constraints

SES Functions

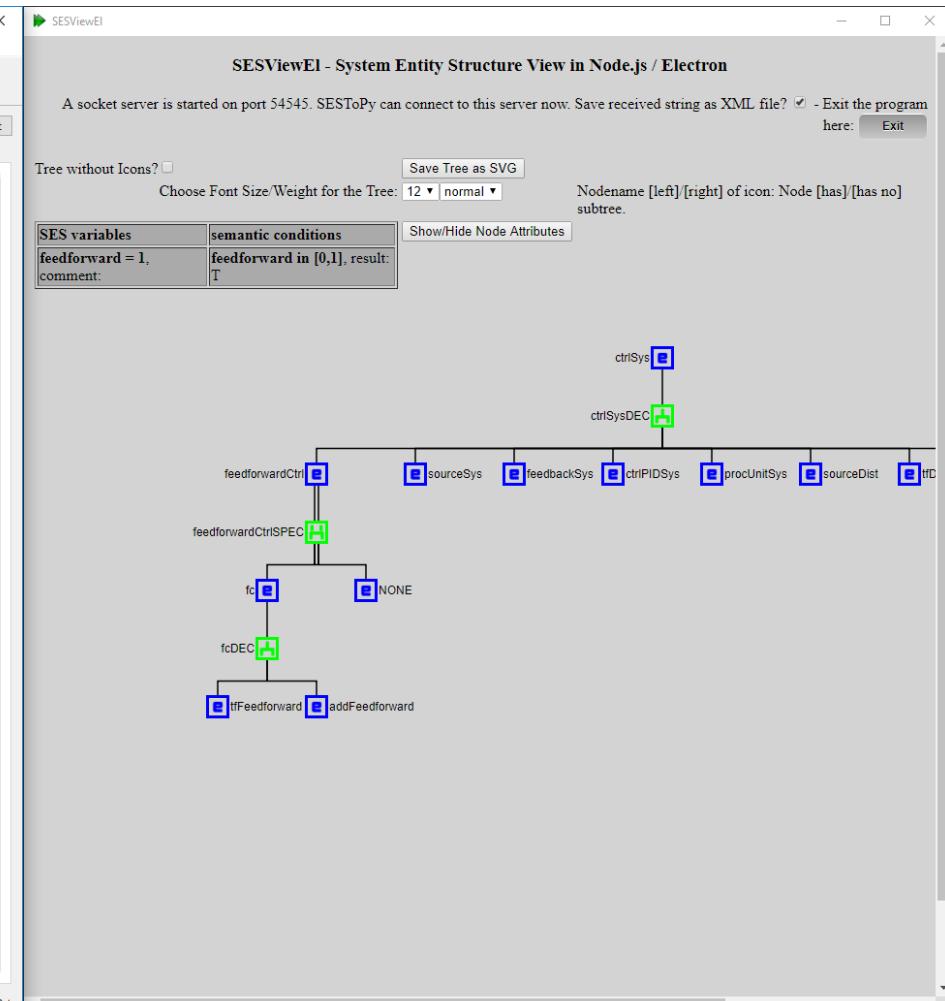
Aspects

Number of Replications

Coupling

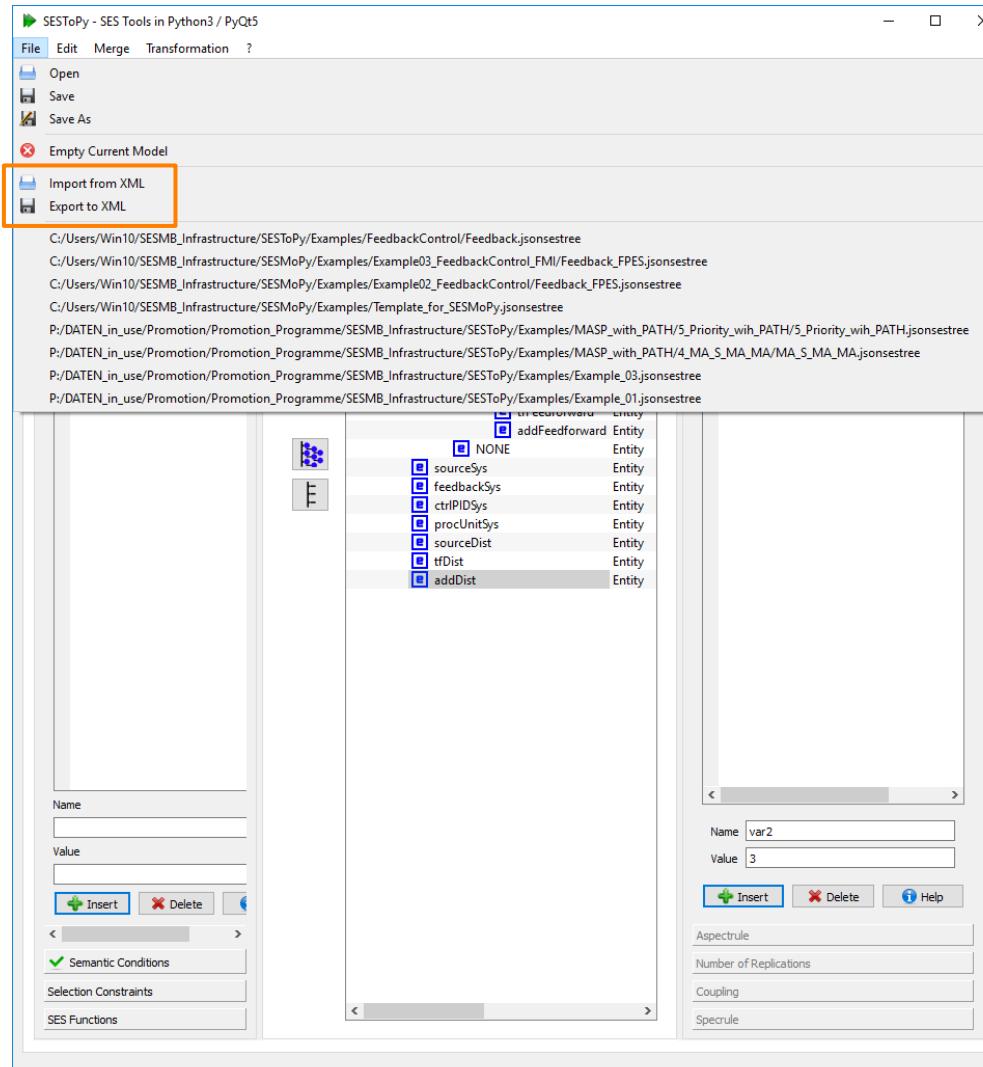
Specule

2021-02-02 / Version 2021.02.02 / File: C:/Users/Win10/SEMB_Infrastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree / Last saved: 2020-01-18 - 17:52:41





SESToPy XML Export / Import





SESToPy SES Variables

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Feedback

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information

SES Variables

Name	Value	Comment
1	feedforward	1

Name:
Value:

Semantic Conditions
Selection Constraints
SES Functions

Hierarchy Model

Entity Node

Tree

- ctrlSys
- ctrlSysDEC
- feedforwardCtrl
- feedforwardCtrlSPEC
- fc
- fcDEC
- tfFeedforward
- addFeedforward
- NONE
- sourceSys
- feedbackSys
- ctrlPIDSys
- procUnitSys
- sourceDist
- tfDist
- addDist

Type MB atr ars

Default Attributes

Name	Value	var/fun	comment
1	mb	'MB/Add'	

Name: var2
Value: 3

Aspectrule
Number of Replications
Coupling
Specrule

2021-02-02 / 19:13:14 / Version 2021.02.02 / File: C:/Users/Win10/SESMB_Infrastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree / Last saved: 2020-01-18 - 17:52:45



SESToPy Semantic Conditions

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Feedback

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings Information SES Variables

Semantic Conditions

Semantic Condition	result
1 feedforward in [0, 1]	T

Semantic Condition

Insert Delete Help

Hierarchy Model

Entity Node

Tree

- ctrlSys
- ctrlSysDEC
- feedforwardCtrl
- fc
- fcDEC
- tfFeedforward
- addFeedforward
- NONE
- sourceSys
- feedbackSys
- ctrlPIDsys
- procUnitSys
- sourceDist
- tfDist
- addDist

Type MB atr ars

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
1 mb	'MB/Add'		

Name Var2
Value 3

Insert Delete Help

Asperrule
Number of Replications
Coupling
Specrule

2021-02-02 / 19:13:55 / Version 2021.02.02 / File: C:/Users/Win10/SESBM_Infrastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree / Last saved: 2020-01-18 - 17:52:45



SESToPy Selection Rules (here Specrule)

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Feedback

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Hierarchy Model

Node: Spec Node

Tree

- ctrlSys
- ctrlSysDEC
- feedforwardCtrl
 - feedforwardCtrlSPEC
 - fc
 - tfFeedforward
 - addFeedforward
 - NONE
 - sourceSys
 - feedbackSys
 - ctrlPIDsys
 - procUnitSys
 - sourceDist
 - tfDist
 - addDist

Global Settings

Information

SES Variables

Semantic Conditions

Semantic Condition	result
1 feedforward in [0, 1]	T

Semantic Condition

Selection Constraints

SES Functions

Node Specific Properties

Default

Attributes

Aspectrule

Number of Replications

Coupling

Specrule

Node	uid	Condition	result	comment
1	fc	25	feedforward==1	T
2	NONE	29	feedforward==0	F

Help

The screenshot shows the SESToPy application window. On the left, there's a sidebar with tabs for Model 1 through Model 10. Below that are sections for Global Settings, Information, SES Variables, and Semantic Conditions. The Semantic Conditions section contains a table with one row: "1 feedforward in [0, 1]" resulting in "T". In the center, there's a "Hierarchy Model" tree view showing a nested structure of entities like ctrlSys, ctrlSysDEC, feedforwardCtrl, and various sub-components. To the right of the tree is a "Node Specific Properties" panel with tabs for Default, Attributes, Aspectrule, Number of Replications, and Coupling. A large orange box highlights the "Specrule" tab, which contains a table of selection rules. This table has columns for Node, uid, Condition, result, and comment. It lists two rules: rule 1 for node fc with uid 25 and condition "feedforward==1" resulting in T; rule 2 for node NONE with uid 29 and condition "feedforward==0" resulting in F.



SESToPy Attributes

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Feedback

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings

Information

SES Variables

Semantic Conditions

Semantic Condition	result
1 feedforward in [0, 1]	T

Hierarchy Model

Node Entity Node

Tree

- ctrlSys
- ctrlSysDEC
- feedforwardCtrl
- feedforwardCtrlSPEC
- fc
- fcDEC
- tfFeedforward
- addFeedforward
- NONE
- sourceSys
- feedbackSrc
- ctrlPIDSys
- procUnitSys
- sourceDist
- tfDist
- addDist

Attributes

Name	Value	var/fun	comment
1 mb	'MB/PID'		
2 k	1		
3 Ti	1		
4 Td	0		

Name var2
Value 3

Insert Delete Help

Aspects

Number of Replications

Coupling

Specrule

The screenshot shows the SESToPy application window. On the left, there are tabs for Model 1 through Model 10. The main area has sections for Global Settings, Information, SES Variables, and Semantic Conditions, which contains a table with one row: '1 feedforward in [0, 1]' resulting in 'T'. The central part is the Hierarchy Model, displaying a tree structure of entities like ctrlSys, ctrlSysDEC, feedforwardCtrl, etc. A specific node, 'ctrlPIDSys', is selected and highlighted with a red box. To the right of the tree is a 'Node Specific Properties' panel with a table for attributes, also highlighted with a red box. The table rows are: 1 mb 'MB/PID', 2 k 1, 3 Ti 1, and 4 Td 0. Below this are fields for Name (var2), Value (3), and buttons for Insert, Delete, and Help. At the bottom, there are sections for Aspects, Number of Replications, Coupling, and Specrule.



SESToPy Coupling List

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Feedback

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings

Information

SES Variables

Semantic Conditions

Semantic Condition	result
1 feedforward in [0, 1]	T

Hierarchy Model

Node: Aspect Node

Tree

- ctrlSys
- ctrlSysDEC
- feedforwardCtrl
- feedforwardCtrlSPEC
- fc
- fcDEC
- tfFeedforward
- addFeedforward
- NONE
- sourceSys
- feedbackSys
- ctrlPIDsys
- procUnitSys
- sourceDist
- tfDist
- addDist

Node Specific Properties

Default

Attributes

Aspectrule

Number of Replications

Coupling

source	uid	port name / type	sink	uid	port name / type
1 fc	25	u1 / SPR	tfFeedforward	27	u / SPR
2 tfFeedforward	27	y / SPR	addFeedforward	28	u1 / SPR
3 fc	25	u2 / SPR	addFeedforward	28	u2 / SPR
4 addFeedforward	28	y / SPR	fc	25	y / SPR

source node sink node

Insert Delete Help

SES function

SES function name

Specrule

2021-02-02 / 19:15:37 / Version 2021.02.02 / File: C:/Users/Win10/SESMB_Infrastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree / Last saved: 2020-01-18 - 17:52:45



SESToPy SES Function / Coupling Function

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Feedback

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings

Information

SES Variables

Semantic Conditions

Selection Constraints

SES Functions

Node Hierarchy Model

Tree

- ctrlSys
- ctrlSysDEC
- feedforwardCtrl
- feedforwardCtrlSPEC
- fc
- fcDEC
- tffeedforward
- addFeedforward
- NONE
- sourceSys
- feedbackSys
- ctrlPIDSys
- procUnitSys
- sourceDist
- tfDist
- addDist

Node Specific Properties

Default

Attributes

Aspectrule

Number of Replications

Coupling

source	uid	port name / type	sink	uid	port name / type	comment

source node sink node

Insert Delete Help

SES function

SES function name

cplfcn(feedforward, CHILDREN)

'y / SPR', 'feedforwardCtrl', 'u2 / SPR', '"]', "[feedforwardCtrl', 'y / SPR', 'procUnitSys', 'u / SPR', '"]'

Specrule

```
name | SES function
def cplfcn(feedforward, children):
    #children[0] is feedforwardCtrl
    #children[1] is sourceSys
    #children[2] is feedbackSys
    #children[3] is ctrlPIDSys
    #children[4] is procUnitSys
    #children[5] is sourceDist
    #children[6] is tfDist
    #children[7] is addDist

    cplg = []

    #fixed couplings
    cplg.append([children[1],"y / SPR",children[2],"u1"])
    cplg.append([children[2],"y / SPR",children[3],"u1"])
    cplg.append([children[4],"y / SPR",children[7],"u2"])
    cplg.append([children[7],"y / SPR",children[2],"u2"])
    cplg.append([children[5],"y / SPR",children[6],"u1"])
    cplg.append([children[6],"y / SPR",children[7],"u1"])

    #variable couplings
    if feedforward==0:
        cplg.append([children[3],"y / SPR"])
    elif feedforward==1:
        cplg.append([children[5],"y / SPR"])
        cplg.append([children[3],"y / SPR"])
        cplg.append([children[0],"y / SPR"])

    #return
    return cplg
```



SESToPy Merging

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Add SubSES Save SubSES Import SubSES from XML Export SubSES to XML

Model 1: F Model 2 Model 3 Model 4 Model 5 Model 6 Model 7 Model 8 Model 9 Model 10

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings

Information

SES Variables

Semantic Conditions

Selection Constraints

SES Functions

```
name SES function
def cplfcn(feedforward, children):
    #children[0] is feedforward
    #children[1] is sourceSys
    #children[2] is feedbackSys
    #children[3] is ctrlPIDSys
    #children[4] is procUnitSys
    #children[5] is sourceDist
    #children[6] is tfDist
    #children[7] is addDist

    cplg = []

    #fixed couplings
    cplg.append([children[1]])
    cplg.append([children[2]])
    cplg.append([children[4]])
    cplg.append([children[7]])
    cplg.append([children[5]])
    cplg.append([children[6]])

    #variable couplings
    if feedforward==0:
        cplg.append([])
    elif feedforward==1:
        cplg.append([])
        cplg.append([])

    #return
    return cplg
```

Import Delete Help

Hierarchy Model

Node Entity Node

Tree	Type	MB	atr	ars	cpl	srs
ctrlSys	Entity				x	
ctrlSysDEC	Aspect					
feedforwardCtrl	Entity					
feedforwardCtrlSPEC	Spec				x	
fc	Entity					
fcDEC	Aspect				x	
tfFeedforward	Entity	'MB/TransferFunction'	x			
addFeedforward	Entity	'MB/Add'	x			
NONE	Entity					
sourceSys	Entity	'MB/Constant'	x			
feedbackSys	Entity	'MB/Feedback'	x			
ctrlPIDSys	Entity	'MB/PID'	x			
procUnitSys	Entity	'MB/TransferFunction'	x			
sourceDist	Entity	'MB/Step'	x			
tfDist	Entity	'MB/TransferFunction'	x			
addDist	Entity	'MB/Add'	x			

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
var2	3		

Name var2 Value 3 Insert Delete Help

Aspectrule

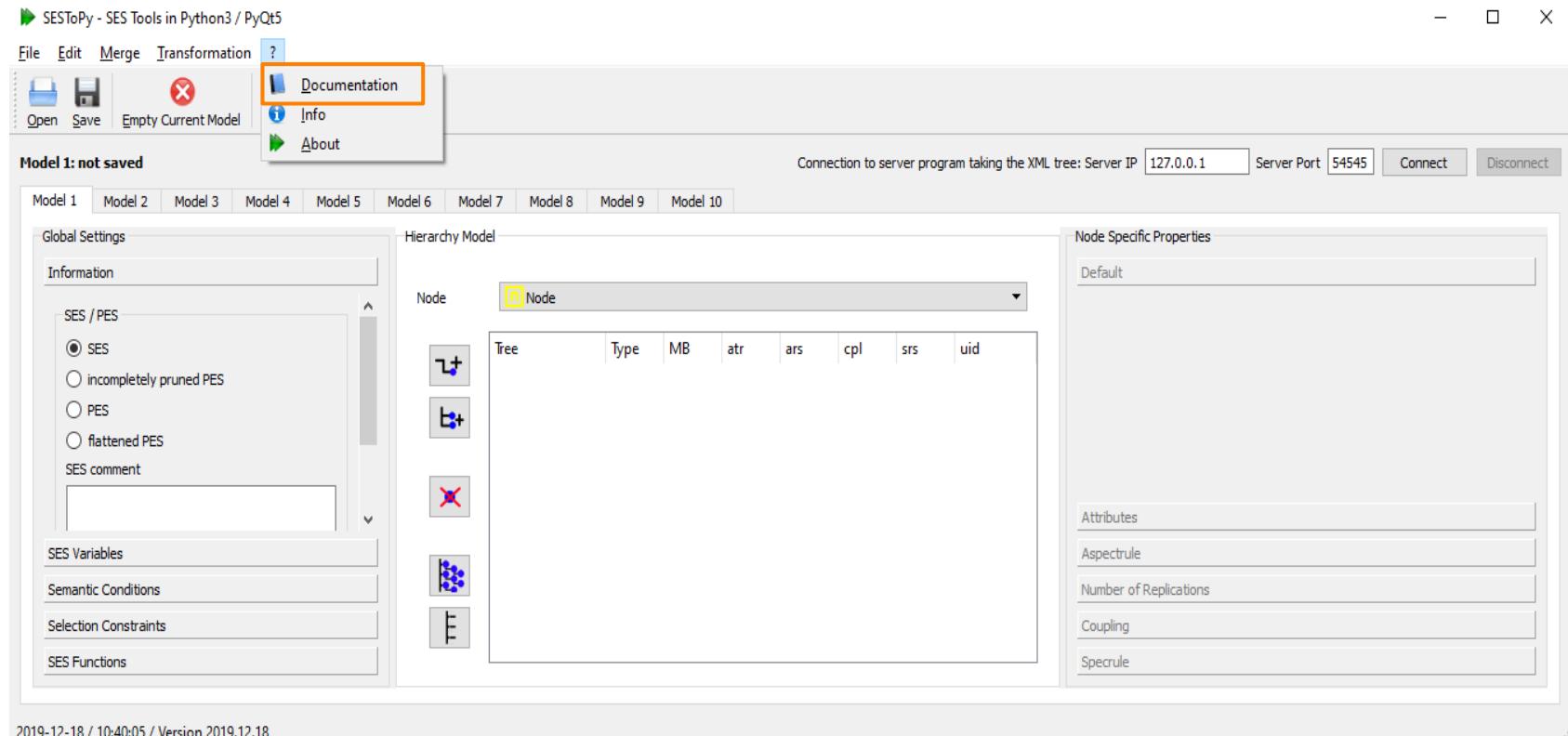
Number of Replications

Coupling Specrule

2021-02-02 / Version 2021.02.02 / File: C:/Users/Win10/SESMB_Infrastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree / Last saved: 2020-01-18 - 17:52:45



SESToPy Documentation



- See the documentation for more information



Chapter 5: Demonstration of SESMoPy Backup Slides



SESMoPy's Provisional Experimental Frame Show in SESToPy

The screenshot shows the SESToPy application window. The top menu bar includes File, Edit, Merge, Transformation, and a question mark icon. Below the menu is a toolbar with Open (highlighted with an orange box), Save, Empty Current Model, Prune, and Flatten. A status bar at the bottom displays the date and version: 2021-02-02 / 20:53:21 / Version 2021.02.02.

The main interface has several sections:

- Model 1: not saved**: A tab bar with Model 1 through Model 10.
- Global Settings**: A panel with "Information" and "SES / PES" sections. The "SES" radio button is selected. Other options include "incompletely pruned PES", "PES", and "flattened PES".
- Hierarchy Model**: A tree view showing a folder structure under "Node". The path "SESMoPy > Examples" is highlighted with an orange box. Inside "Examples", there are several subfolders: Example01, Example02_FeedbackControl, Example03_FeedbackControl_FMI, Example04_RLC_FMI, Example05_MSD_FMI, Example06_Clock_Software_Generation, and "Template_for_SESMoPy.jsonsestree".
- Node Specific Properties**: A panel on the right showing "Default" settings for aspects like Number of Replications, Coupling, and Specrule.

A connection status bar at the top right indicates "Connection to server program taking the XML tree; Server IP 127.0.0.1 Server Port 54545 Connect Disconnect".



Different Simulators & Interfaces can be set

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Blättern

Model 1: Template_for_SESMoPy

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings

Information

SES Variables

Name	Value	Comment
1	mysim	"OpenModelica"
2	myinterface	"native"

Hierarchy Model

Node Entity Node

Tree	Type	MB	atr	ars	cpl	srs	uid
exp	Entity				x		1
expDEC	Aspect					x	2
simMethod	Entity		x				4
simModel	Entity						21
expMethod	Entity		x				20

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
1	SIMULATOR	mysim	x simulator to use
2	INTERFACE	myinterface	x interface to use

Name

Value

Insert Delete Help

Semantic Conditions

Selection Constraints

SES Functions

Aspectrule

Number of Replications

Coupling

Specrule

2021-02-02 / 20:54:15 / Version 2021.02.02 / File: C:/Users/Win10/SESMoPy_Infrastructure/SESMoPy/Examples/Template_for_SESMoPy.jsonsestree / Last saved: 2020-09-17 - 21:54:54

..



Merge Feedback SES to Provisional Experimental Frame

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Add SubSES

Open Save Import SubSES from XML Export SubSES to XML

Model 1: Model 1 Model 2 Model 3 Model 4 Model 5 Model 6 Model 7 Model 8 Model 9 Model 10

Connection to server program taking the XML tree: Server IP: 127.0.0.1 Server Port: 5545 Connect Disconnect

Global Settings

Information

SES Variables

Name	Value	Comment
1 mysim	"OpenModelica"	
2 myinterface	"native"	

Hierarchy Model

Node Entity Node

Tree	Type	MB	atr	ars	cpl	srs	uid
exp	Entity						1
expDEC	Aspect						2
simMethod	Entity	x					4
ctrlSys	Entity						21
expivmethod	Entity	x					20

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
------	-------	---------	---------

Name Value var/fun comment

Insert Delete Help

Semantic Conditions

Selection Constraints

SES Functions

Name

Value

Aspectrule

Number of Replications

Coupling

Specrule

Insert Delete Help

2021-02-02 / 20:55:04 / Version 2021.02.02 / File: C:/Users/Win10/SESMoB_Infrastructure/SESMoPy/Examples/Template_for_SESMoPy.jsonsestree / Last saved: 2020-09-17 - 21:54:54



Merge Feedback SES to Provisional Experimental Frame (2)

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Template_for_SESMoPy

Model 1 Model 2 Model 3 Model 4 Model 5 Model 6 Model 7 Model 8 Model 9 Model 10

Global Settings

Information

SES Variables

Name	Value	Comment
1 mysim	"OpenModelica"	
2 myinterface	"native"	
3 feedforward	1	

Hierarchy Model

Node Entity Node

Tree

- exp
- expDEC
- simMethod
- ctrlSys
- ctrlSysDEC
- feedforwardCtrl
- feedforwardCtrlSPEC
- fc
- fcDEC
- tfFeedforward
- addFeedforward
- NONE
- sourceSys
- feedbackSys
- ctrlPIDSys
- procUnitSys
- sourceDist
- tfDist
- addDist
- expMethod

Connection to server program taking the XML tree: Server IP: 127.0.0.1 Server Port: 54545 Connect Disconnect

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
1 mb	'MB/Add'		

Name

Value

Insert Delete Help

Semantic Conditions

Selection Constraints

SES Functions

Aspectrule

Number of Replications

Coupling

Specrule

2021-02-02 / 20:55:49 / Version 2021.02.02 / File: C:/Users/Win10/SESMB_Infrastructure/SESMoPy/Examples/Template_for_SESMoPy.jsonsestree / Last saved: 2020-09-17 - 21:54:54



Different Parameters can be set

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Template_for_SESMoPy

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Model 1 Model 2 Model 3 Model 4 Model 5 Model 6 Model 7 Model 8 Model 9 Model 10

Global Settings

Information

SES Variables

Name	Value	Comment
1 mysim	"OpenModelica"	
2 myinterface	"native"	
3 feedforward	1	

Hierarchy Model

Node Entity Node

Tree

- exp
 - expDEC
 - simMethod
 - ctrlSys
 - ctrlSysDEC
 - feedforwardCtrl
 - feedforwardCtrlSPEC
 - fc
 - fcDEC
 - tfFeedforward
 - addFeedforward
 - NONE
 - sourceSys
 - feedbackSys
 - ctrlPIDSys
 - procUnitSys
 - sourceDist
 - tfDist
 - addDist
 - expMethod

Type MB atr

Default

Attributes

Name	Value	var/fun	comment
1 PARAMVARY1	"ctrlPIDSys.k=[1,5]"		template: ...
2 PARAMVARY2	"ctrlPIDSys.Ti=[1,0.5]"		template: ...

Name

Value

Insert Delete Help

Semantic Conditions

Selection Constraints

SES Functions

Node Specific Properties

Name

Value

Insert Delete Help

Aspectrule

Number of Replications

Coupling

Specrule

The screenshot shows the SESToPy application window. On the left, there's a 'Global Settings' panel with tabs for 'Information' and 'SES Variables', and a table of variables. In the center, the 'Hierarchy Model' pane displays a tree structure of entities and aspects. On the right, the 'Node Specific Properties' pane is open, showing details for 'PARAMVARY1' and 'PARAMVARY2'. The 'Attributes' table in this pane has two rows: one for 'PARAMVARY1' with value 'ctrlPIDSys.k=[1,5]' and one for 'PARAMVARY2' with value 'ctrlPIDSys.Ti=[1,0.5]'. Both rows have a 'comment' field starting with 'template: ...'. The 'Node Specific Properties' pane also includes sections for 'Aspectrule', 'Number of Replications', 'Coupling', and 'Specrule'.



Prune & Flatten for feedforward=0

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 1: Template_for_SESMoPy

Model 1 Model 2 Model 3 Model 4 Model 5 Model 6 Model 7 Model 8 Model 9 Model 10

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings

Information

SES Variables

Name	Value	Comment
1 mysim	"OpenModelica"	
2 myinterface	"native"	
3 feedforward	0	

Hierarchy Model

Node Entity Node

Tree	Type	MB	atr
exp	Entity		
expDEC	Aspect		
simMethod	Entity		x
ctrlSys	Entity		
ctrlSysDEC	Aspect		
feedforwardCtrl	Entity		
feedforwardCtrlSPEC	Spec		
fc	Entity		
fcDEC	Aspect		
tfFeedforward	Entity	'MB/TransferFunction'	x
addFeedforward	Entity	'MB/Add'	x
NONE	Entity		
sourceSys	Entity	'MB/Constant'	x
feedbackSys	Entity	'MB/Feedback'	x
ctrlPIDSys	Entity	'MB/PID'	x
procUnitSys	Entity	'MB/TransferFunction'	x
sourceDist	Entity	'MB/Step'	x
tfDist	Entity	'MB/TransferFunction'	x
addDist	Entity	'MB/Add'	x
expMethod	Entity		x

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
1 PARAMVARY1	"ctrlPIDSys.k=[1,5]"		template: ...
2 PARAMVARY2	"ctrlPIDSys.Ti=[1,0.5]"		template: ...

Name

Value

Insert Delete Help

Semantic Conditions

Selection Constraints

SES Functions

2021-02-02 / 20:57:07 / Version 2021.02.02 / File: C:/Users/Win10/SESMoPy_Infrastructure/SESMoPy/Examples/Template_for_SESMoPy.jsonsestree / Last saved: 2020-09-17 - 21:54:54



Flattened PES for feedforward=0

SESToPy - SES Tools in Python3 / PyQt5

File Edit Merge Transformation ?

Open Save Empty Current Model Prune Flatten

Model 3: not saved

Model 1 Model 2 Model 3 Model 4 Model 5 Model 6 Model 7 Model 8 Model 9 Model 10

Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545 Connect Disconnect

Global Settings

Information

SES / PES

- SES
- incompletely pruned PES
- PES
- flattened PES

SES comment

Hierarchy Model

Entity Node

Tree

Node	Type	MB	atr	ars	cpl	srs
exp	Entity					
expDEC	Aspect			x		
simMethod	Entity		x			
NONE	Entity					
sourceSys	Entity	'MB/Constant'	x			
feedbackSys	Entity	'MB/Feedback'	x			
ctrlPIDSys	Entity	'MB/PID'	x			
procUnitSys	Entity	'MB/TransferFunction'	x			
sourceDist	Entity	'MB/Step'	x			
tfDist	Entity	'MB/TransferFunction'	x			
addDist	Entity	'MB/Add'	x			
expMethod	Entity		x			

Node Specific Properties

Default

Attributes

Name	Value	var/fun	comment
1 PARAMVARY1	"ctrlPIDSys.k=[1,5]"		template: ...
2 PARAMVARY2	"ctrlPIDSys.Ti=[1,0.5]"		template: ...

Name _____

Value _____

Insert Delete Help

Aspectrule

Number of Replications _____

Coupling _____

Specrule _____

Help

SES Variables

Semantic Conditions

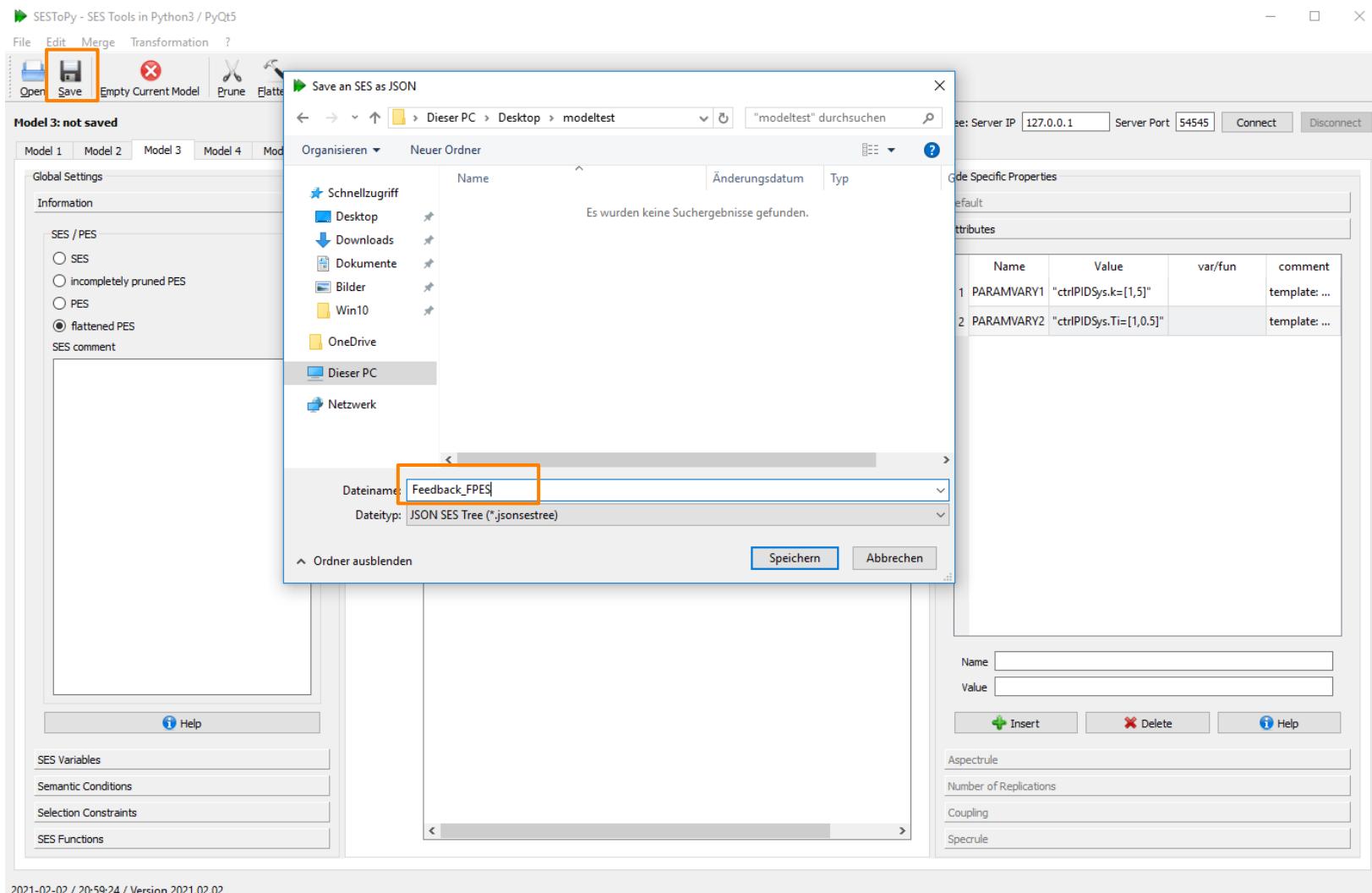
Selection Constraints

SES Functions

2021-02-02 / 20:57:58 / Version 2021.02.02

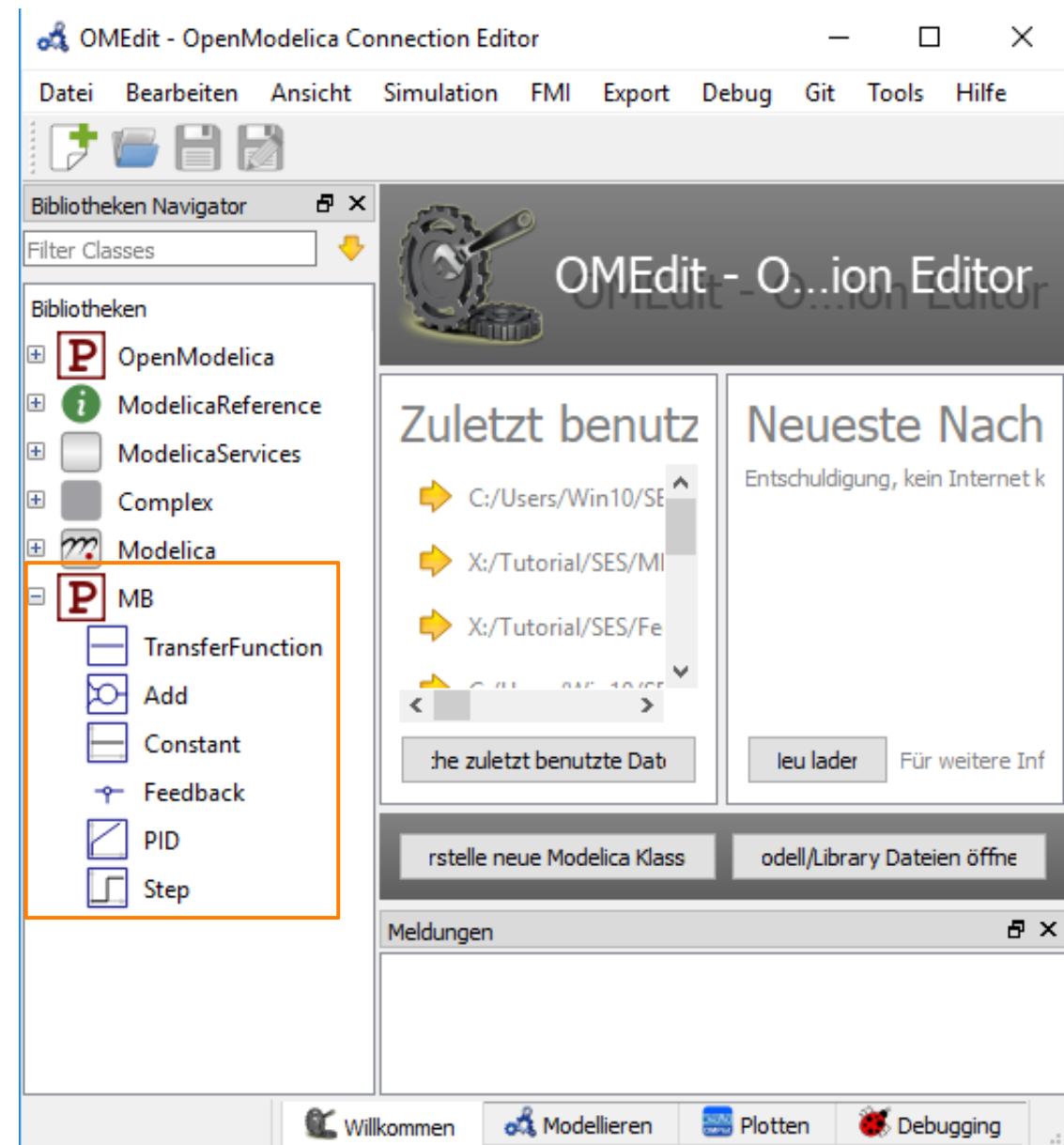


Save Flattened PES for feedforward=0



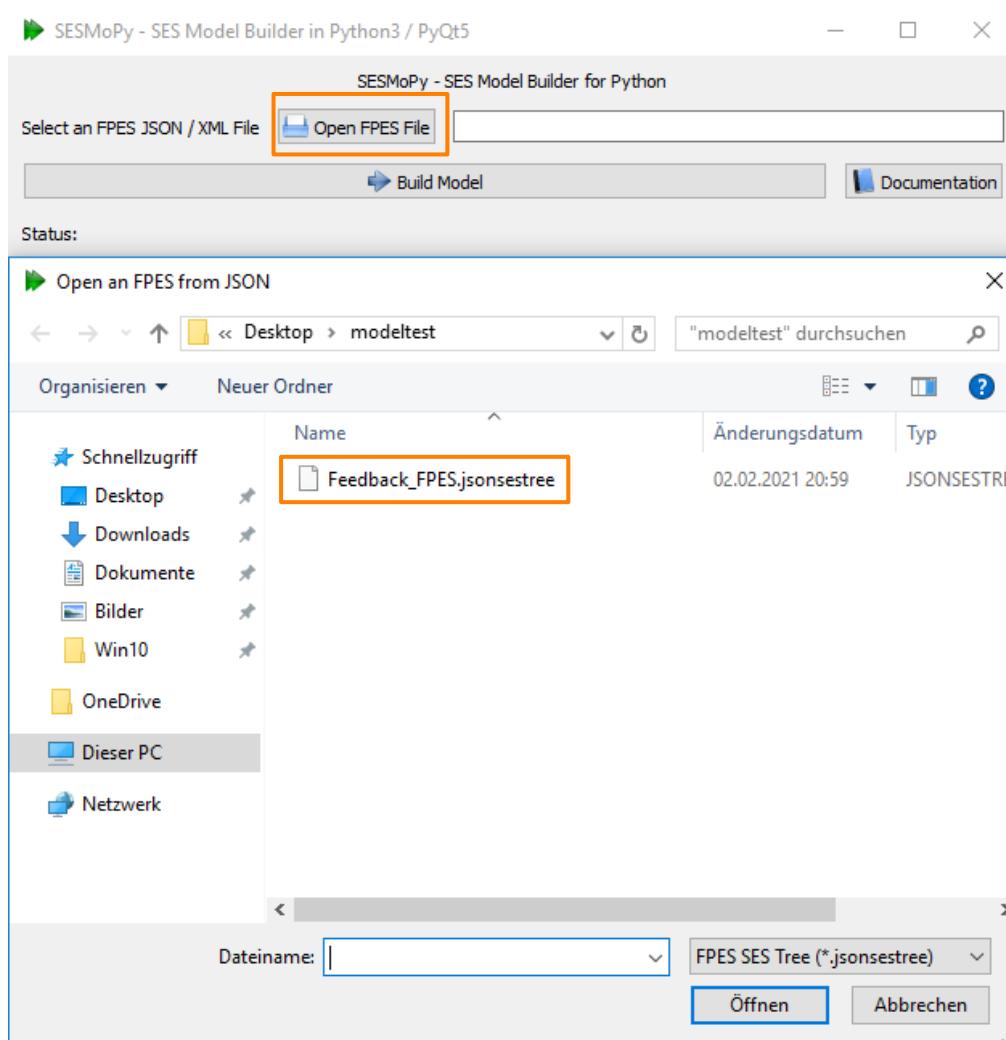


OpenModelica MB



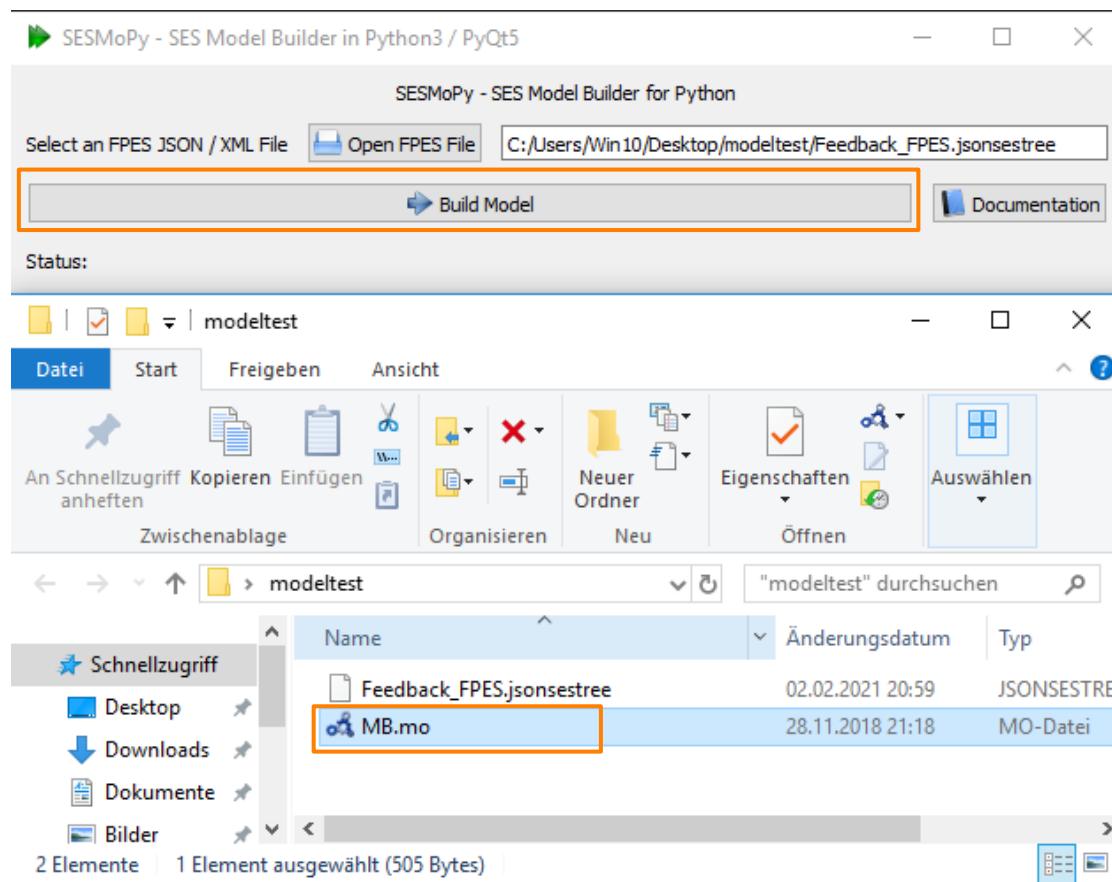


Open Flattened PES in SESMoPy





Add MB in Model folder & Build Model





Created Models

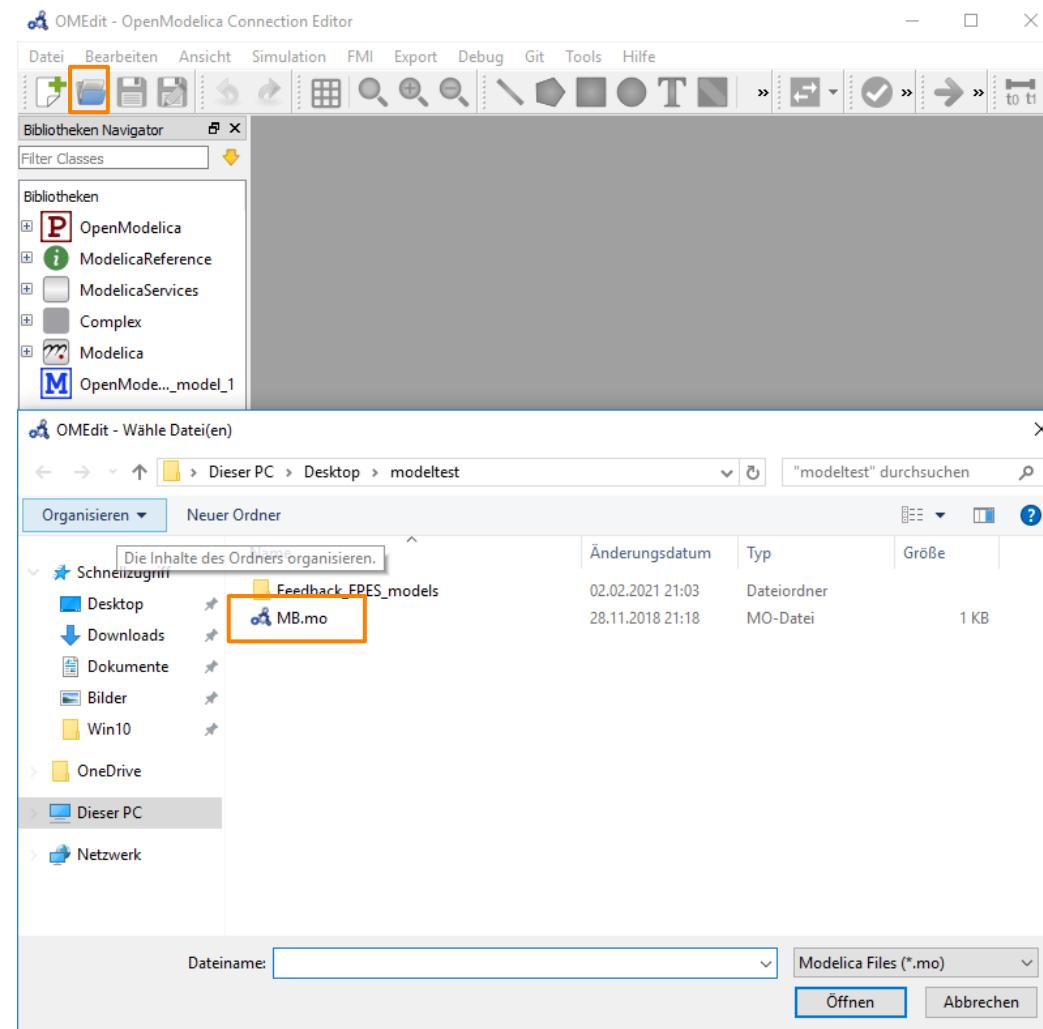
The screenshot shows a Windows File Explorer window with the following details:

- Title Bar:** Feedback_FPES_models
- Toolbar:** Includes icons for Start (D), Freigeben (R), Ansicht (A), An Schnellzugriff anheften (pin to Start), Kopieren (copy), Einfügen (paste), Organisieren (organize), Neuer Ordner (new folder), Eigenschaften (properties), and Auswählen (select).
- Address Bar:** modeltest > Feedback_FPES_models
- Content Area:** A list of files and folders:

Name	Änderungsdatum	Typ
config.txt	02.02.2021 21:03	Textdokument
MB.mo	02.02.2021 21:03	MO-Datei
OpenModelica_Feedback_FPES_model_1.mo	02.02.2021 21:03	MO-Datei
OpenModelica_Feedback_FPES_model_2.mo	02.02.2021 21:03	MO-Datei
- Left Sidebar:** Shows quick access links: Schnellzugriff, Desktop, Downloads, Dokumente, Bilder, Win10, OneDrive, Dieser PC.
- Bottom Status Bar:** Shows "4 Elemente".



Open Created Models in OpenModelica & Load MB





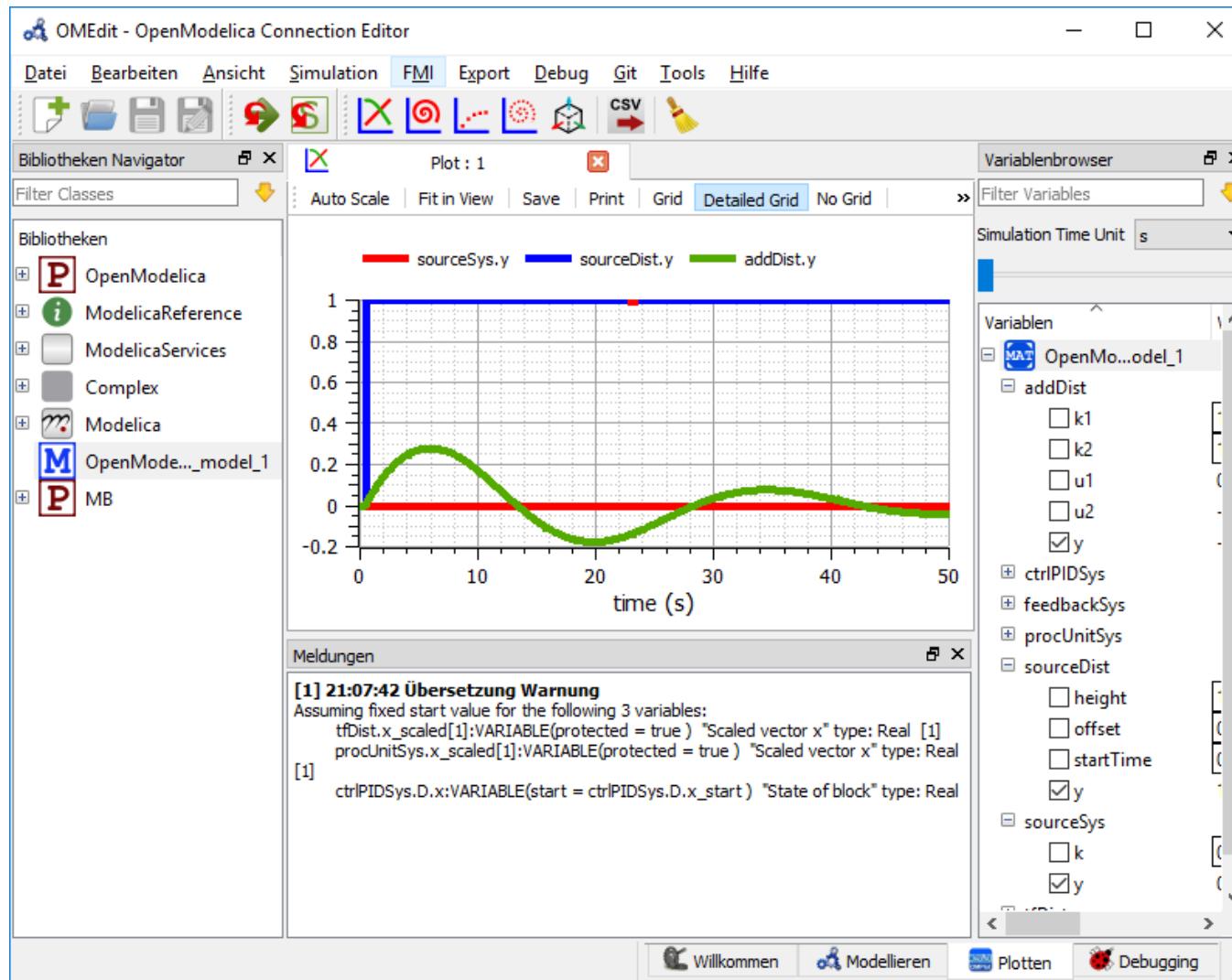
Model & Setup Simulation in OpenModelica

The screenshot displays two instances of the OMEdit - OpenModelica Connection Editor. The left instance shows the 'Simulation Setup' dialog for the model 'OpenModelica_Feedback_FPES_model_1'. The 'Stopzeit' field is highlighted with a red box and contains the value '50'. The right instance shows the model code in the 'Text View' tab.

```
model OpenModelica_Feedback_FPES_model_1
  MB.Constant sourceSys(k=0);
  MB.Feedback feedbackSys();
  MB.PID ctrlPIDSys(k=1,Ti=1,Td=0);
  MB.TransferFunction procUnitSys(b={1},a={20,1});
  MB.Step sourceDist(startTime=0.5);
  MB.TransferFunction tfDist(b={1},a={10,1});
  MB.Add addDist();
equation
  connect(sourceSys.y, feedbackSys.u1);
  connect(feedbackSys.y, ctrlPIDSys.u);
  connect(procUnitSys.y, addDist.u2);
  connect(addDist.y, feedbackSys.u2);
  connect(sourceDist.y, tfDist.u);
  connect(tfDist.y, addDist.u1);
  connect(ctrlPIDSys.y, procUnitSys.u);
end OpenModelica_Feedback_FPES_model_1;
```

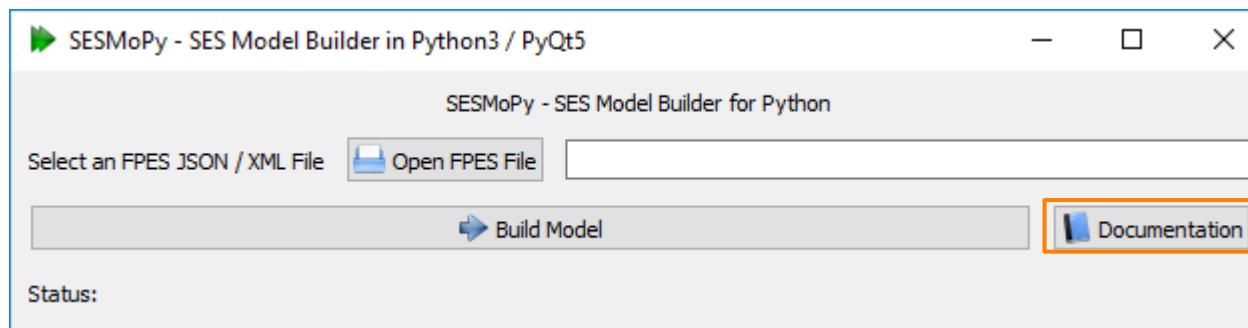


OpenModelica Simulation Results





SESMoPy Documentation



- See the documentation for more information

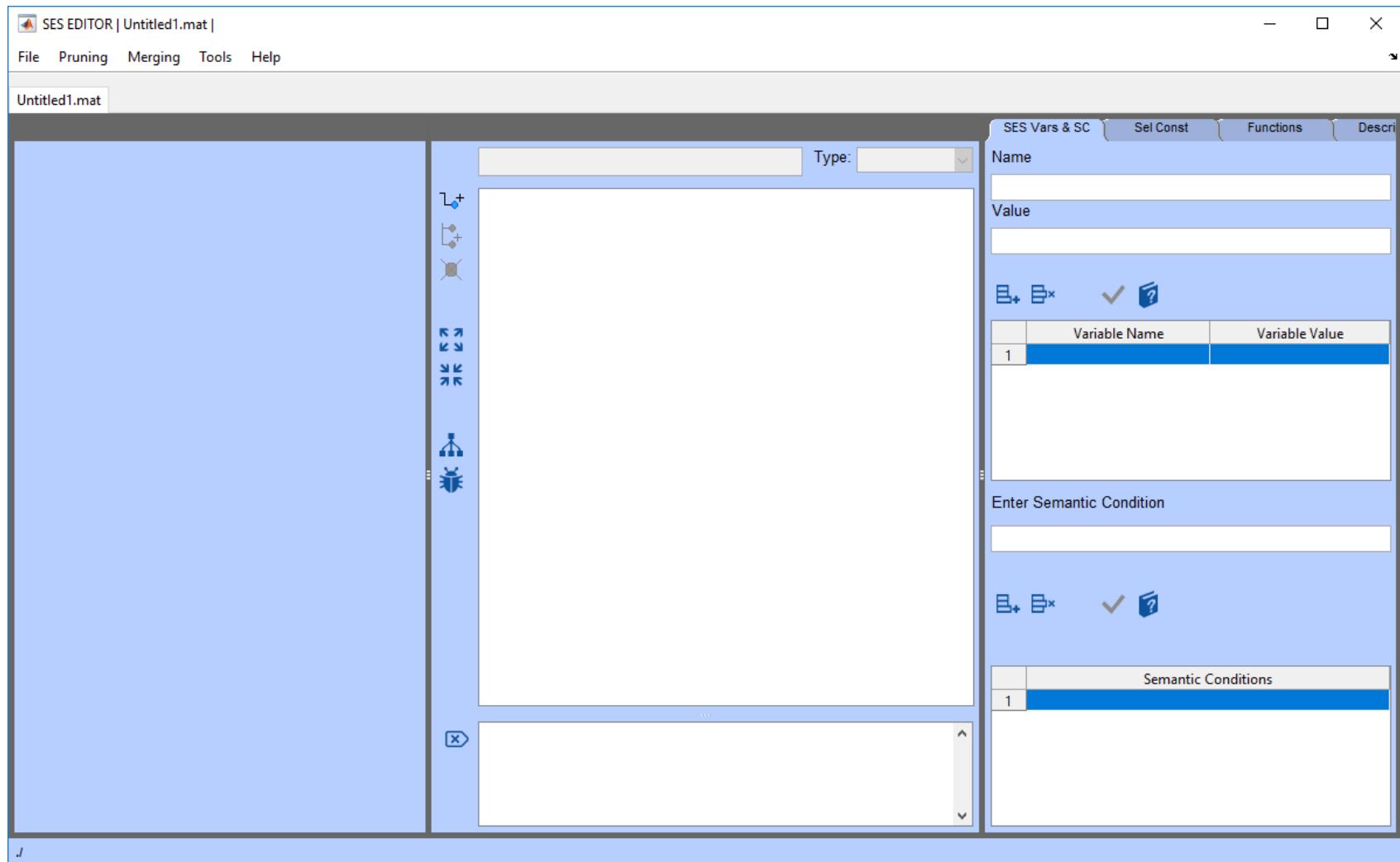


Chapter 6: Demonstration of the SES Toolbox

Matlab Backup Slides

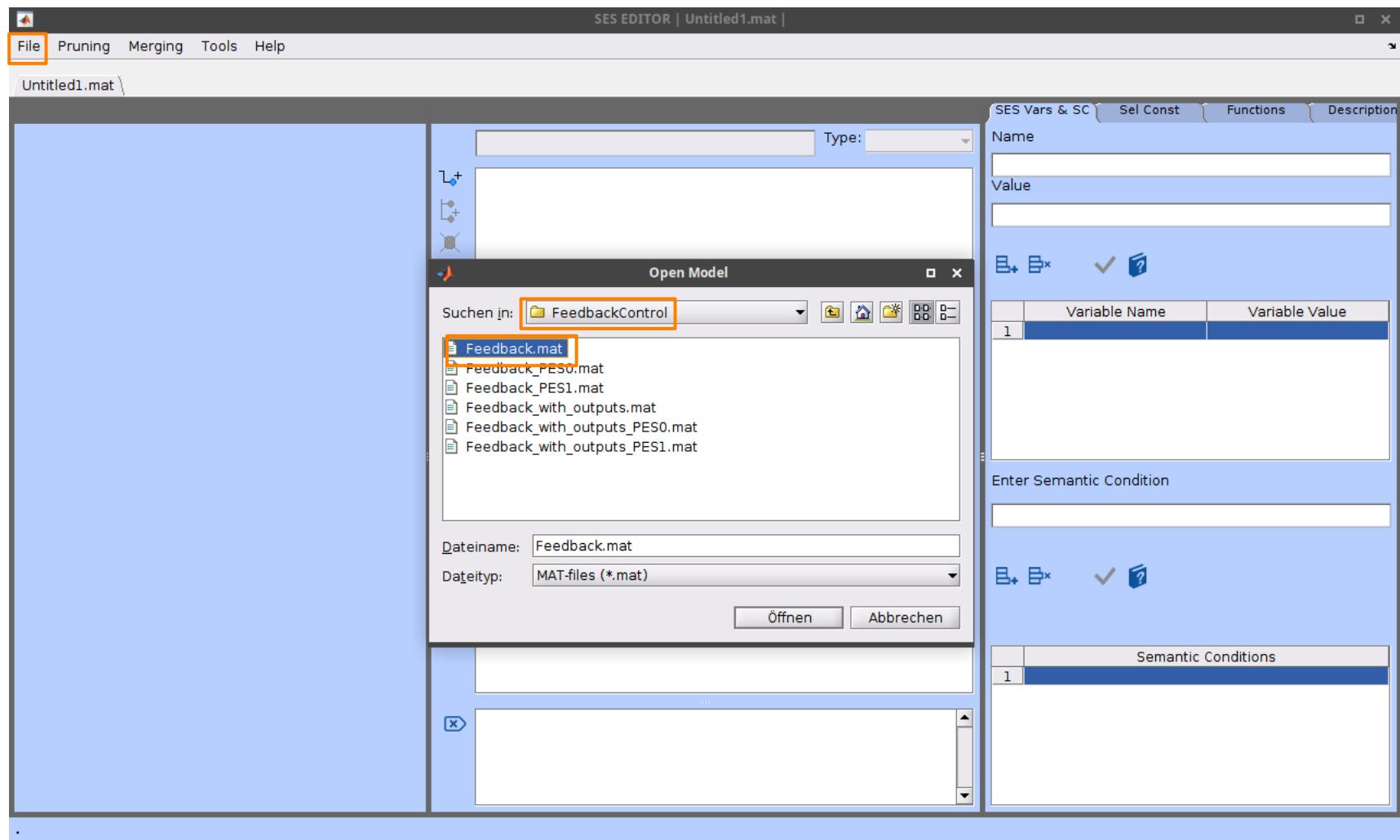


SES Tbx Matlab





Open the Feedback Example





Show Tree in SESViewEI

The image displays two software interfaces for system entity structure (SES) management:

SES EDITOR (Left): A graphical interface for editing system entities. It shows a tree structure under the entity "ctrlSys". The tree includes nodes like "ctrlSysDEC", "feedbackCtrl", "feedforwardCtrlSPEC", "fc", "fcDEC", "tfFeedforward", and "addFeedforward". A node labeled "NONE" is also present. A semantic condition table shows "feedforward = 0" with the condition "ismember(feedforward,[0,1])". A separate table lists "Semantic Conditions" with the entry "1 ismember(feedforward,[0,1])".

SESViewEI (Right): A graphical representation of the system entity structure. The tree starts with "ctrlSys", which branches into "ctrlSysDEC" and other components like "addDist", "ctrPIDSys", "feedbackForwardCtrl", "procUnitSys", and "sourceDist". "ctrlSysDEC" further branches into "feedforwardCtrlSPEC", which contains "NONE" and "fcDEC". "fcDEC" branches into "addFeedforward" and "tfFeedforward". A status bar at the bottom indicates the path: C:/Users/Win10/SESMB_Infrastructure/SES_Toolbox_Examples/FeedbackControl/

SESViewEI - System Entity Structure View in Node.js / Electron

A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? - Exit the program here: [Exit](#)

Tree without Icons Save Tree as SVG

Choose Font Size/Weight for the Tree: 10 normal

Nodename [left]/[right] of icon: Node [has no] subtree.

SES variables	semantic conditions
feedforward = 0, comment:	ismember(feedforward,[0,1]), result:



PES for feedforward=0

SES EDITOR Feedback_PES0.mat | 18-Feb-2021 14:26:14

File Pruning Merging Tools Help

Untitled1.mat \ Feedback.mat \ Feedback_PES0.mat

Attributes Description

Name:
Value:

ctrlSys Entity

- e ctrlSys
- e ctrlSysDFC
- e NONE_feedforwardCtrl
- e sourceSys
- e feedbackSys
- e ctrlPIDSys
- e procUnitSys
- e sourceDist
- e tfDist
- e addDist

SES Vars & SC Sel Const Functions Description

Name: feedforward
Value: 0

Variable Name	Variable Value
1 feedforward	0

Enter Semantic Condition

Semantic Conditions

14:26:03 Pruning is allowed!

/home/tina/Data/00-HS/Forschung/SES-Tbx_Matlab/00-new-SES-Git/more_examples/FeedbackControl/

The screenshot shows the SES Editor interface with the following details:

- Top Bar:** SES EDITOR, Feedback_PES0.mat | 18-Feb-2021 14:26:14, File, Pruning, Merging, Tools, Help.
- Left Panel:** Untitled1.mat \ Feedback.mat \ Feedback_PES0.mat, Attributes, Description, Name (empty), Value (empty), icons for add, remove, and save.
- Middle Panel:** A tree view of the system structure under "ctrlSys":
 - e ctrlSys
 - e ctrlSysDFC
 - e NONE_feedforwardCtrl (highlighted with an orange box)
 - e sourceSys
 - e feedbackSys
 - e ctrlPIDSys
 - e procUnitSys
 - e sourceDist
 - e tfDist
 - e addDist
- Right Panel:** SES Vars & SC, Sel Const, Functions, Description, Name: feedforward, Value: 0, Variable Name: feedforward, Variable Value: 0, Enter Semantic Condition, Semantic Conditions, 14:26:03 Pruning is allowed!, icons for add, remove, and save.
- Bottom:** /home/tina/Data/00-HS/Forschung/SES-Tbx_Matlab/00-new-SES-Git/more_examples/FeedbackControl/



PES for feedforward=1

SES EDITOR | Feedback_PES1.mat | 18-Feb-2021 14:26:58

File Pruning Merging Tools Help

Untitled1.mat \ Feedback.mat \ Feedback_PES0.mat \ Feedback_PES1.mat

Attributes Description

Name Value

ctrl15sys

Type: Entity

ctrlSys

fc feedforwardCtrl

fcDEC

tfFeedforward

addFeedforward

sourceSys

feedbackSys

ctrlPIDSys

procUnitSys

sourceDist

tfDist

addDist

SES Vars & SC Sel Const Functions Description

Name feedforward

Value 1

Variable Name Variable Value

1 feedforward 1

Enter Semantic Condition

Semantic Conditions

14:26:03 Pruning is allowed!

14:26:47 Pruning is allowed!

/home/tina/Data/00-HS/Forschung/SES-Tbx_Matlab/00-new/00-SES-Git/more_examples/FeedbackControl/

The screenshot shows the SES Editor interface with the following details:

- Title Bar:** SES EDITOR | Feedback_PES1.mat | 18-Feb-2021 14:26:58
- Menu Bar:** File, Pruning, Merging, Tools, Help
- Path Bar:** Untitled1.mat \ Feedback.mat \ Feedback_PES0.mat \ Feedback_PES1.mat
- Left Panel:** Attributes, Description. Contains Name and Value fields.
- Middle Panel:** A tree view of system components:
 - ctrl15sys (Type: Entity)
 - ctrlSys
 - fc feedforwardCtrl
 - fcDEC
 - tfFeedforward
 - addFeedforward
 - sourceSys
 - feedbackSys
 - ctrlPIDSys
 - procUnitSys
 - sourceDist
 - tfDist
 - addDist
 - A message box at the bottom left indicates: "14:26:03 Pruning is allowed!" and "14:26:47 Pruning is allowed!"
- Right Panel:**
 - SES Vars & SC:** A table with one row: Name feedforward, Value 1.
 - Description:** A section labeled "Enter Semantic Condition" with a text input field and a message box below it.
 - Sel Const:** A table with one row: Variable Name feedforward, Variable Value 1.
 - Functions:** A table with one row: Semantic Conditions.
 - Description:** A table with one row: 1.



Feedback Example Out Nodes

SES EDITOR | Feedback_with_outputs.mat | 13-Feb-2021 09:58:57

File Pruning Merging Tools Help

Untitled1.mat Feedback_with_outputs.mat

Attributes Description

Name Value

ctrlSys

Type: Entity

ctrlSys
ctrlSysDEC
feedforwardCtrl
fc
fcDEC
tfFeedforward
addFeedforward
NONE
sourceSys
sourceSys_out
feedbackSys
ctrlPIDSys
procUnitSys
sourceDist
sourceDist_out
tfDist
addDist
addDist_out

SES Vars & SC Sel Const Functions Desc

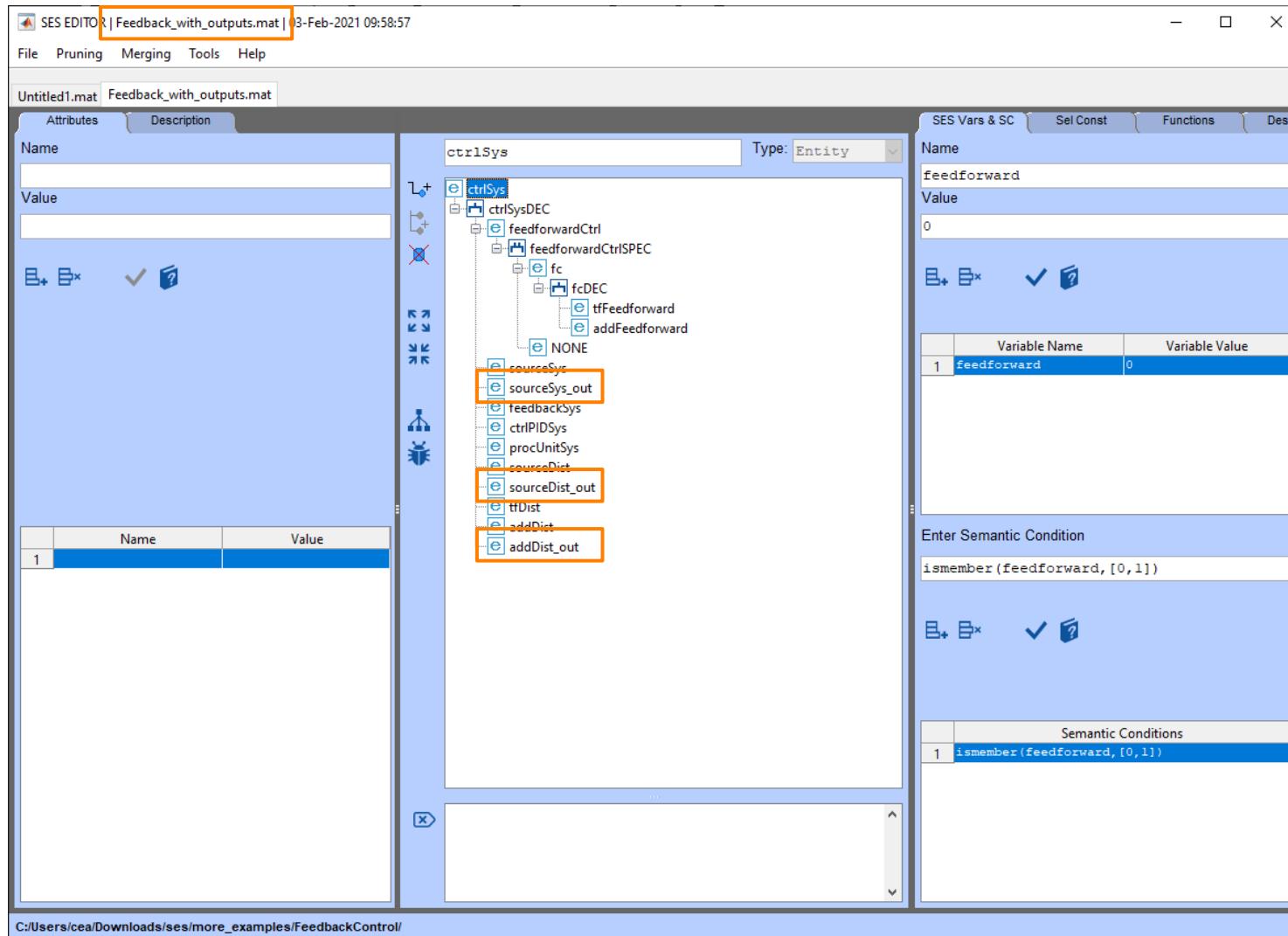
Name
feedforward
Value
0

Variable Name Variable Value
1 feedforward 0

Enter Semantic Condition
`ismember(feedforward, [0,1])`

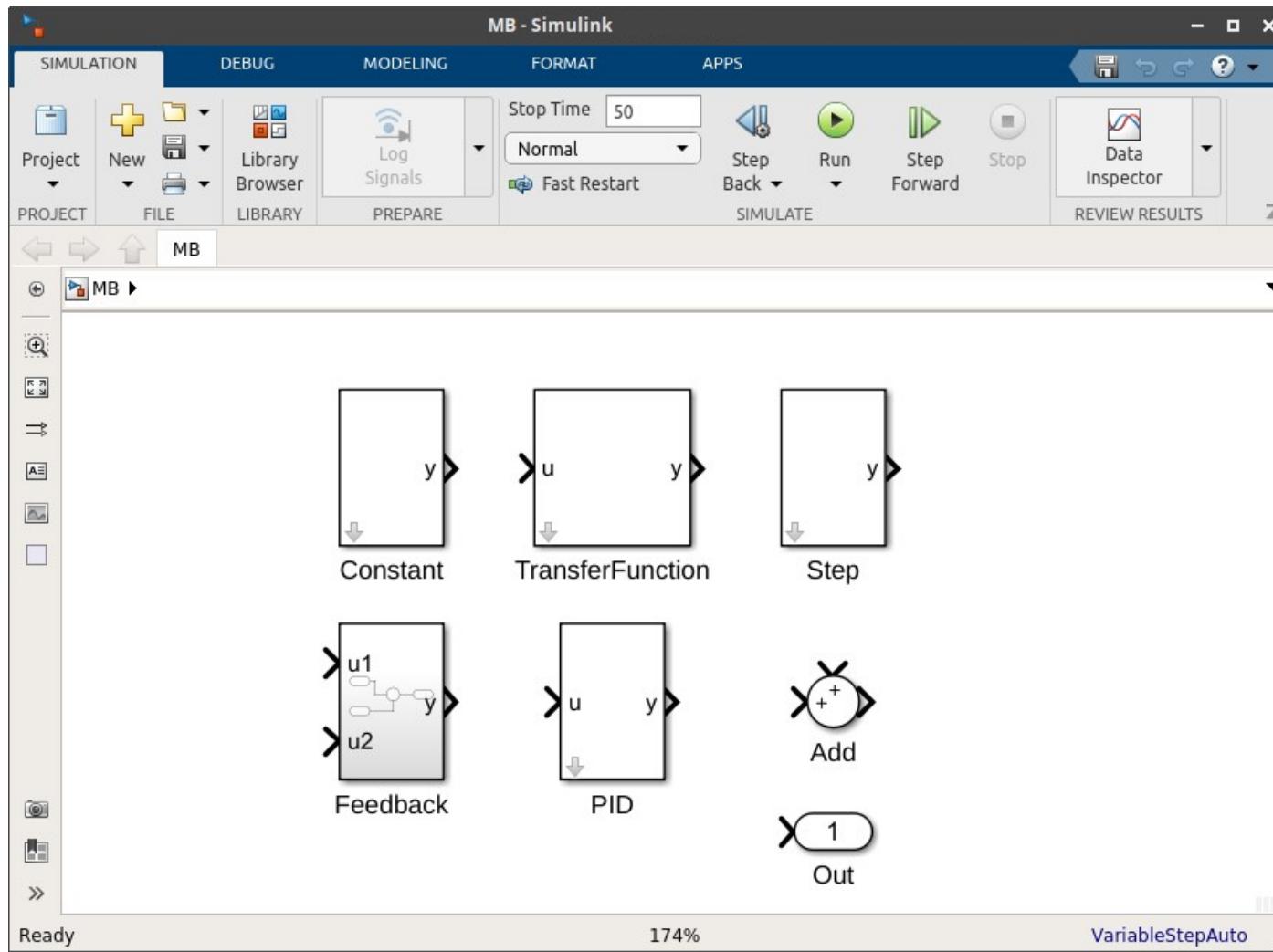
Semantic Conditions
1 `ismember(feedforward, [0,1])`

C:/Users/cea/Downloads/ses/more_examples/FeedbackControl/





Simulink MB





Feedback Example Directory

The screenshot shows the MATLAB R2020a interface. The top menu bar includes HOME, PLOTS, APPS, and various workspace management options like New Variable, Open Variable, and Clear Workspace. The toolbar below the menu has icons for New Script, New Live Script, New File, Open File, Find Files, Import Data, Save Workspace, and Clear Workspace. The Current Folder browser on the left lists files and folders: build_simulate_feedback.m, build_simulate_feedback_with_outputs.m (selected and highlighted with a blue border), cplgFunCtrlSys.m, cplgFunCtrlSys_without.m, Feedback.mat, Feedback_PES0.mat, Feedback_PES1.mat, Feedback_with_outputs.mat, Feedback_with_outputs_PES0.mat, Feedback_with_outputs_PES1.mat, help_finding_params.m, MB.slx, MB_R2018a.slx, and problems.m. The Command Window in the center shows the path: home > tina > Data > 00-HS > Forschung > SES-Tbx_Matlab > 00-new > 00-SES-Git > more_examples > FeedbackControl. Below the path, it says "New to MATLAB? See resources for [Getting Started](#)." The Command Window itself is mostly empty, with only the prompt "fx >>" visible. The Workspace browser on the right shows the variable "ans". A tooltip for "build_simulate_feedback_with_outputs.m" is displayed at the bottom left, stating: "this function sets options for model_builder and starts simulation" and showing the function signature "f build_simulate_feedback_with_outputs(mode)".



Feedback Example Function to Start Model Builder

The screenshot shows the MATLAB IDE interface with the 'EDITOR' tab selected. Two tabs are visible: 'build_simulate_feedback.m' and 'build_simulate_feedback_with_outputs.m'. The code in 'build_simulate_feedback.m' is annotated with orange boxes:

- Line 1: `function simresults = build_simulate_feedback_with_outputs(mode)` (highlighted by a box)
- Line 8: `mbOpts.backend = 'Simulink';` (highlighted by a box)
- Line 12: `% SES options` (highlighted by a box)
- Line 13: `%mbOpts.ses.file = 'Feedback.mat';`
- Line 14: `mbOpts.ses.file = 'Feedback_with_outputs.mat';`
- Line 15: `mbOpts.ses.opts = {'feedforward'};`
- Line 16: `mbOpts.ses.vals = {mode};`
- Line 18: `% simulation options` (highlighted by a box)
- Line 20: `mbOpts.sim.solver = 'ode45';`
- Line 21: `mbOpts.sim.stopTime = 50;`
- Line 22: `mbOpts.sim.maxStep = 0.1;`
- Line 24: `% plot Options` (highlighted by a box)
- Line 25: `%mbOpts.plot.showResults = false;`
- Line 26: `mbOpts.plot.sizeX = 500;`
- Line 27: `mbOpts.plot.sizeY = 500;`
- Line 28: `mbOpts.plot.nPlots = 3;`
- Line 29: `mbOpts.plot.title = 'Simulation Results';`
- Line 30: `mbOpts.plot.xLabel = {'t [s]', 't [s]', 't [s]'};`
- Line 31: `mbOpts.plot.yLabel = {'controlled variable', 'disturbance', 'setpoint'};`
- Line 36: `simresults = modelBuilder(mbOpts);` (highlighted by a box)
- Line 37: `return`

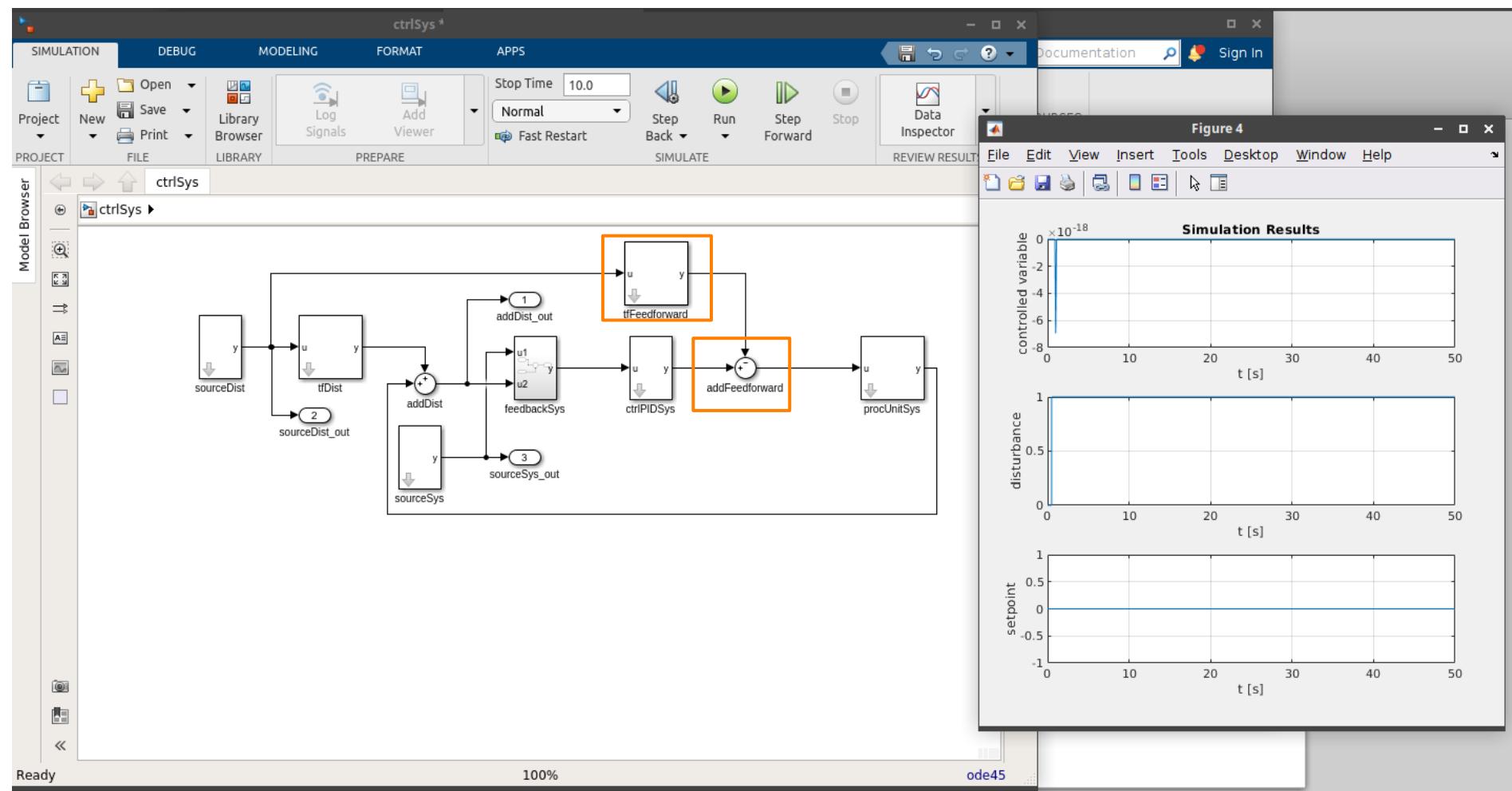


Feedback Example Start Model Building

The screenshot shows the MATLAB R2020a interface. The Command Window displays the command `simresults = build_simulate_feedback_with_outputs(1)`, which is highlighted with a red box. The Current Folder browser shows a list of files in the directory `home/tina/Data/00-HS/Forschung/SES-Tbx_Matlab/00-new/00-SES-Git/more_examples/FeedbackControl`. The file `build_simulate_feedback_with_outputs.m` is selected in the browser. A preview pane for this file shows its documentation: "this function sets options for model_builder and starts simulation" and the function signature `build_simulate_feedback_with_outputs(mode)`.



Feedback Example Simulink Model for feedforward=1 and Results





Challenges in Development of the Matlab Model Builder

- Convention for port names used in the SES couplings: PNNN for 'normal' ports, CNN for entity ports.
→ port names are not the same as in model base.
- Parameters of Simulink blocks can be set via a mask of a block in the MB.
- Variables and data types defined in the SES need to match mask variables/Simulink parameters exactly.
- Parameter variations can be achieved via SES variables.
- Simulink needs Out blocks to return simulation results.
- Until now, they must be added to the SES, although not part of the system.



Chapter 7: FMI & FMI Model Specification Backup Slides



Model Building – FMI Usage



Model Building – FMI Usage

- One model FMU for each parameter variation of one structure variant



Model Building – FMI Usage

- One model FMU for each parameter variation of one structure variant
- Model FMU includes all logic and parameters
 - Completely simulator independent



Model Building – FMI Usage

- One model FMU for each parameter variation of one structure variant
- Model FMU includes all logic and parameters
 - Completely simulator independent
- **Not needed:**
 - Simulator specific couplings (portnames, porttypes)
 - Simulator specific parameterization
 - Simulator specific syntax

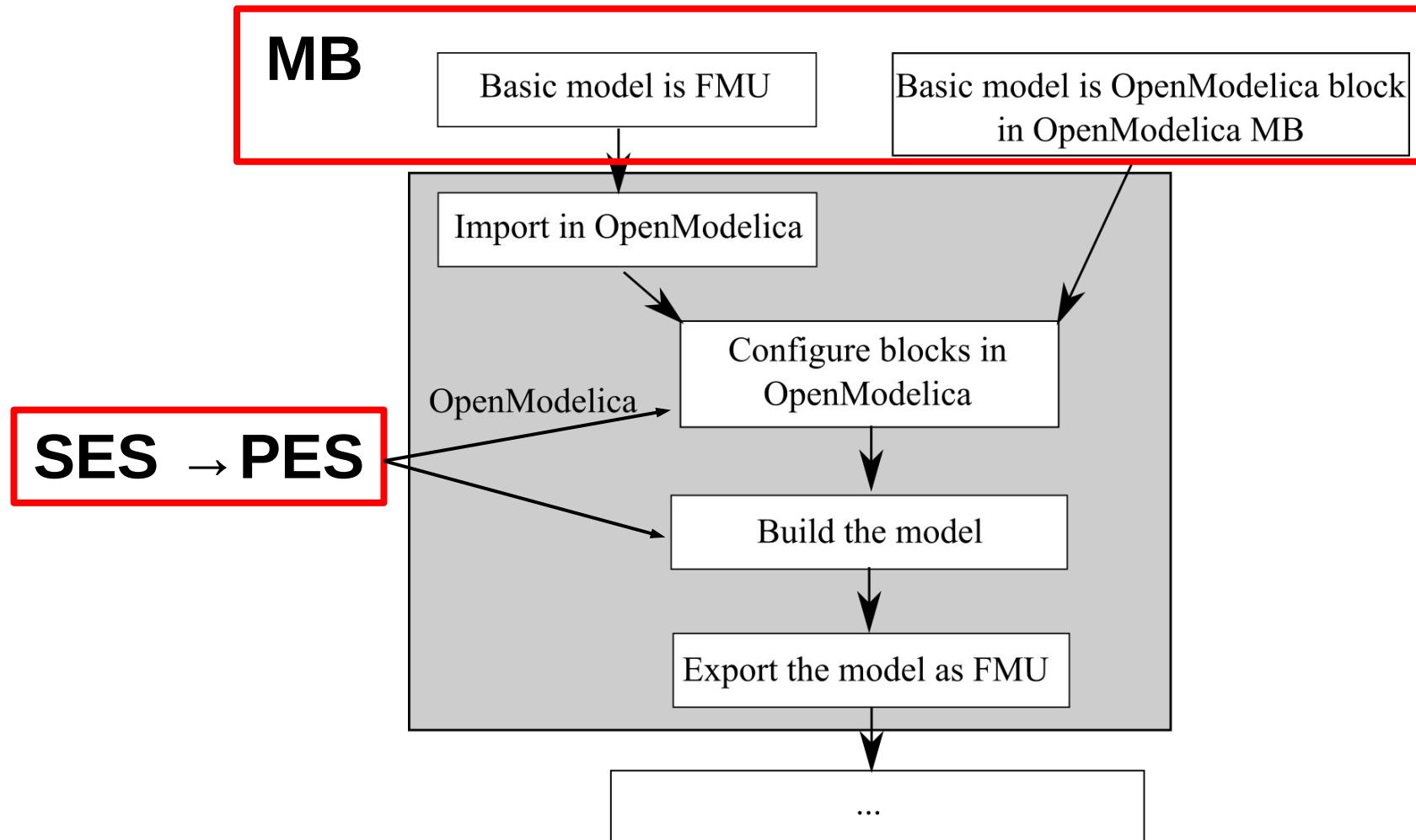


Model Building – FMI Usage

- One model FMU for each parameter variation of one structure variant
- Model FMU includes all logic and parameters
 - Completely simulator independent
- **Not needed:**
 - Simulator specific couplings (portnames, porttypes)
 - Simulator specific parameterization
 - Simulator specific syntax
- **Needed:**
 - Simulator needs to support FMI

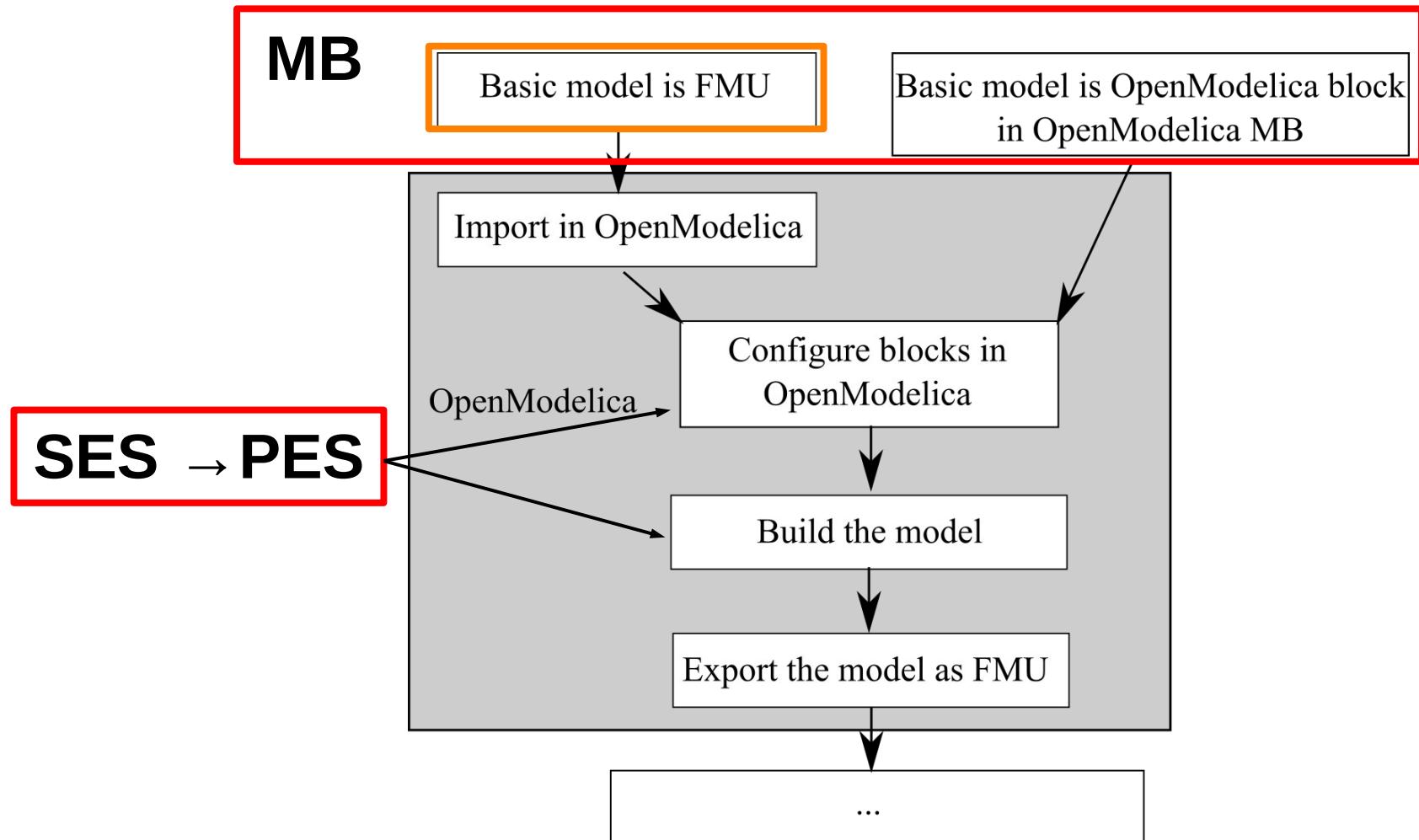


Model Building Using FMI (state of the art)



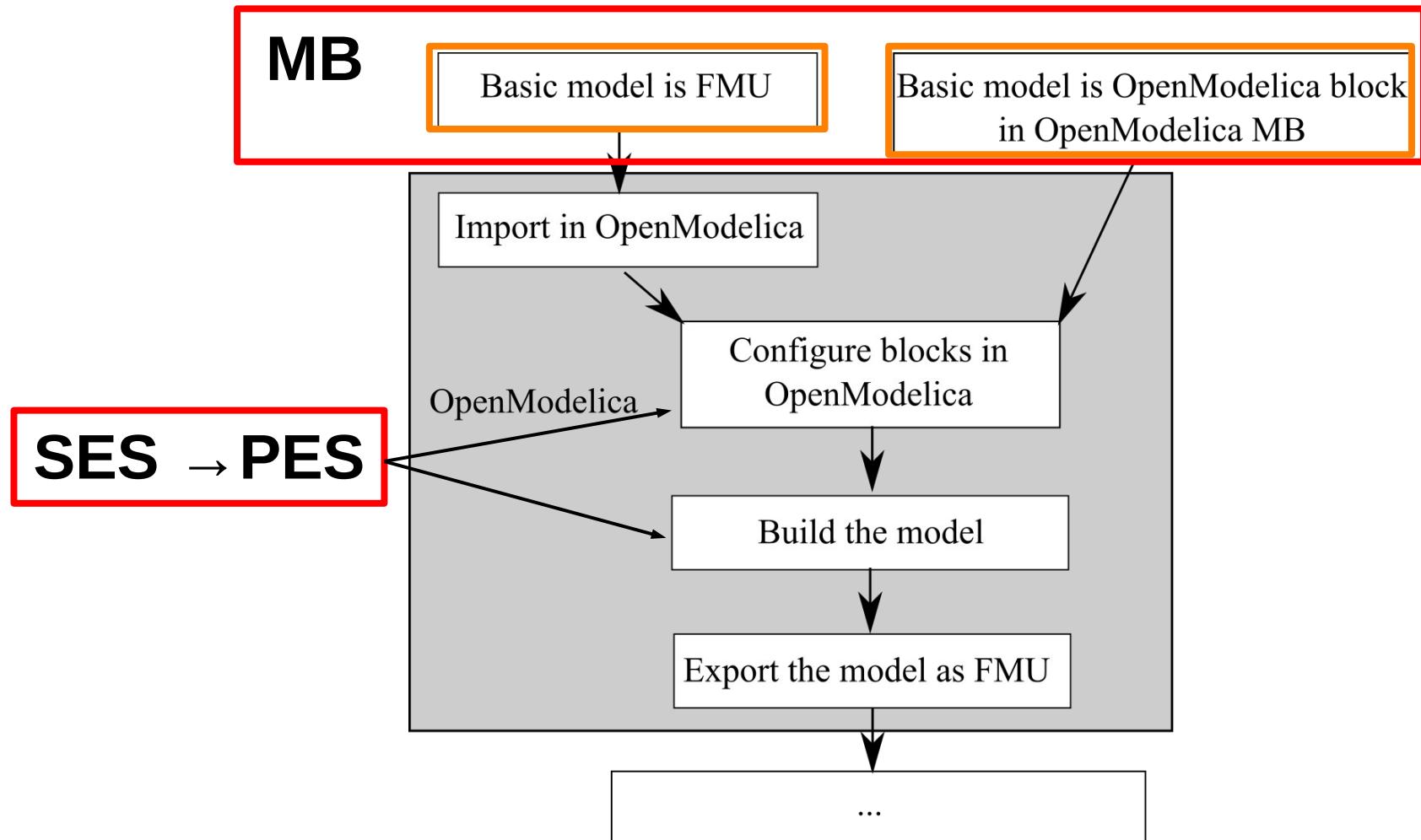


Model Building Using FMI (state of the art)



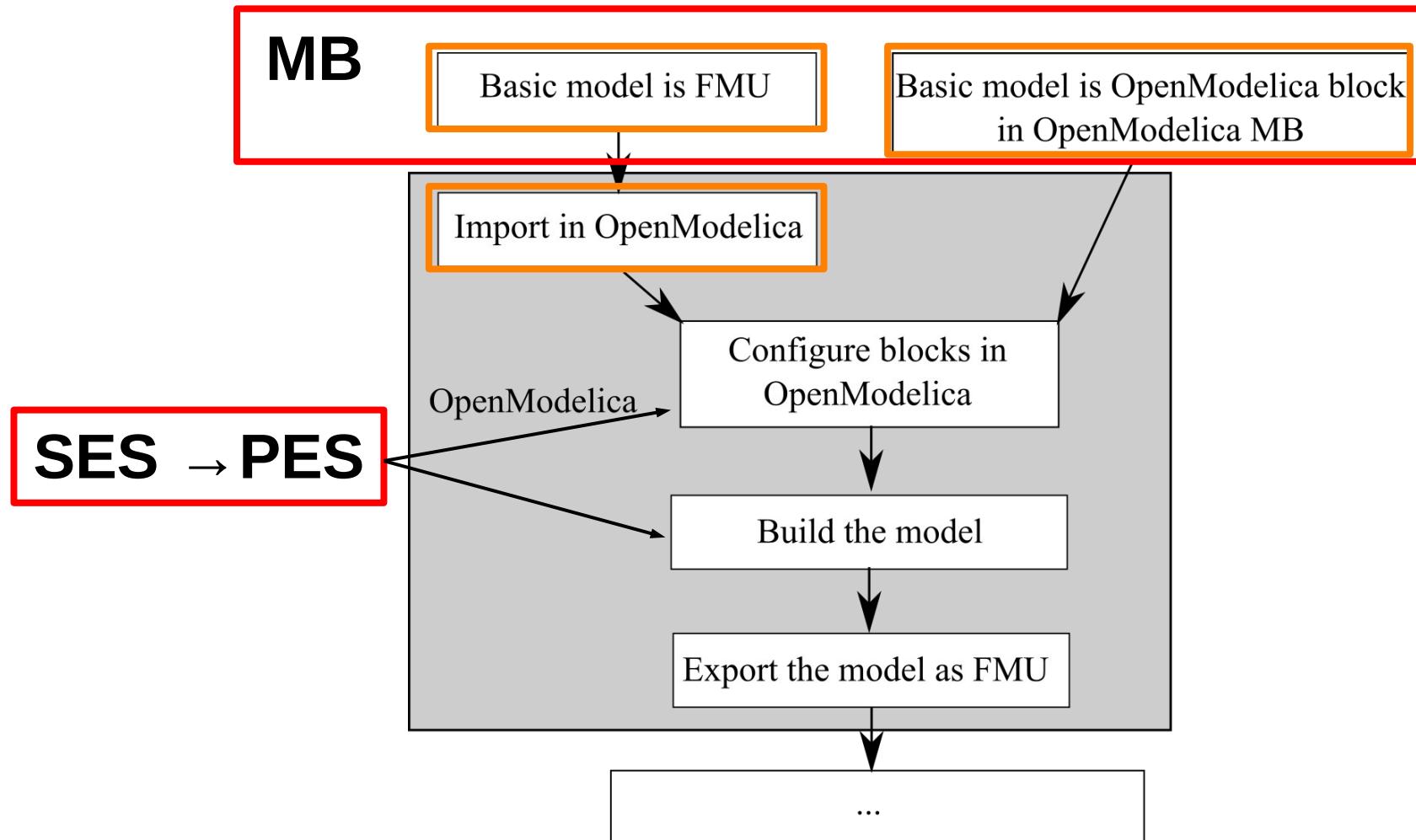


Model Building Using FMI (state of the art)



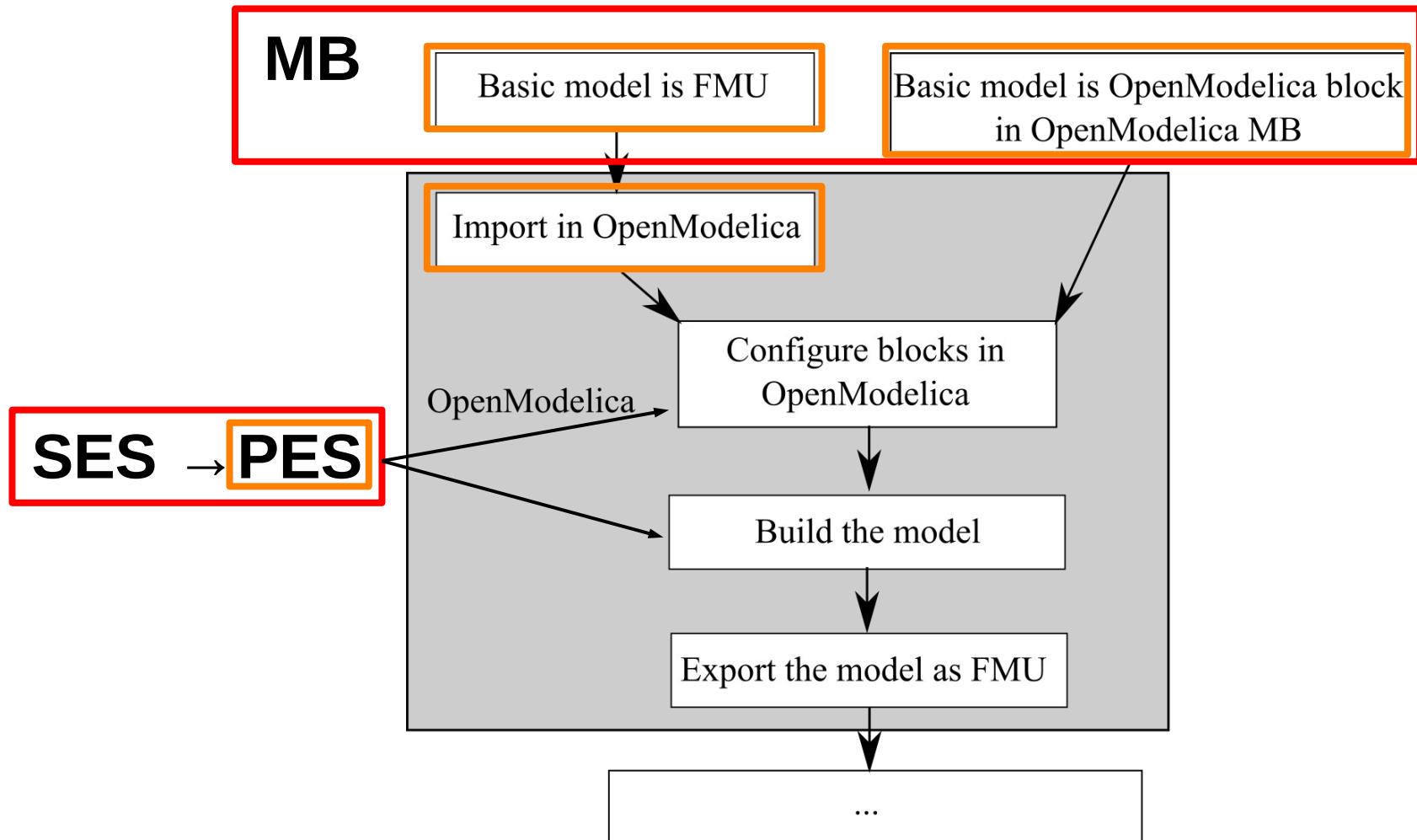


Model Building Using FMI (state of the art)



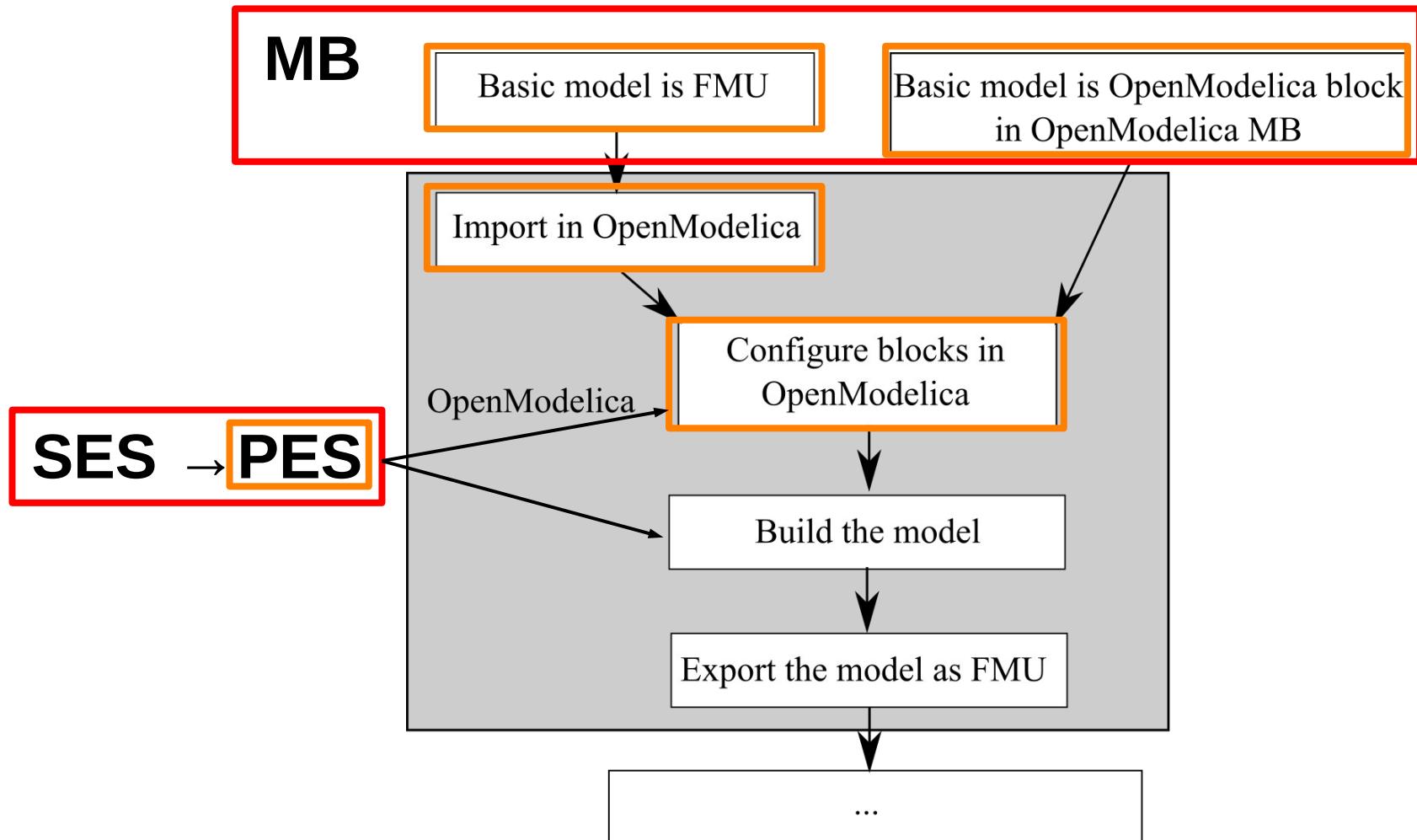


Model Building Using FMI (state of the art)



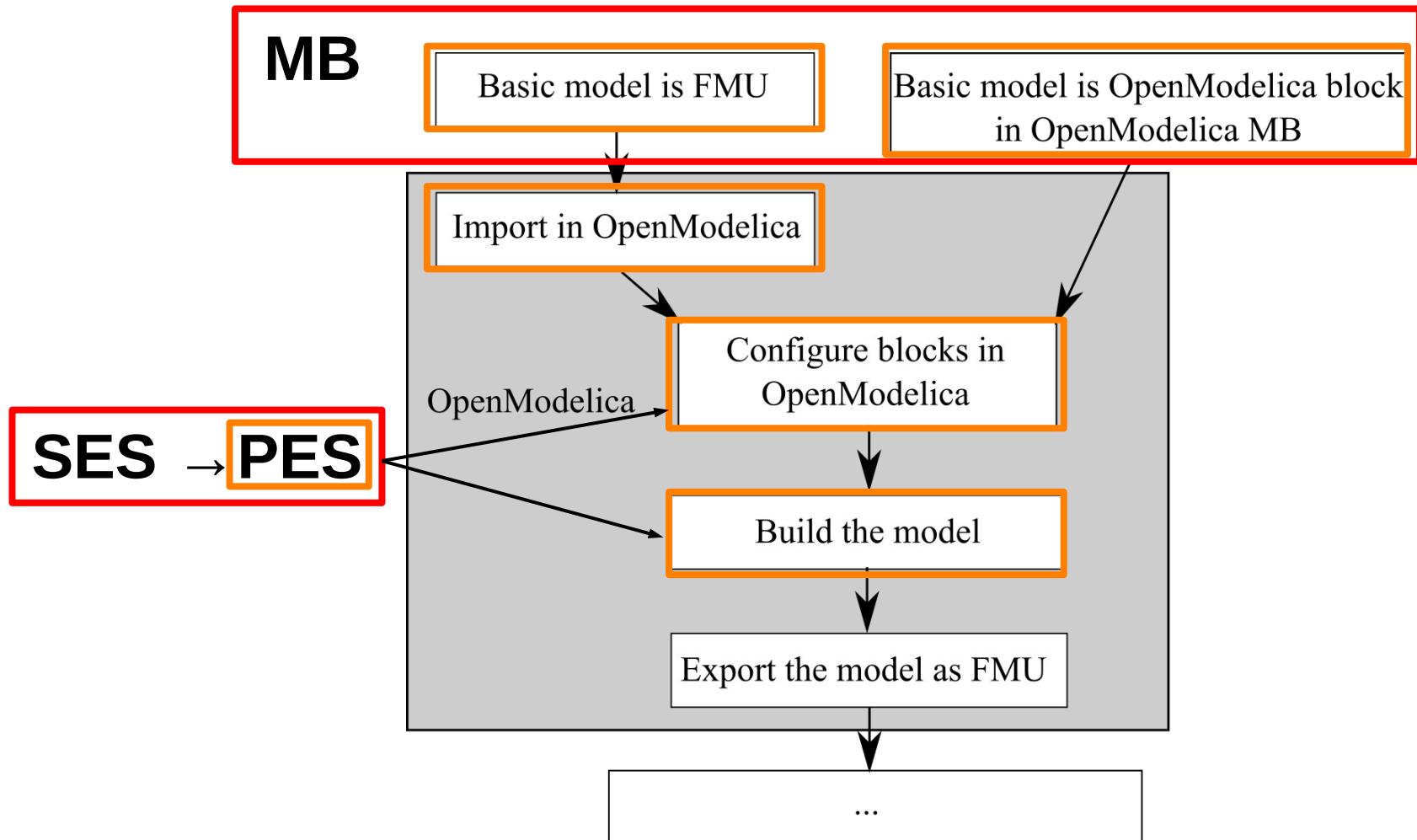


Model Building Using FMI (state of the art)



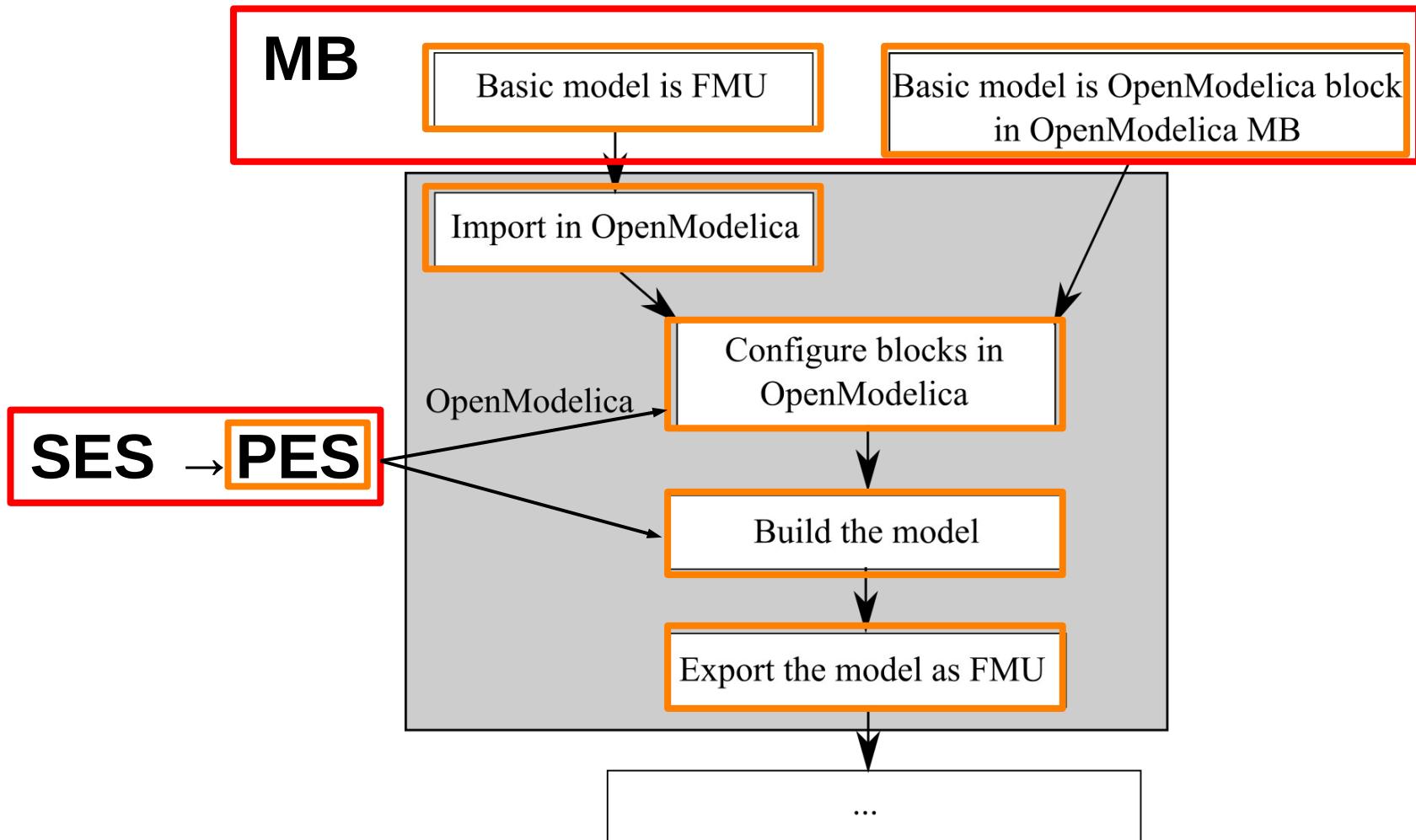


Model Building Using FMI (state of the art)



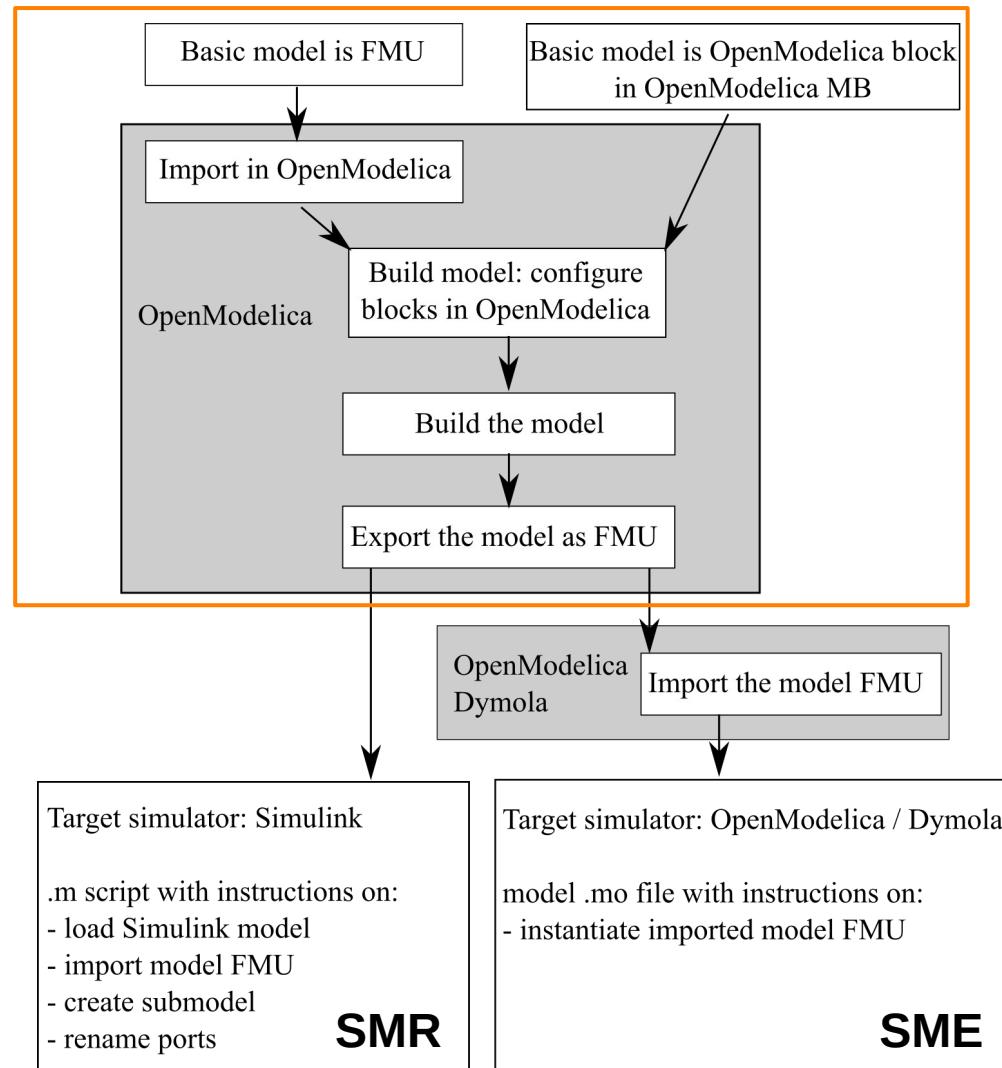


Model Building Using FMI (state of the art)





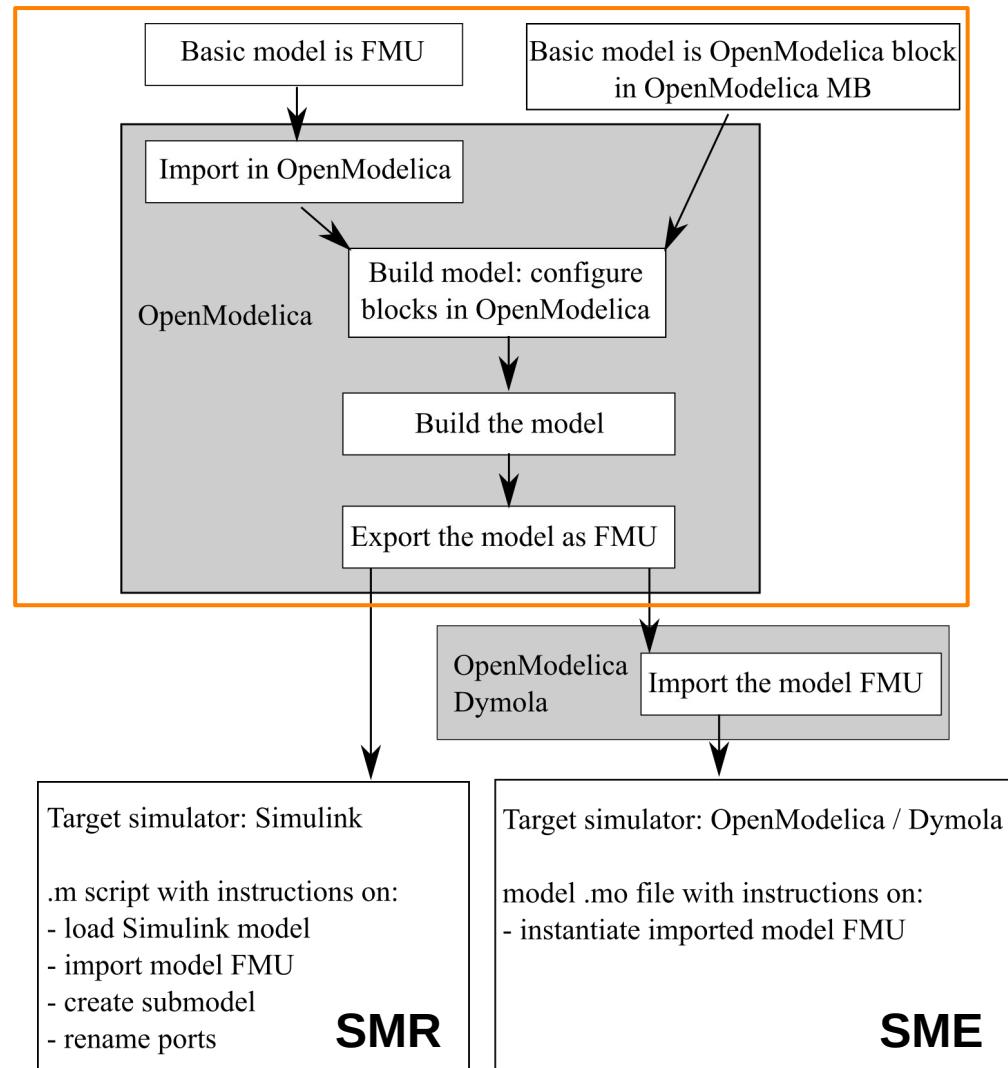
Model Building Using FMI (2) (state of the art)





Model Building Using FMI (2) (state of the art)

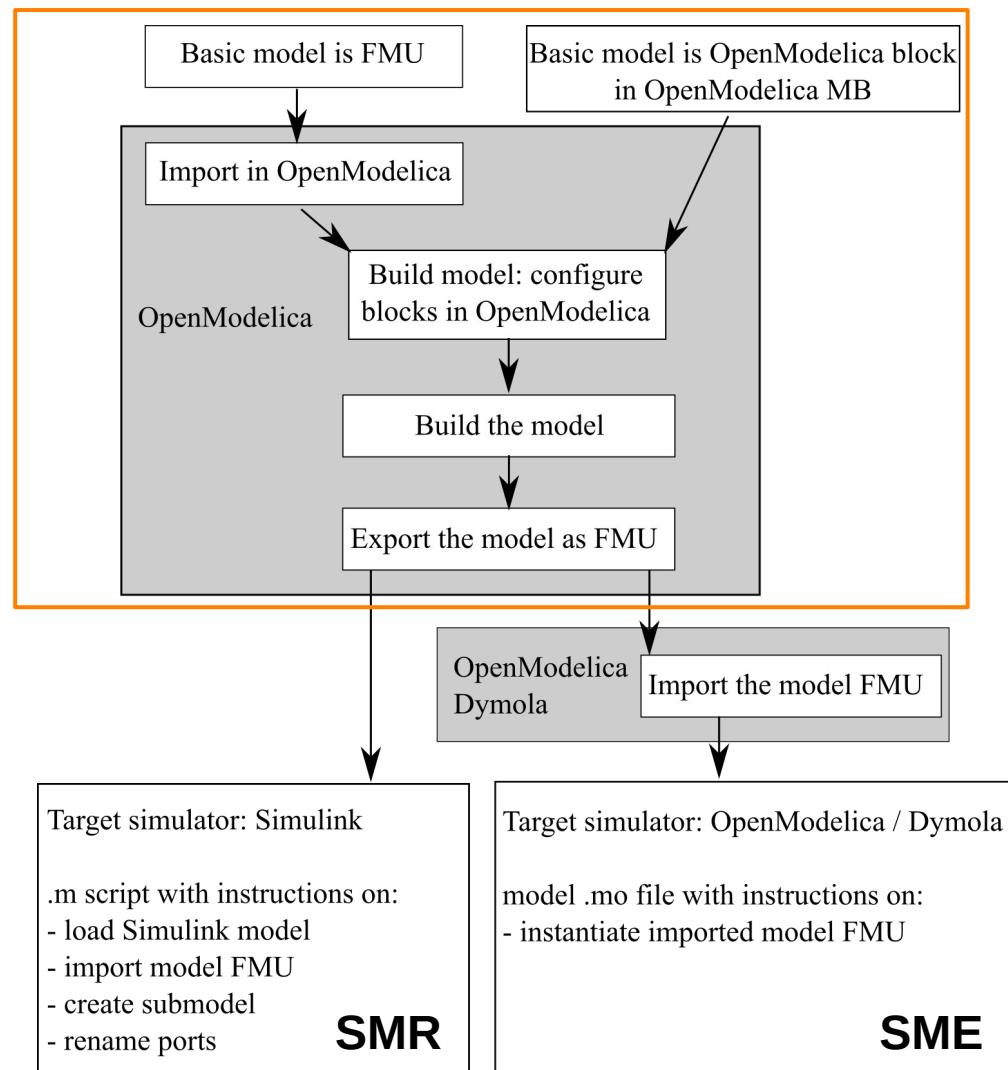
- Create simulator specific instructions on how to execute the model FMU





Model Building Using FMI (2) (state of the art)

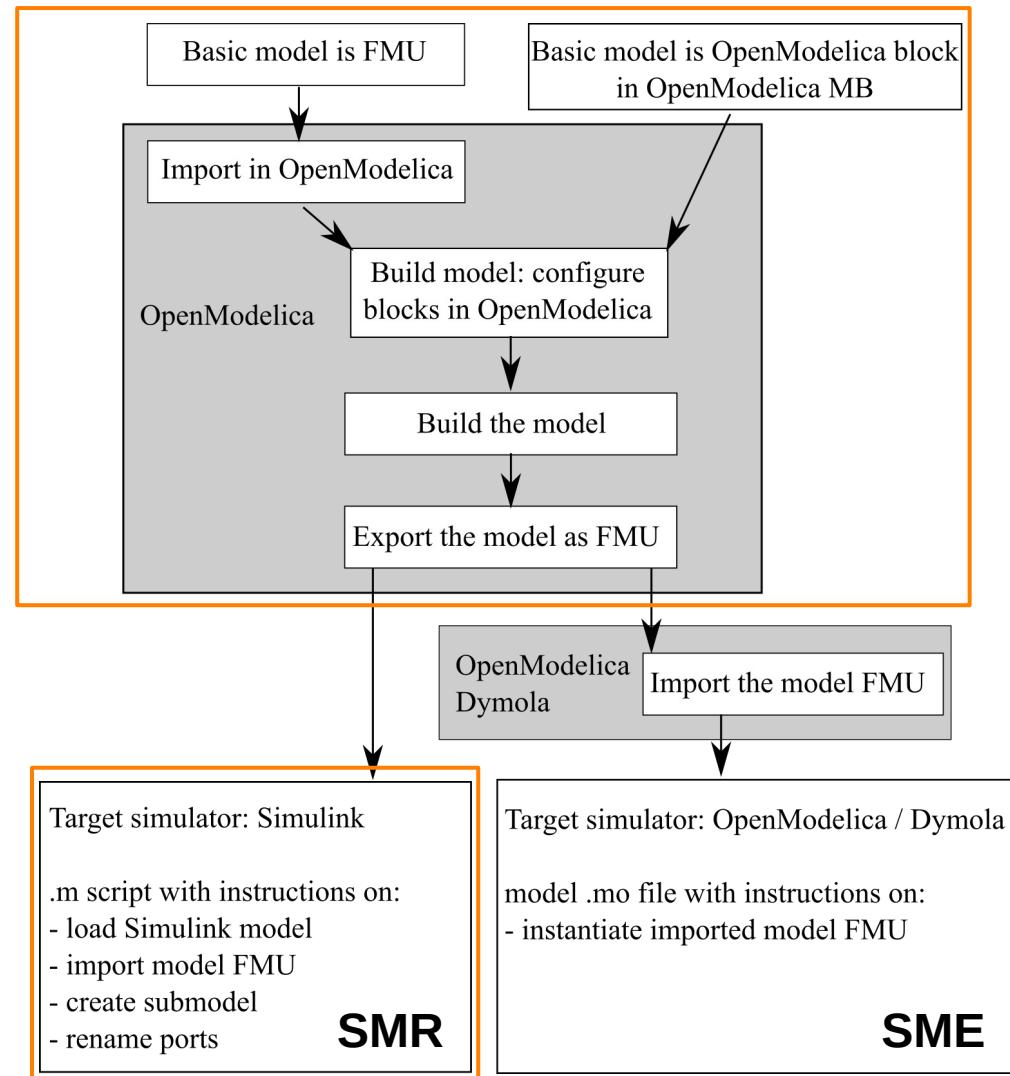
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **"Simulation Model Representation" (SMR)**





Model Building Using FMI (2) (state of the art)

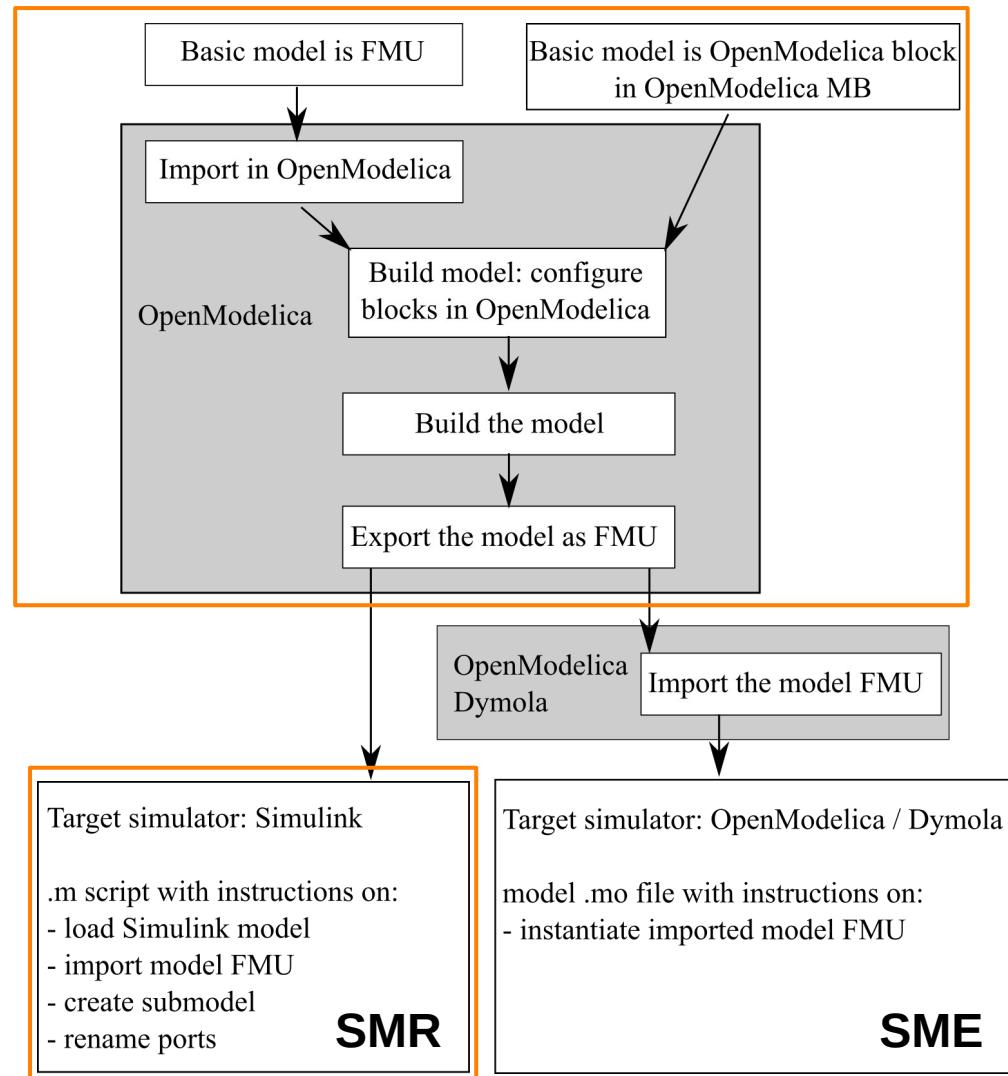
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **"Simulation Model Representation" (SMR)**





Model Building Using FMI (2) (state of the art)

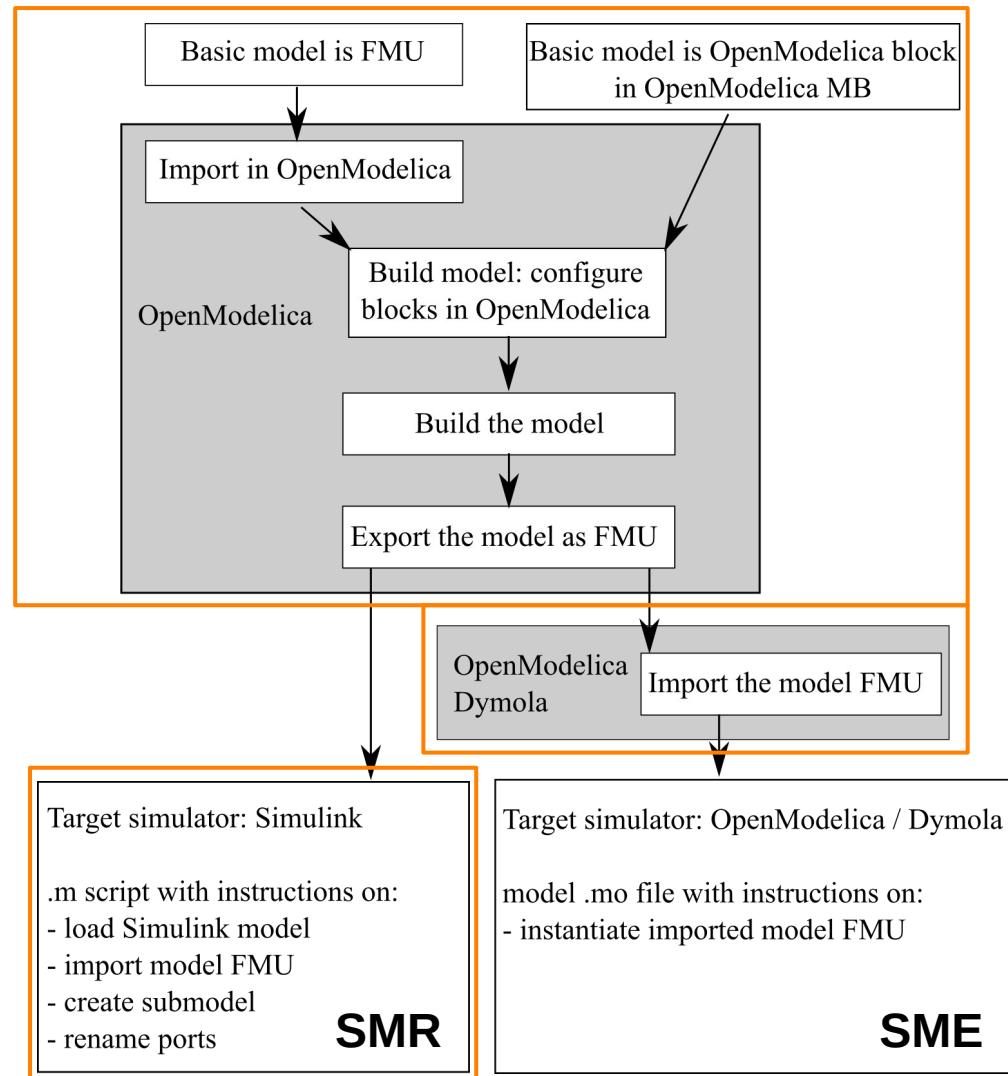
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **“Simulation Model Representation” (SMR)**
- *OpenModelica / Dymola* models are textfiles defining the executable model → **“Simulation Model Executable” (SME)**





Model Building Using FMI (2) (state of the art)

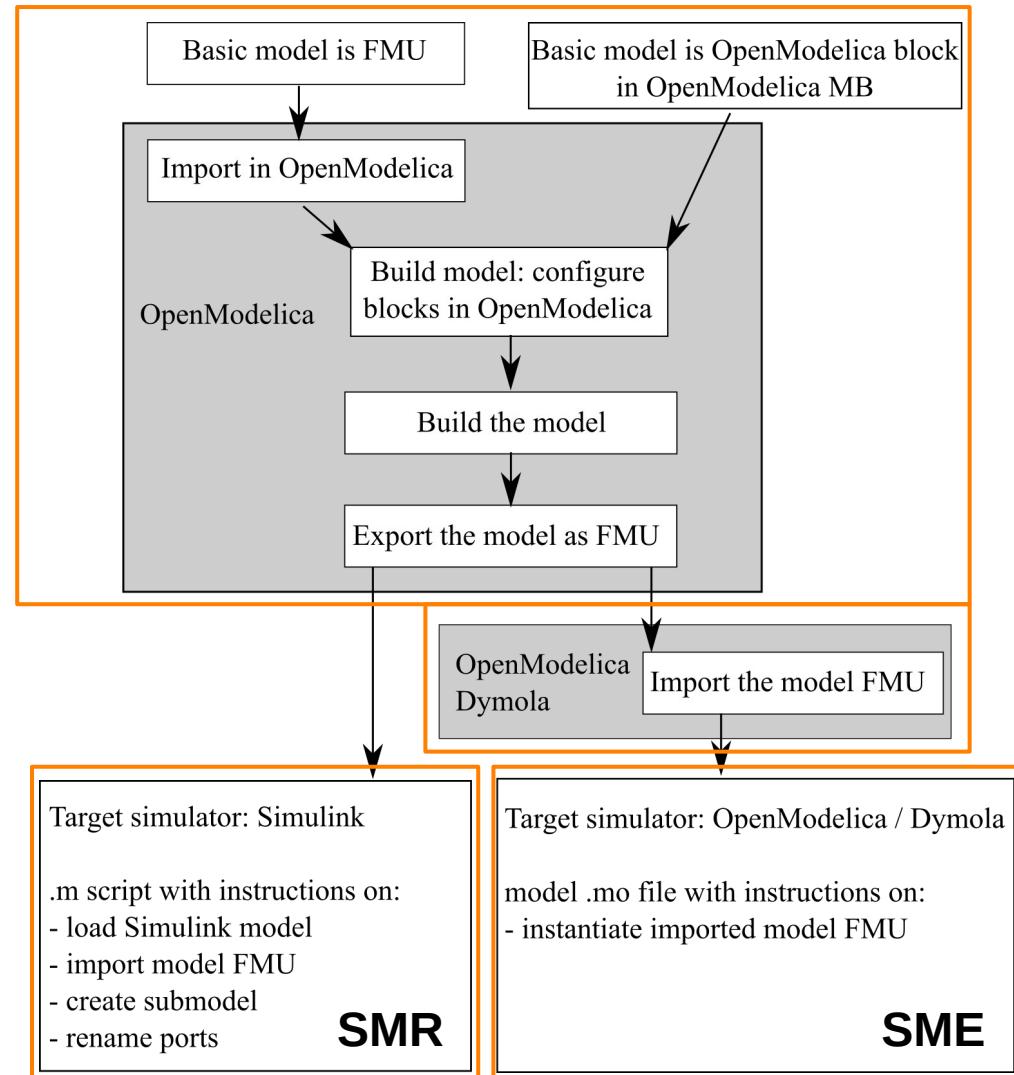
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **“Simulation Model Representation” (SMR)**
- *OpenModelica / Dymola* models are textfiles defining the executable model → **“Simulation Model Executable” (SME)**





Model Building Using FMI (2) (state of the art)

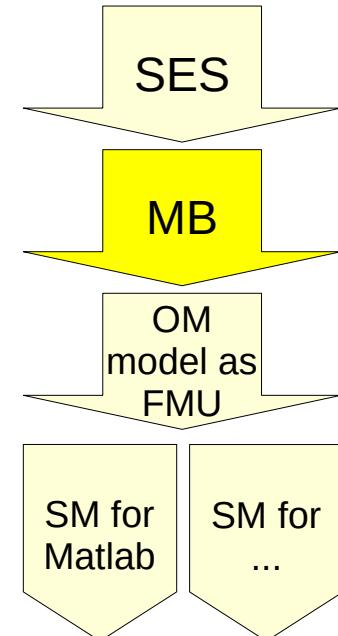
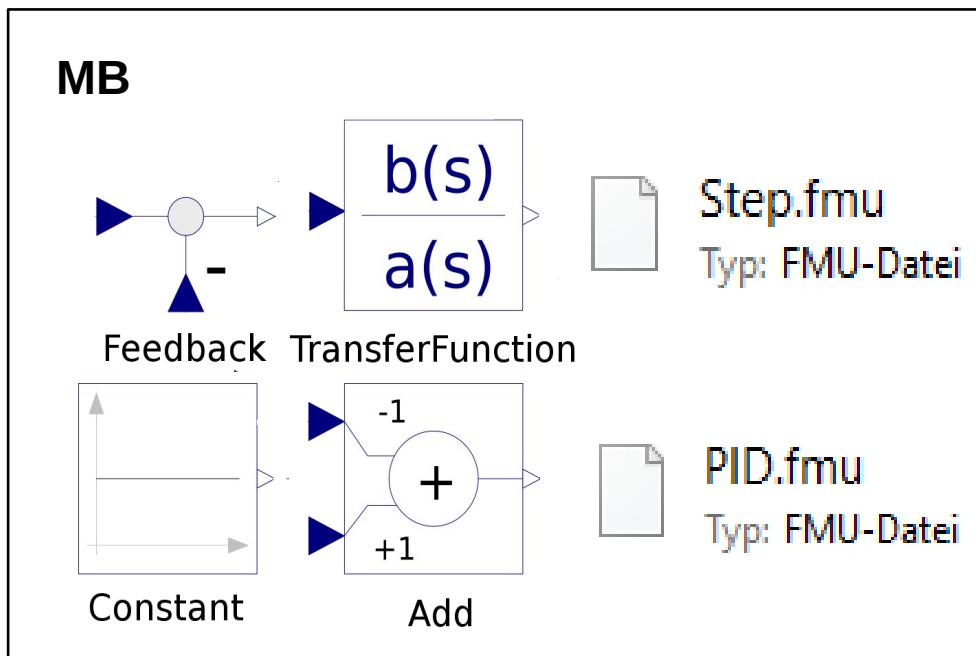
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **“Simulation Model Representation” (SMR)**
- *OpenModelica / Dymola* models are textfiles defining the executable model → **“Simulation Model Executable” (SME)**





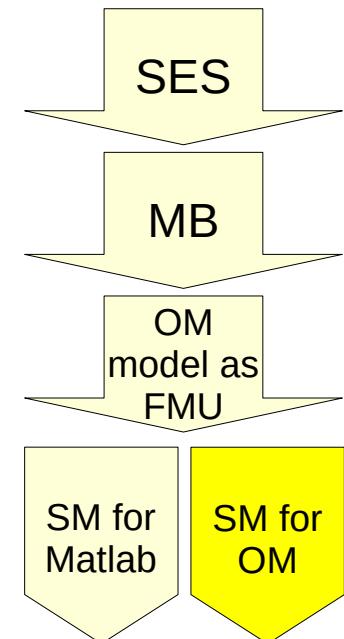
Demonstration of the SES and a Model Base with OpenModelica and FMI Components

- SES nearly like before, but interface is ‘FMI’ and mb-attributes differ
→ prune, flatten
- Model generation takes a while → presented on the next slides





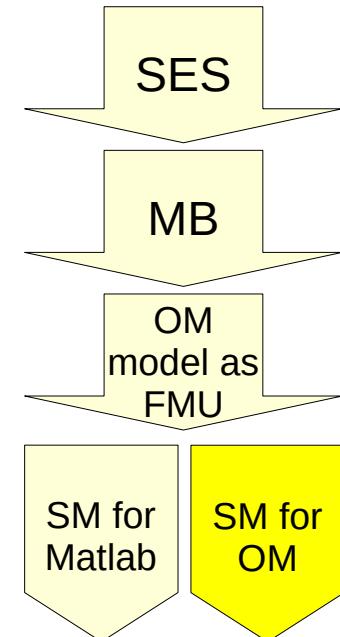
Case Study: FMU Imported in OpenModelica





Case Study: FMU Imported in OpenModelica

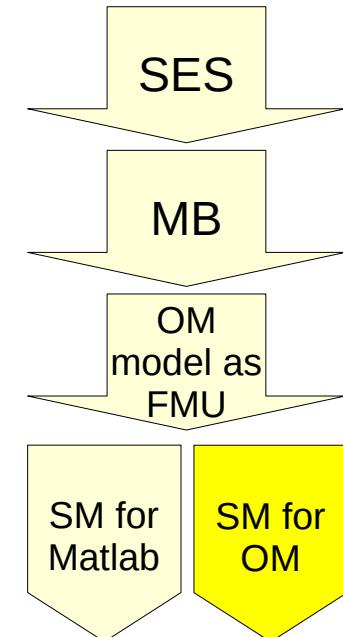
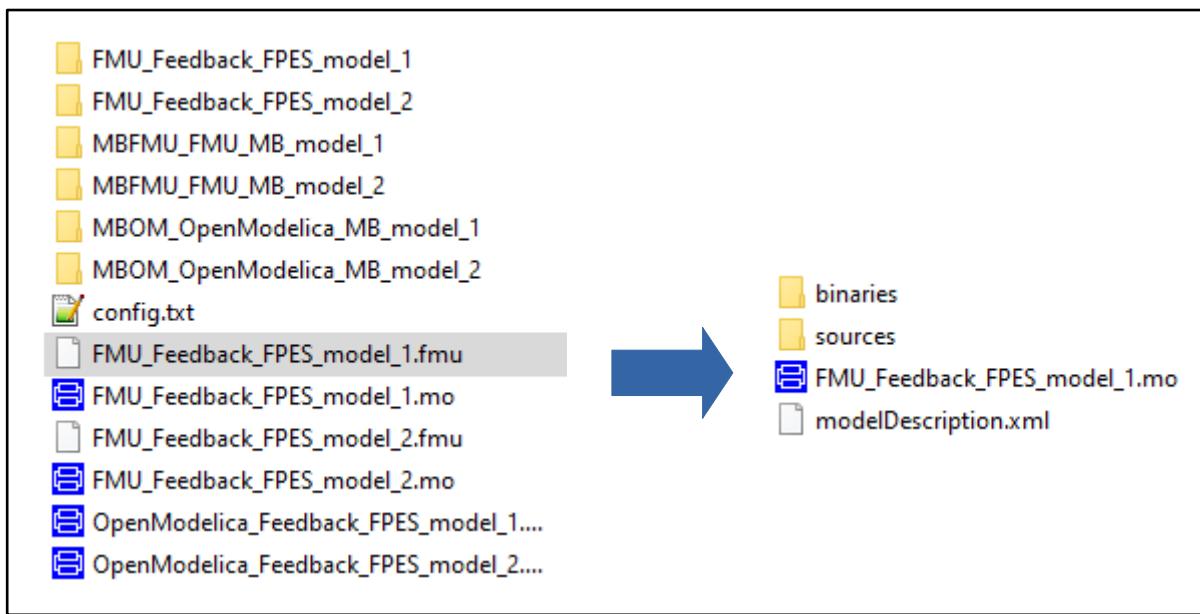
- Import the model FMU in the simulator
→ model (.mo file) is created





Case Study: FMU Imported in OpenModelica

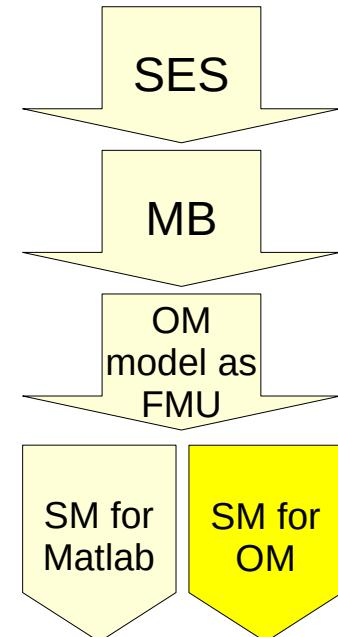
- Import the model FMU in the simulator
→ model (.mo file) is created





Case Study: FMU Imported in OpenModelica

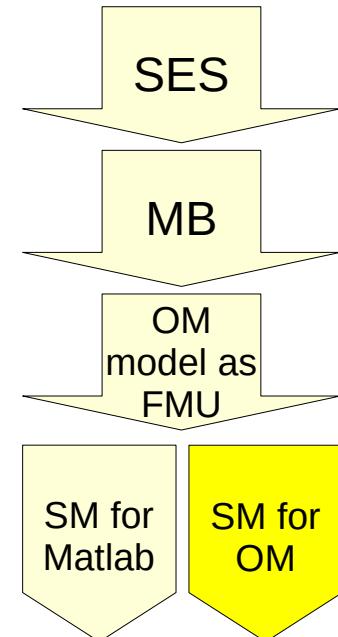
- Import the model FMU in the simulator
→ model (.mo file) is created





Case Study: FMU Imported in OpenModelica

- Import the model FMU in the simulator
→ model (.mo file) is created
- Create an OpenModelica model:
 - Instantiate imported FMU





Case Study: FMU Imported in OpenModelica

- Import the model FMU in the simulator
→ model (.mo file) is created
- Create an OpenModelica model:
 - Instantiate imported FMU

```
model OpenModelica_Feedback_FPES_model_1
    FMU_Feedback_FPES_model_1 FMU_Feedback_FPES_model_11();
equation
end OpenModelica_Feedback_FPES_model_1;
```

