Model Behaviour Generation for Multiple Simulators

Faculty of Engineering / Research Group CEA

Thorsten Pawletta
Hendrik Folkerts

E-Mail:
thorsten.pawletta@hs-wismar.de
hendrik.folkerts@hs-wismar.de

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

HOCHSCHULE WISMAR
UNIVERSITY OF APPLIED SCIENCES
TECHNOLOGY, BUSINESS AND DESIGN

Prerequisite

A prerequisite for this supplementary material is the knowledge of Chapter 1.5, where basic concepts of the System Entity Structure (SES) are described.

Outline

1. Case study
2. Implementation of the SES and an MB
3. Model selection and model generation
4. Organization of a simulator-independent MB
5. Full automation of simulation experiments
6. Summary

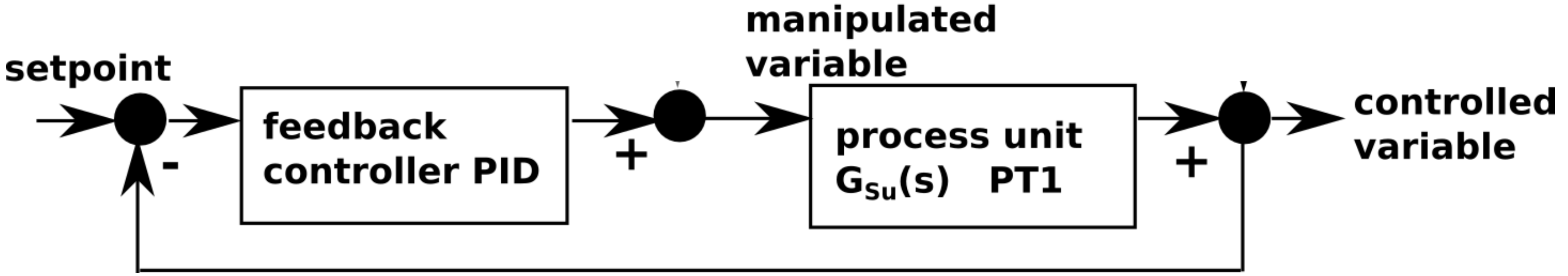
Outline

1. **Case study**
2. Implementation of the SES and an MB
3. Model selection and model generation
4. Organization of a simulator-independent MB
5. Full automation of simulation experiments
6. Summary

Case Study

Case Study

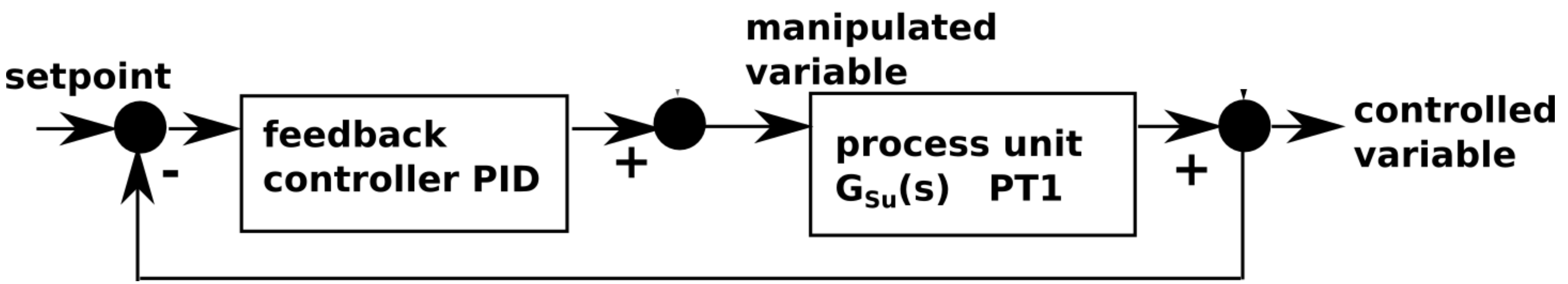
- Feedback control system



Case Study

- Feedback control system
- Described by transfer functions

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

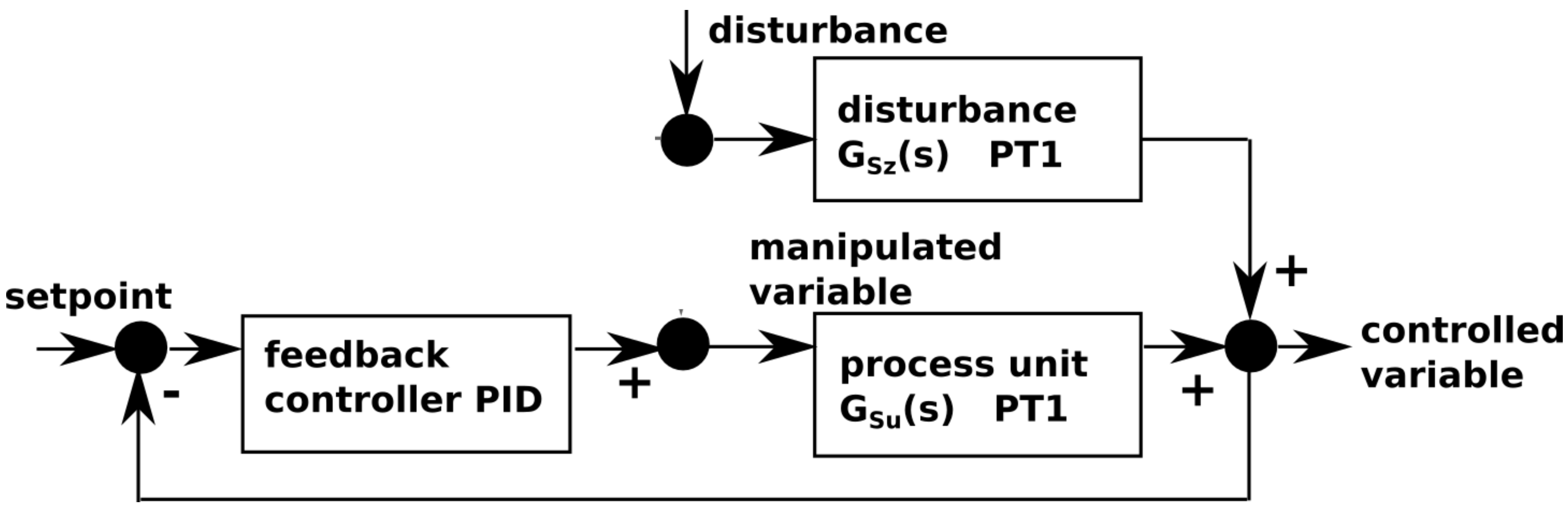


Case Study

- 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

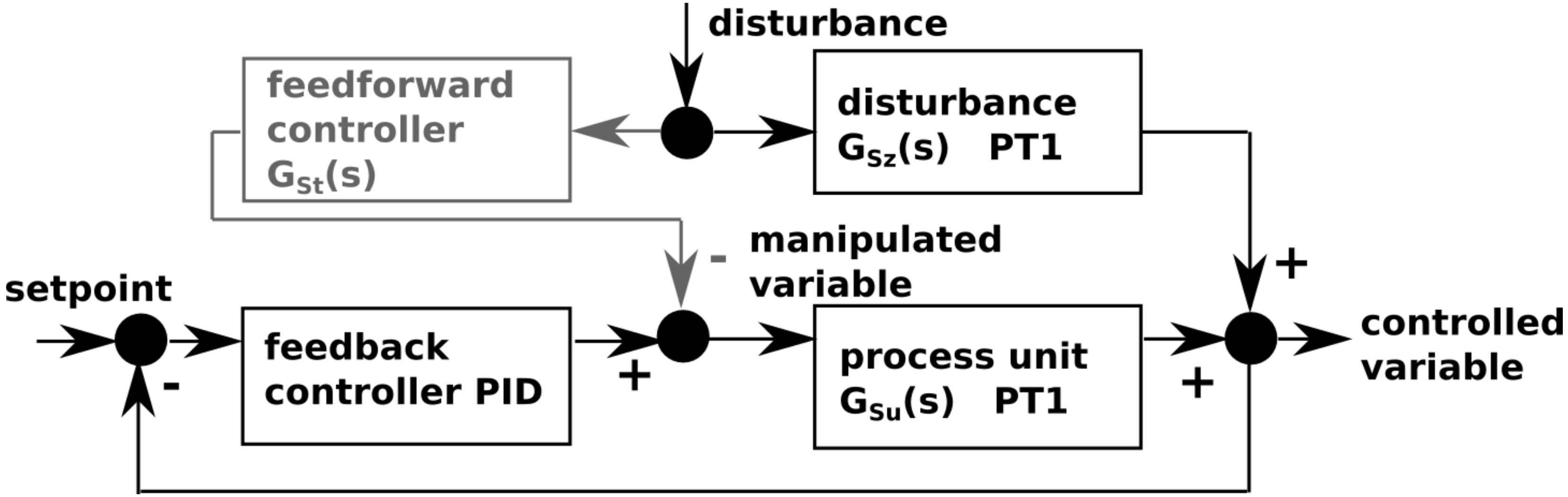
- Feedback control system
- Described by transfer functions
- Influenced by disturbances
- Measurable disturbances

  → Compensated with feedforward control

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

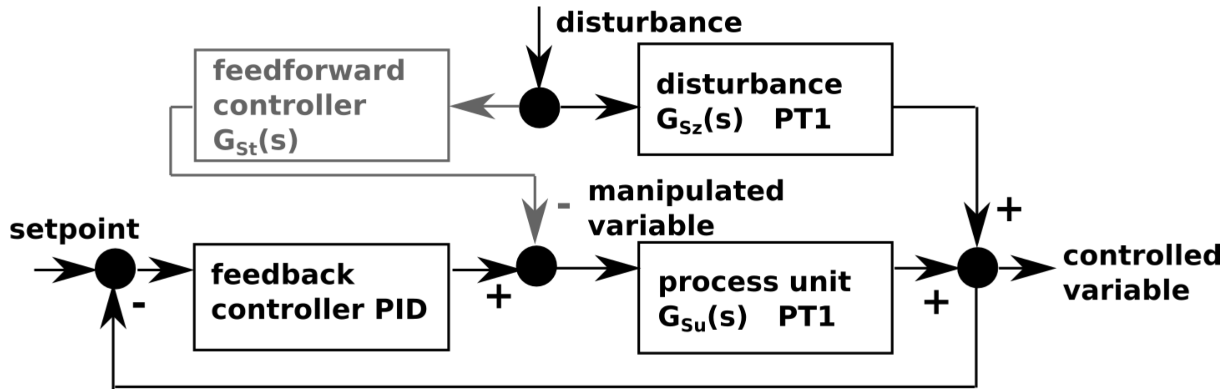
$$G_{Sz}(s) = \frac{1}{10 \cdot s + 1}$$

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



Case Study (2)

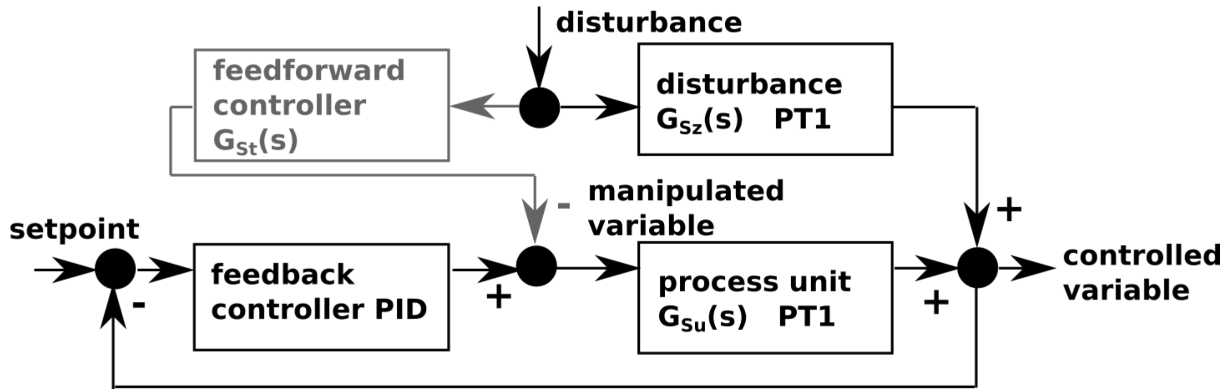
- **Two system structure variants**
  - Without feedforward control: `feedforward=0`
  - With feedforward control: `feedforward=1`



Case Study (2)

- **Two system structure variants**
  - ➢ Without feedforward control: `feedforward=0`
  - ➢ With feedforward control: `feedforward=1`
- For every structure variant

→ **Different parameter configurations of PID controller** (we consider two)



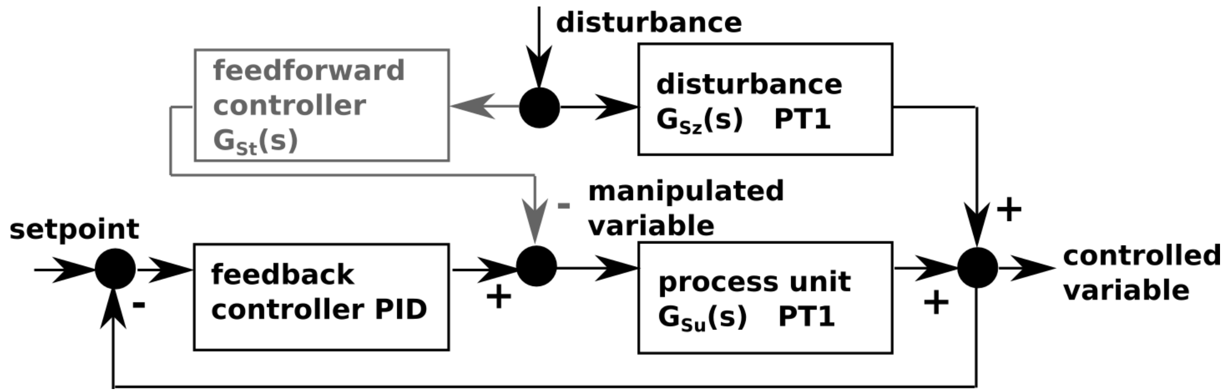
Case Study (2)

Design objective:
Find best control configuration.

- **Two system structure variants**
  - Without feedforward control: `feedforward=0`
  - With feedforward control: `feedforward=1`
- For every structure variant

  → **Different parameter configurations of PID controller**

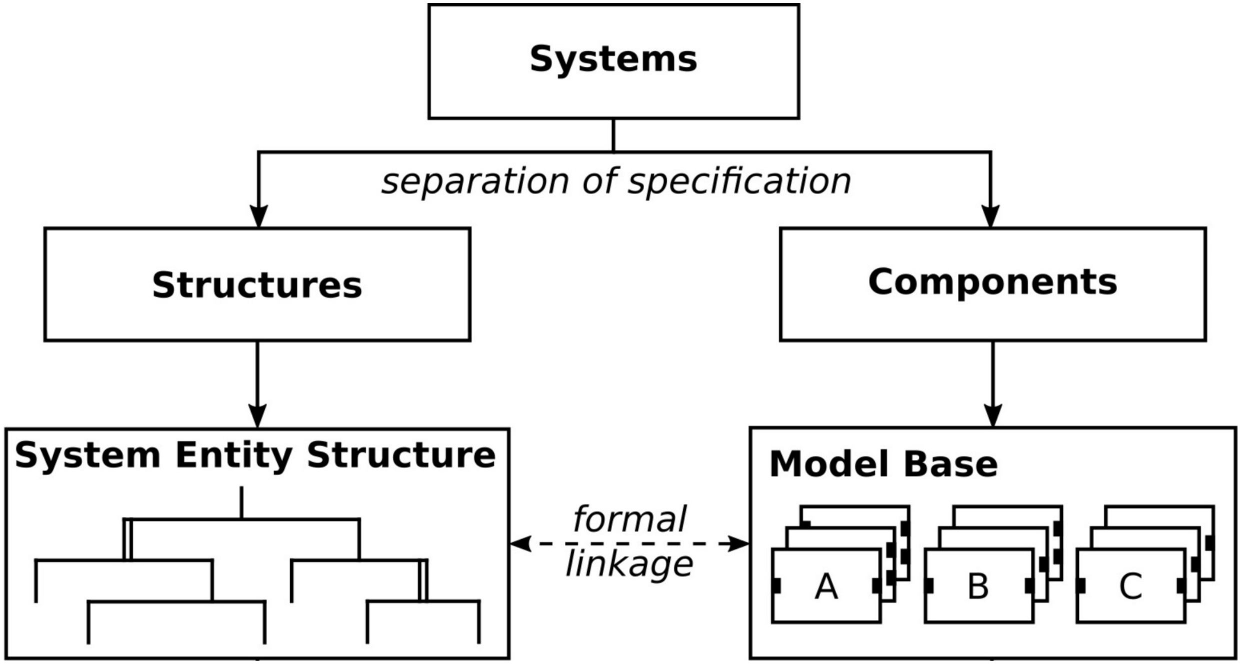
  (we consider two)



Outline

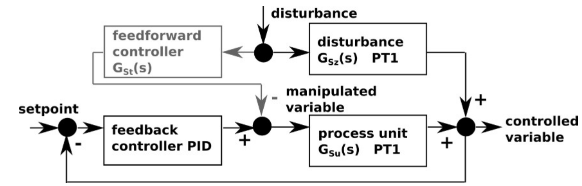
1. Case study
2. **Implementation of the SES and an MB**
3. Model selection and model generation
4. Organization of a simulator-independent MB
5. Full automation of simulation experiments
6. Summary

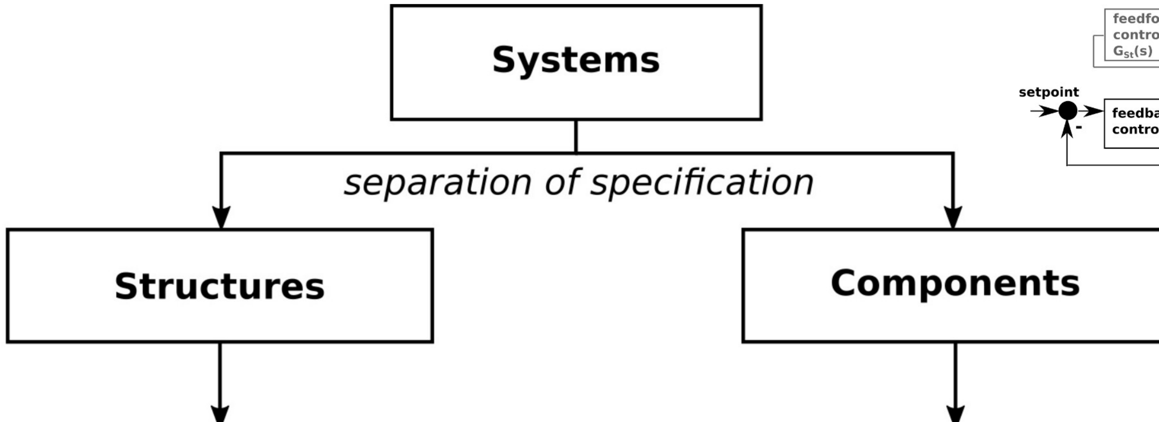
SES/MB-based Modeling

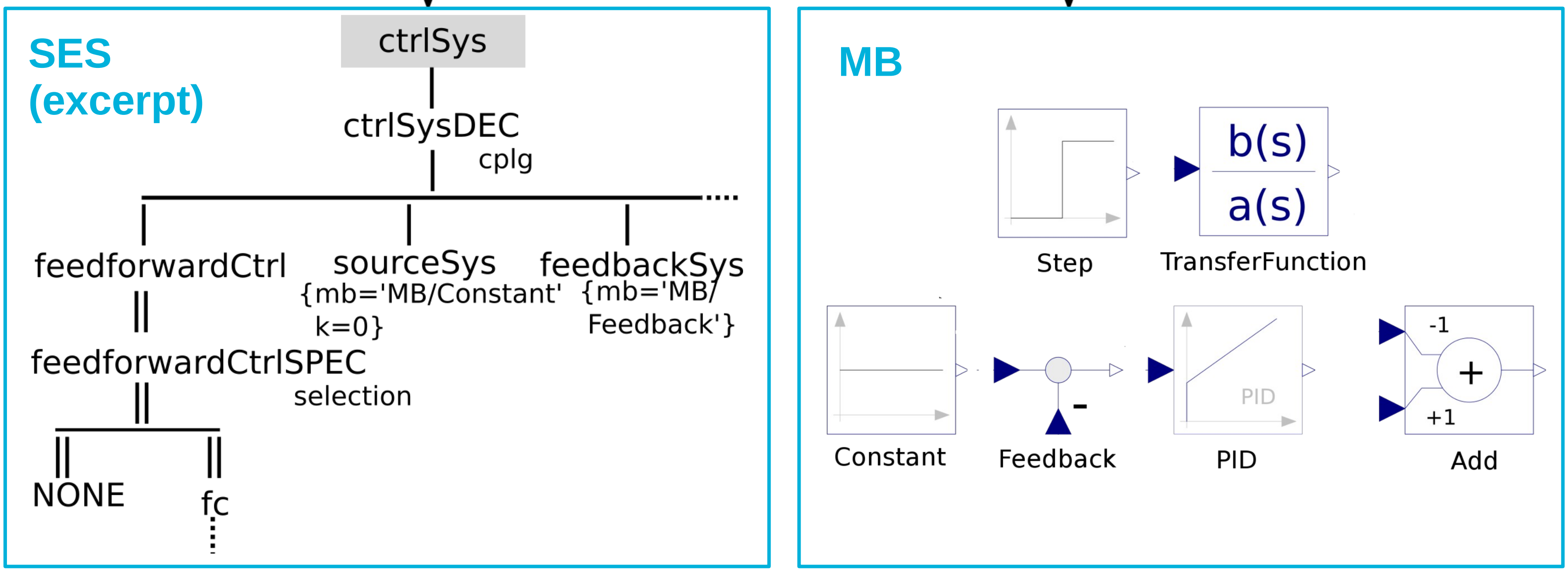


- **SES** describes permissible structure & parameter variants (simulator-independent)
- **MB** defines basic dynamic models (usually simulator dependent)

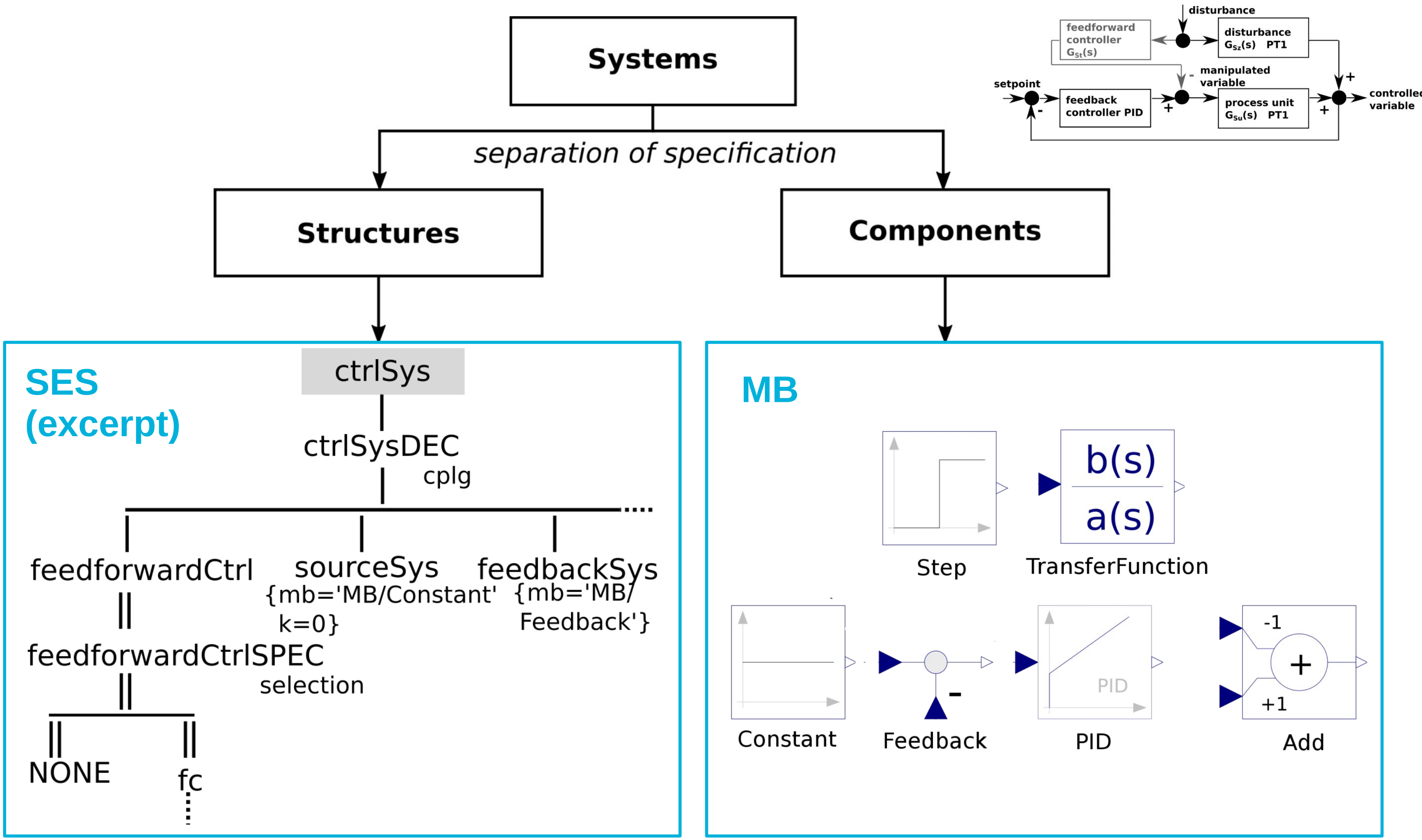
SES/MB-based Modeling of the Case Study



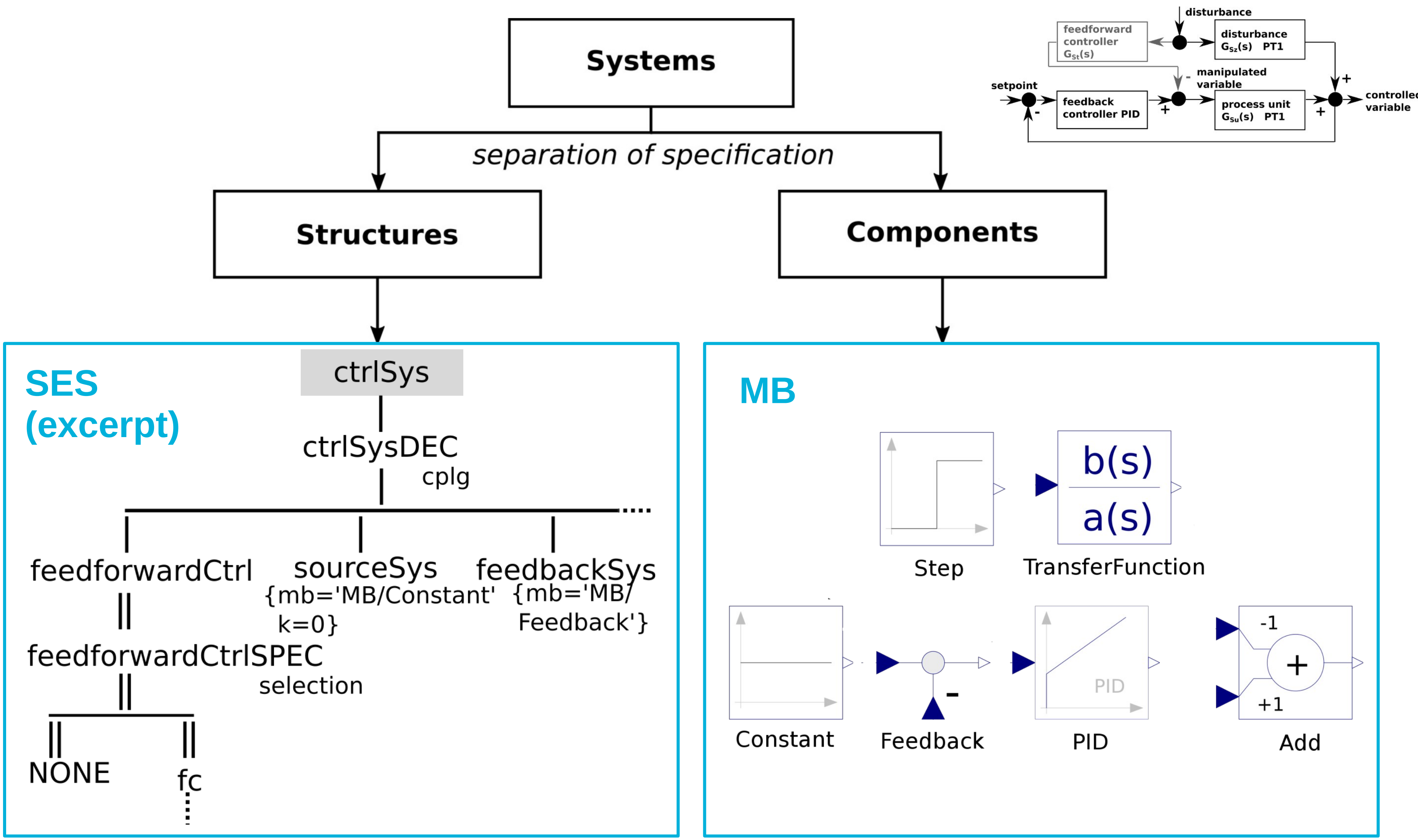




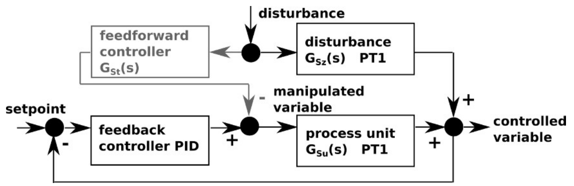
SES/MB-based Modeling of the Case Study

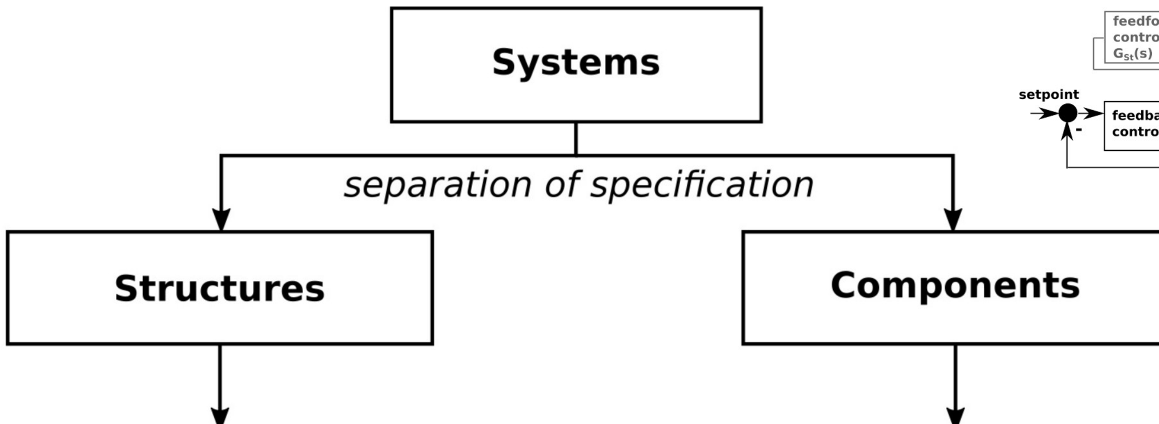


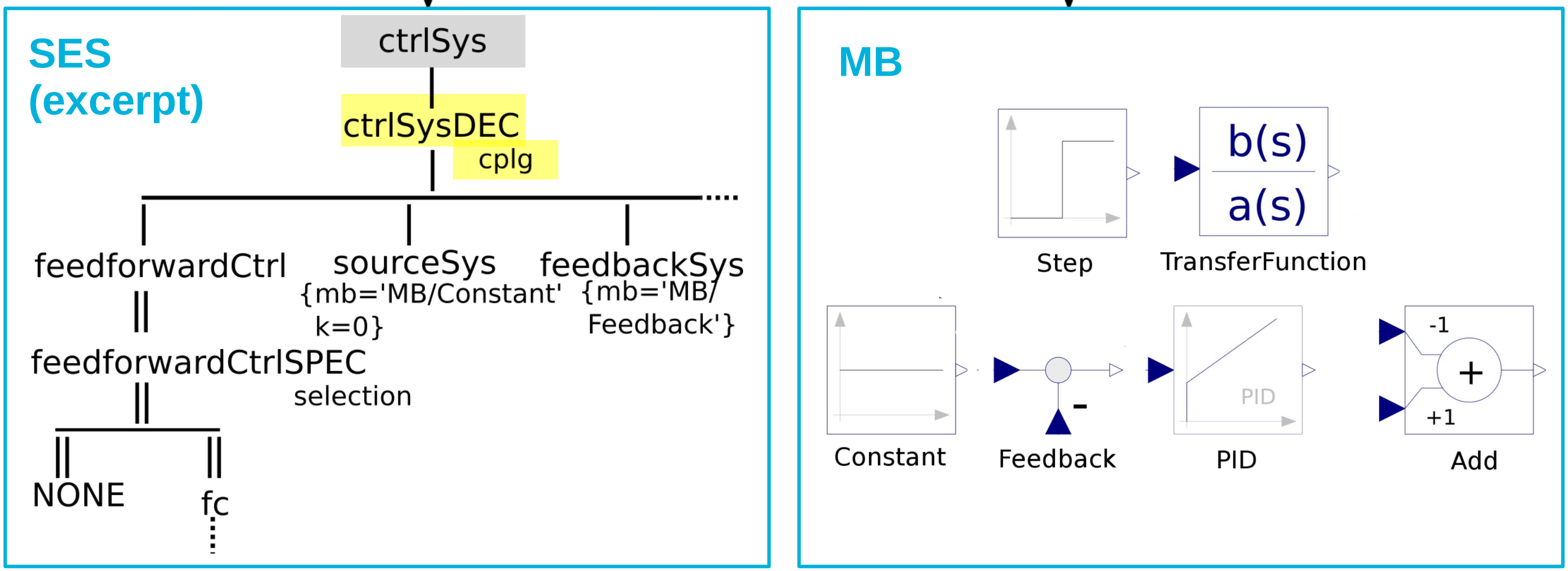
SES/MB-based Modeling of the Case Study



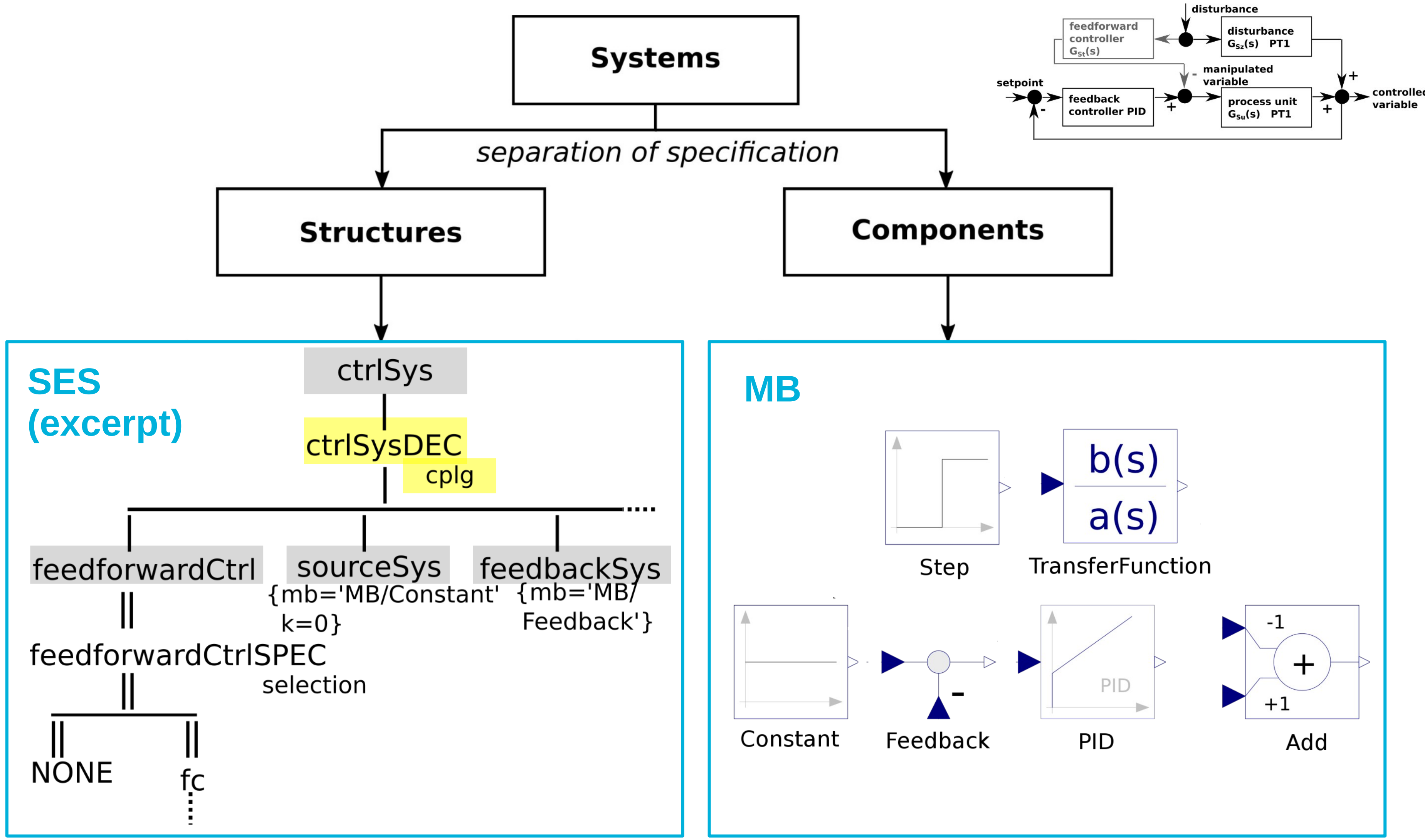
SES/MB-based Modeling of the Case Study



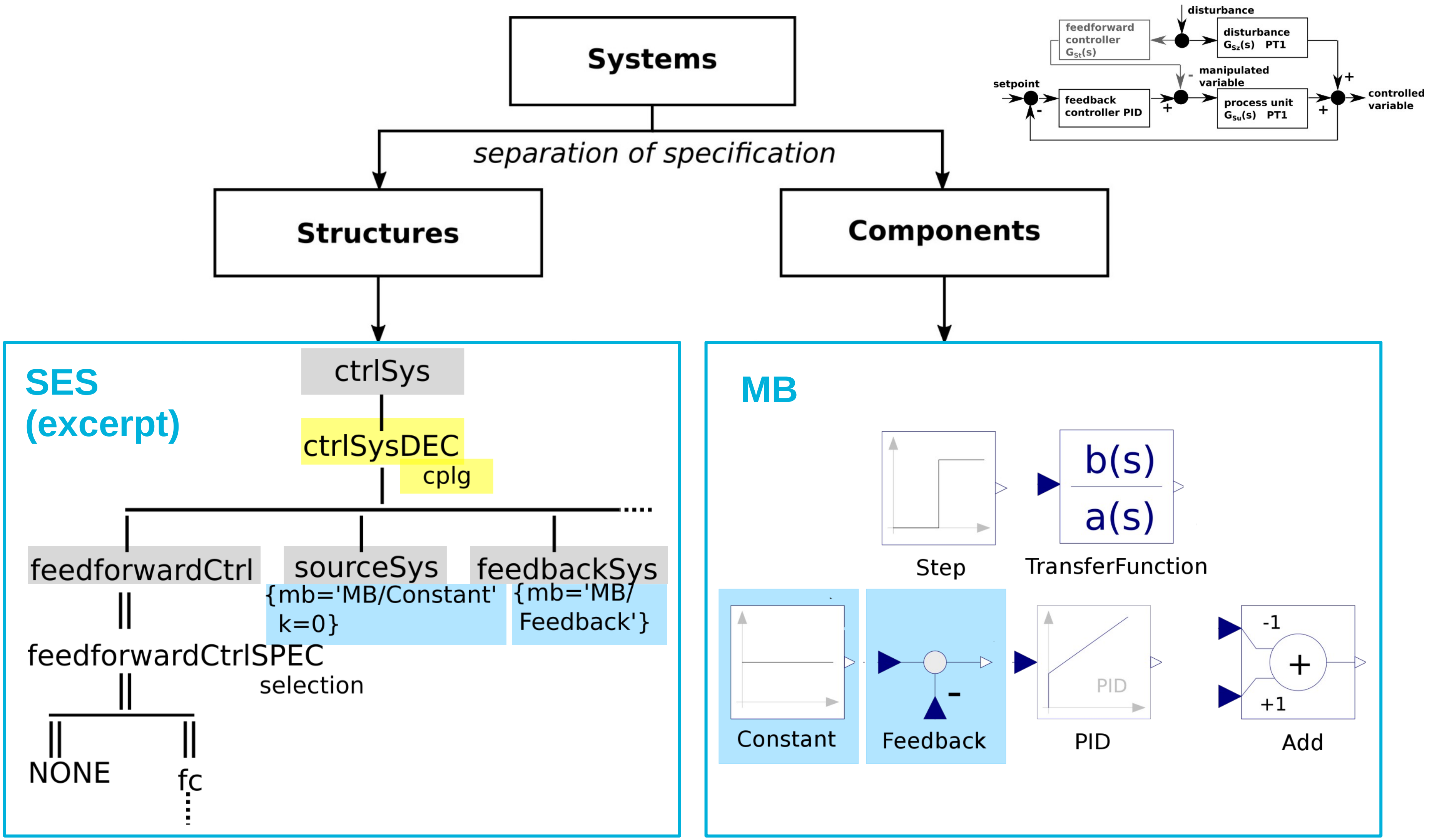




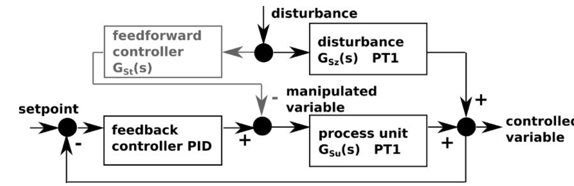
SES/MB-based Modeling of the Case Study

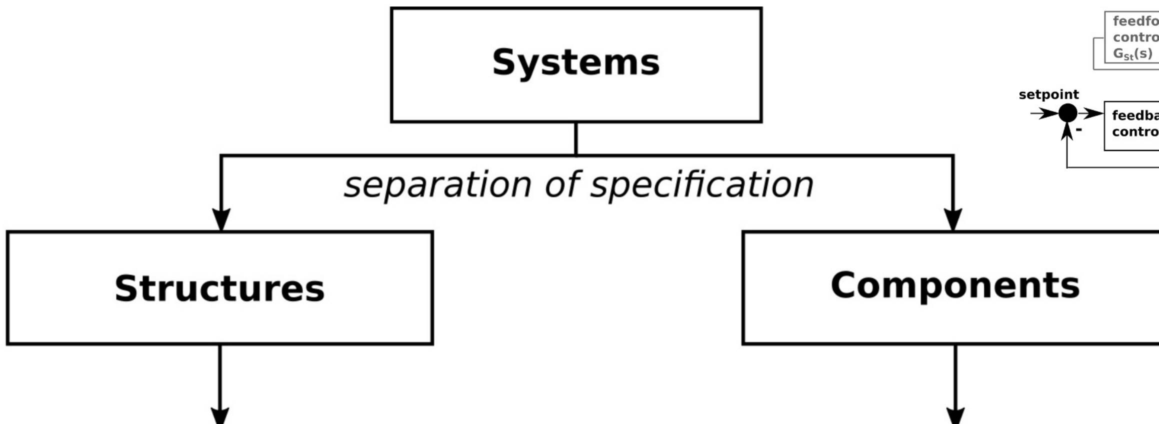


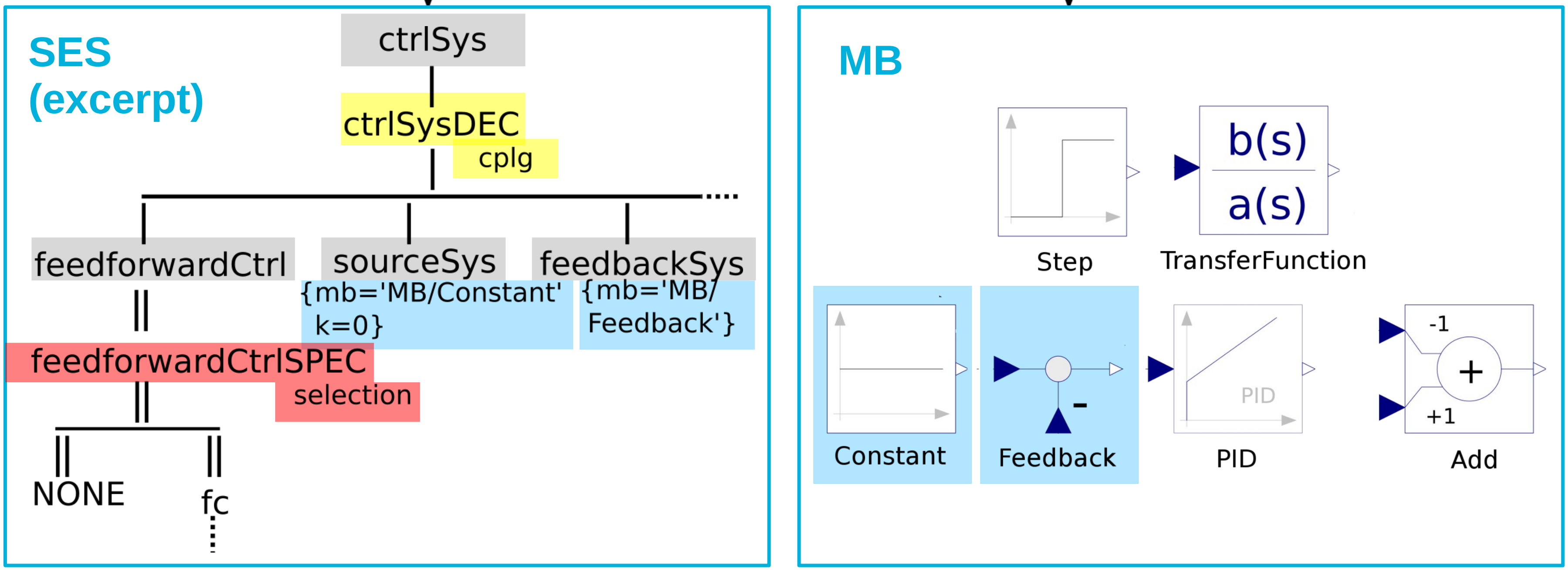
SES/MB-based Modeling of the Case Study



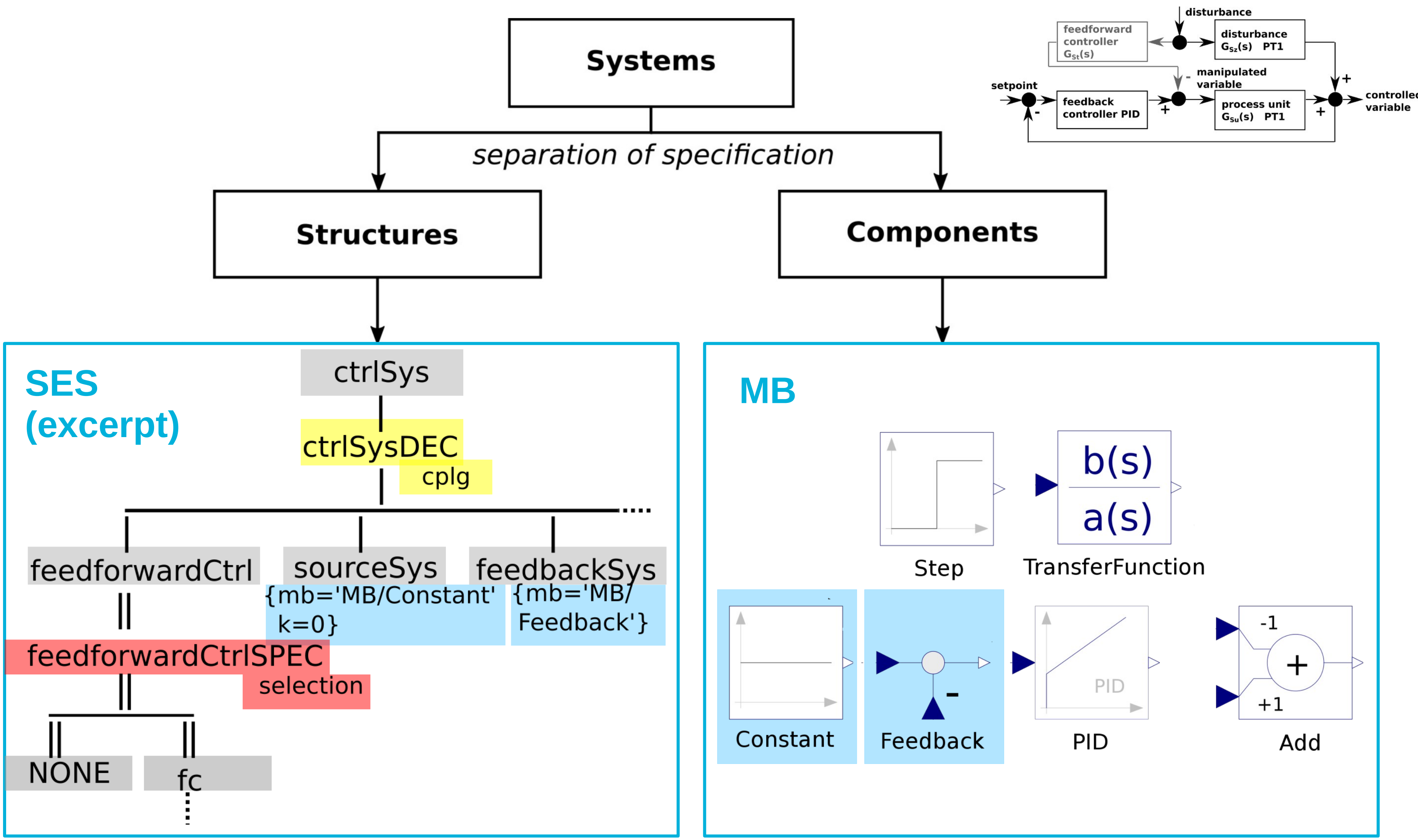
SES/MB-based Modeling of the Case Study



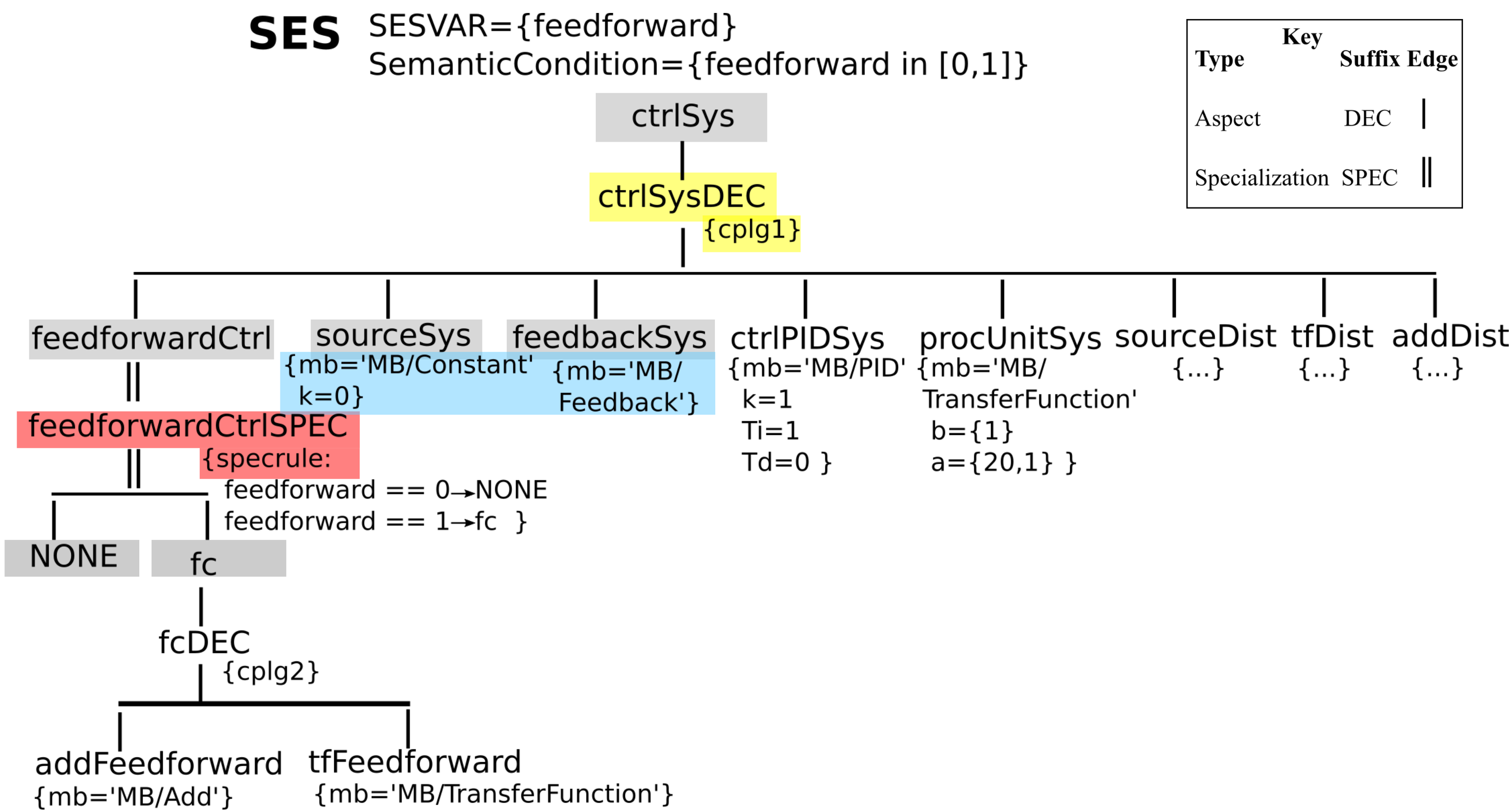




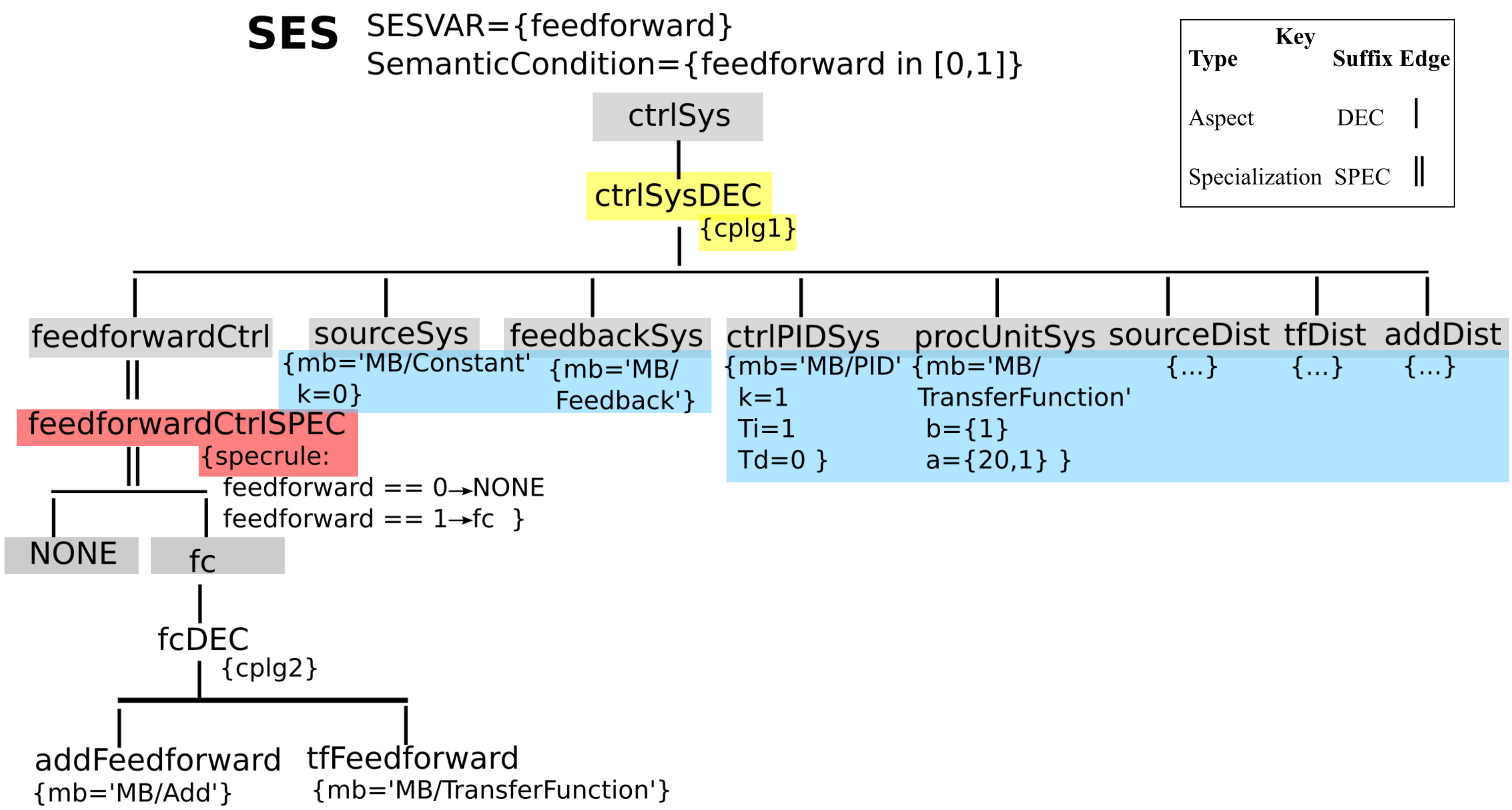
SES/MB-based Modeling of the Case Study



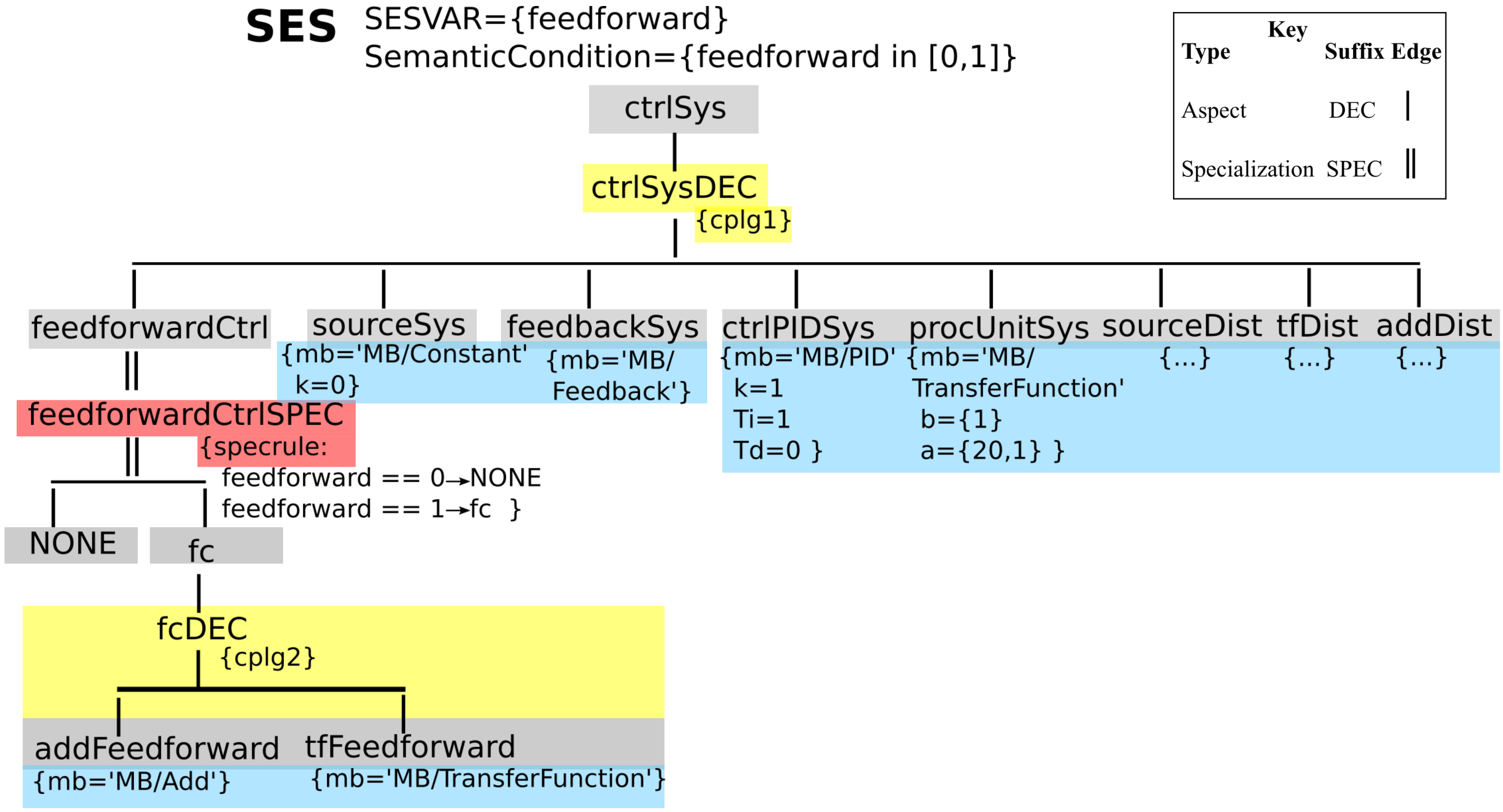
More Detailed Extract of the SES



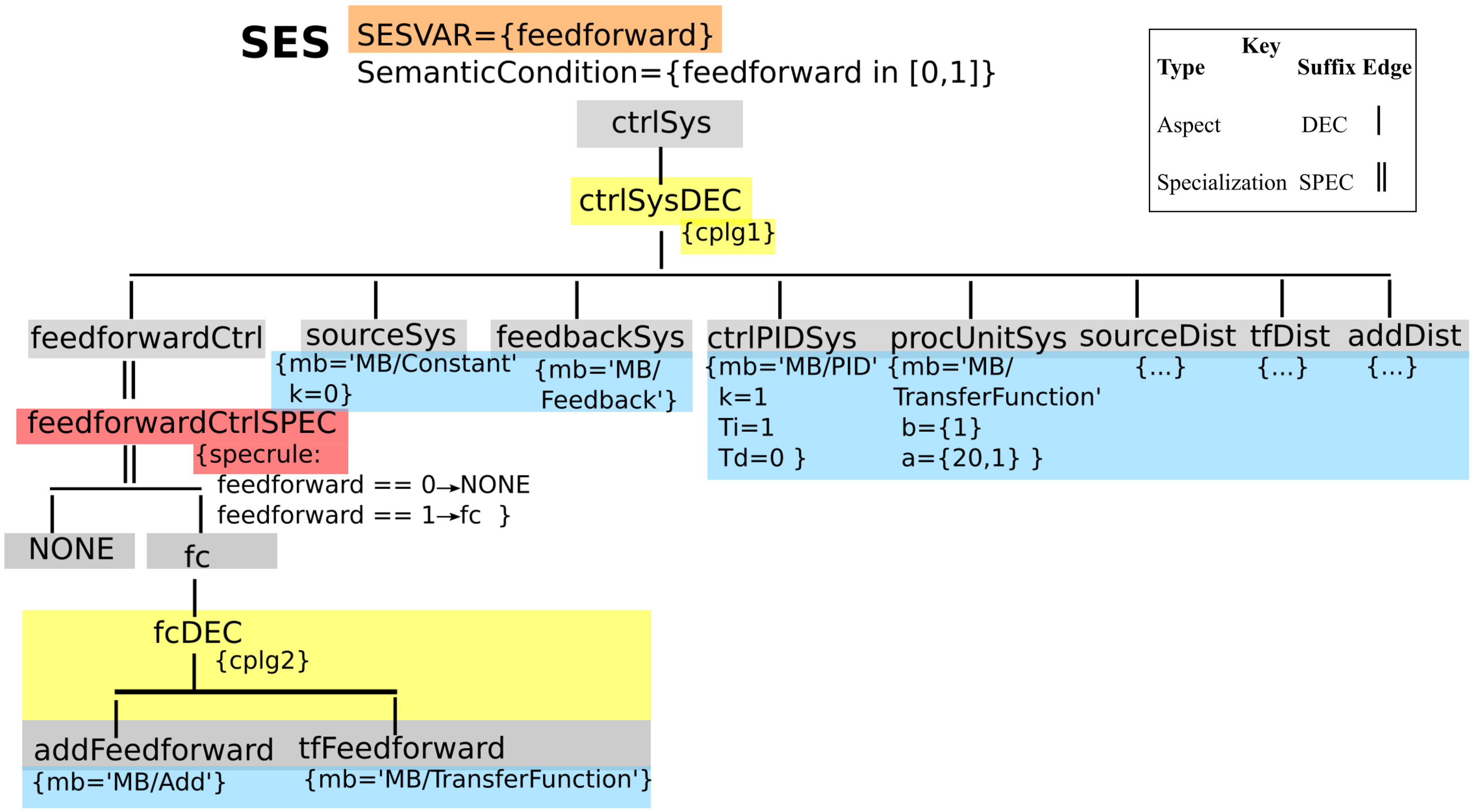
More Detailed Extract of the SES



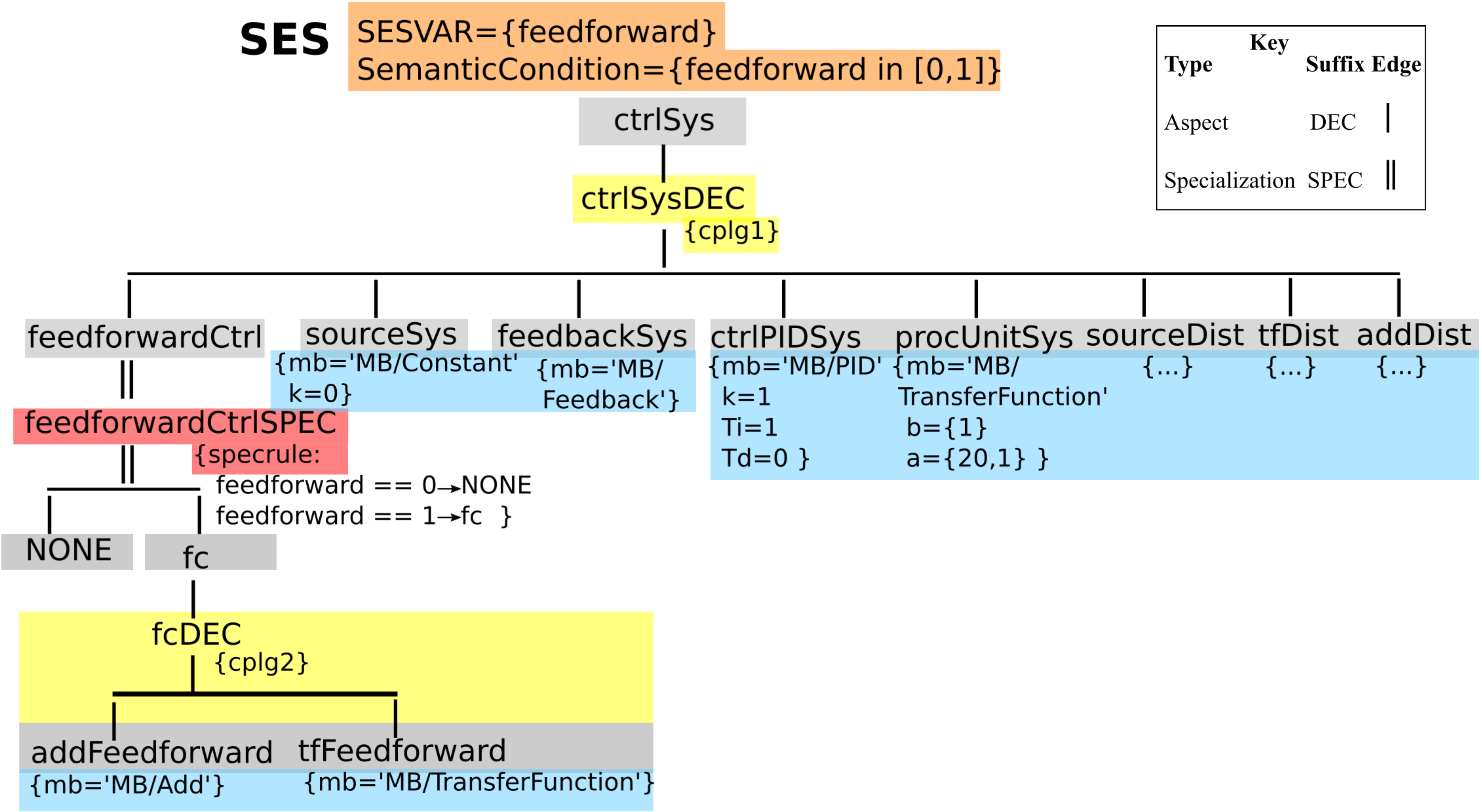
More Detailed Extract of the SES



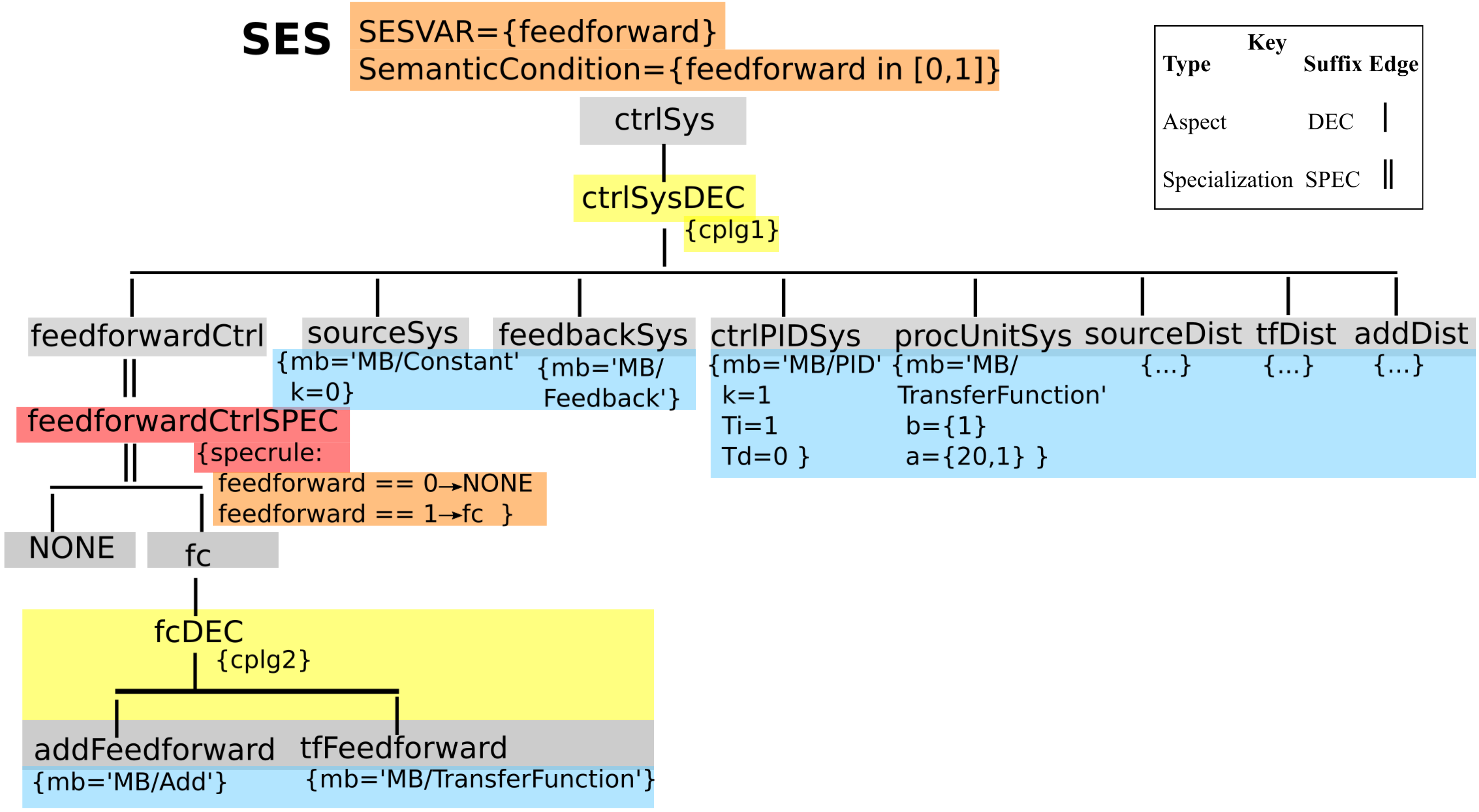
More Detailed Extract of the SES



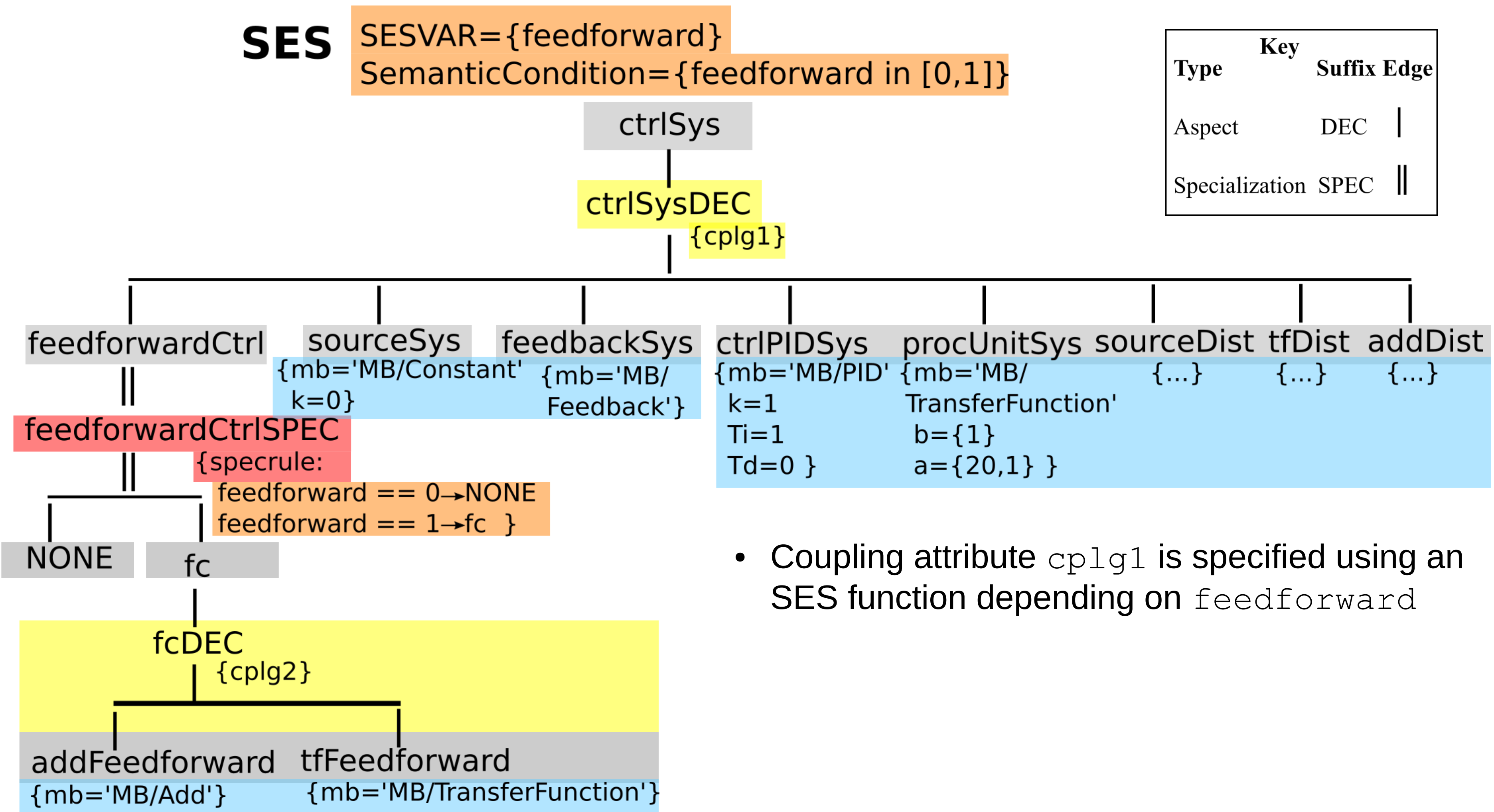
More Detailed Extract of the SES



More Detailed Extract of the SES

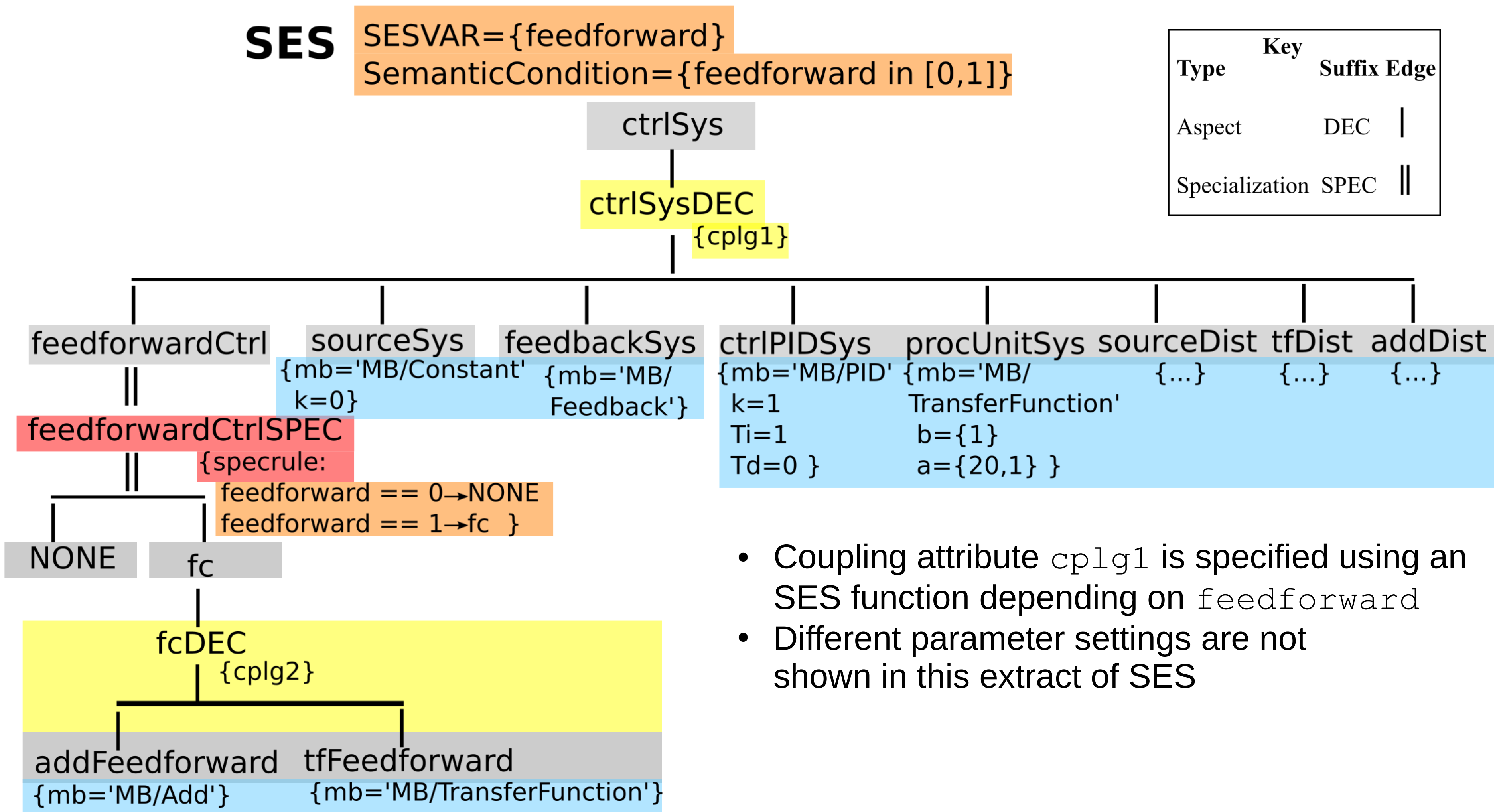


More Detailed Extract of the SES



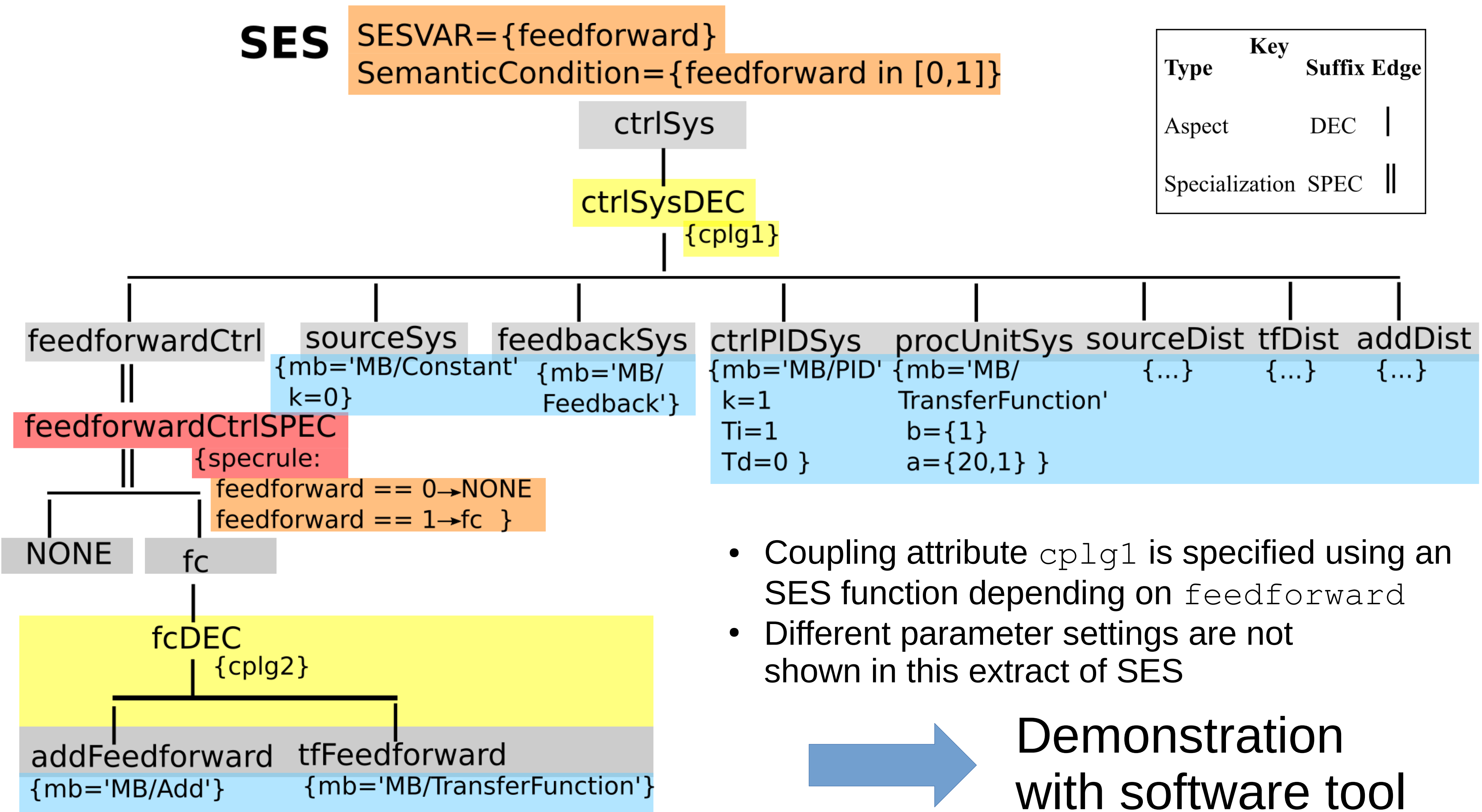
- Coupling attribute `cplg1` is specified using an SES function depending on `feedforward`

More Detailed Extract of the SES



- 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



- 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

Python Toolset

- Available:
  `https://github.com/cea-wismar/SESMB_Inf_Python/`

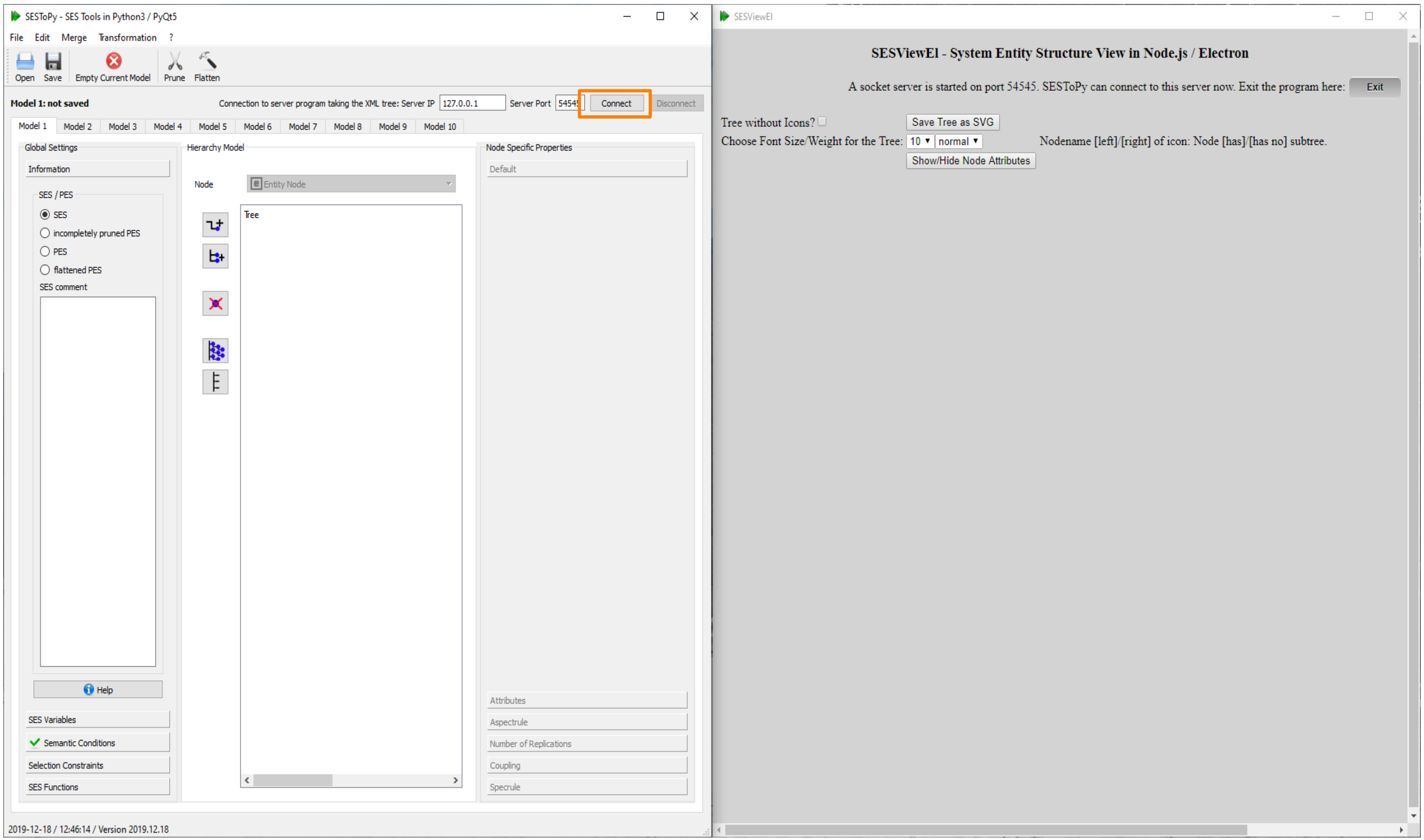
- Tools
  - **SESToPy** → SES editor and IDE
  - **SESViewEl** → SES tree viewer

  - SESMoPy
  - SESEuPy
  - SESEcPy

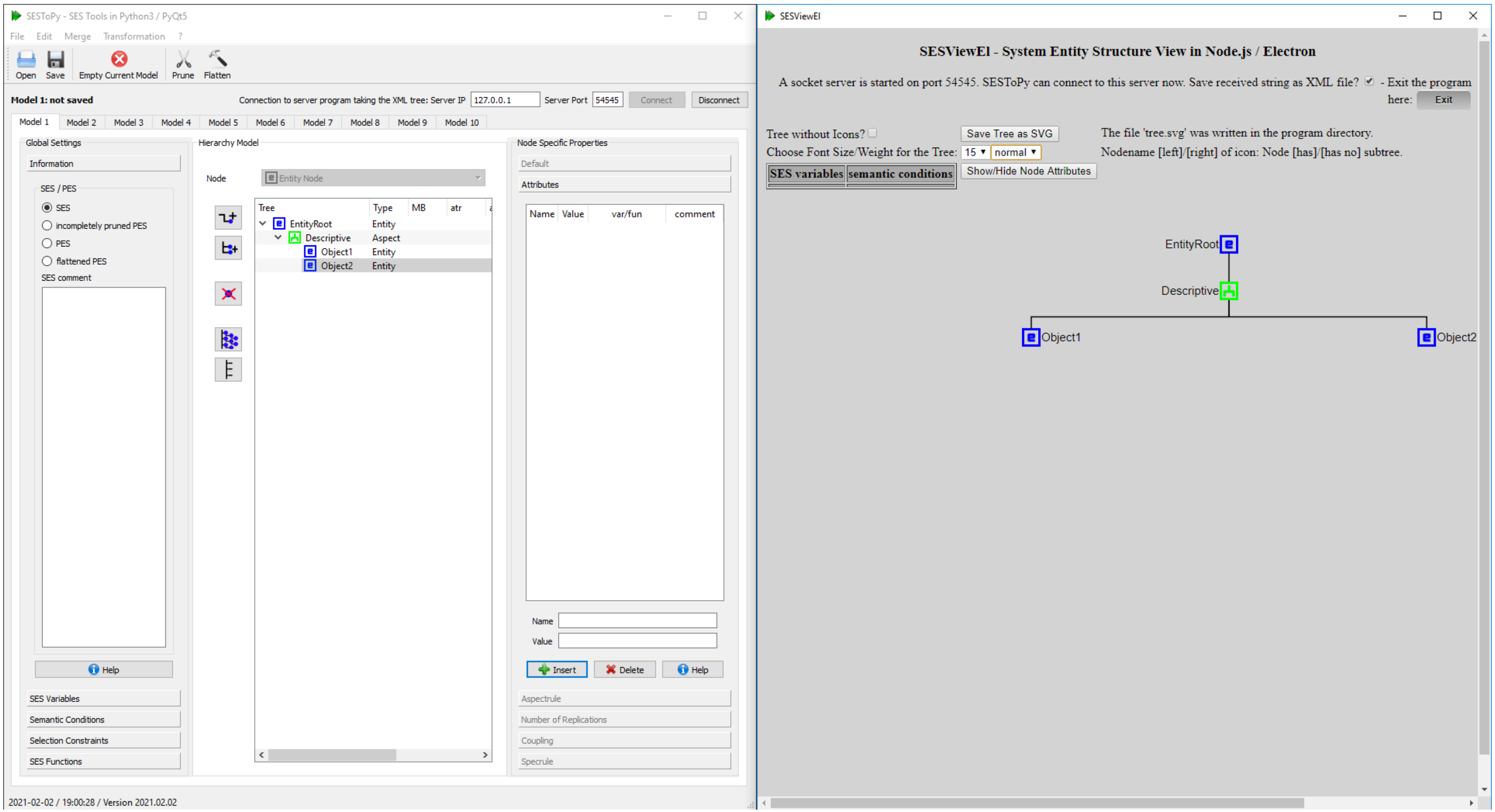
Demonstration of SESToPy with SESViewEl (case study) → Screenshots on Next Slides

- Connect SESToPy with SESViewEl (show SESToPy and SESViewEl 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
- Merging

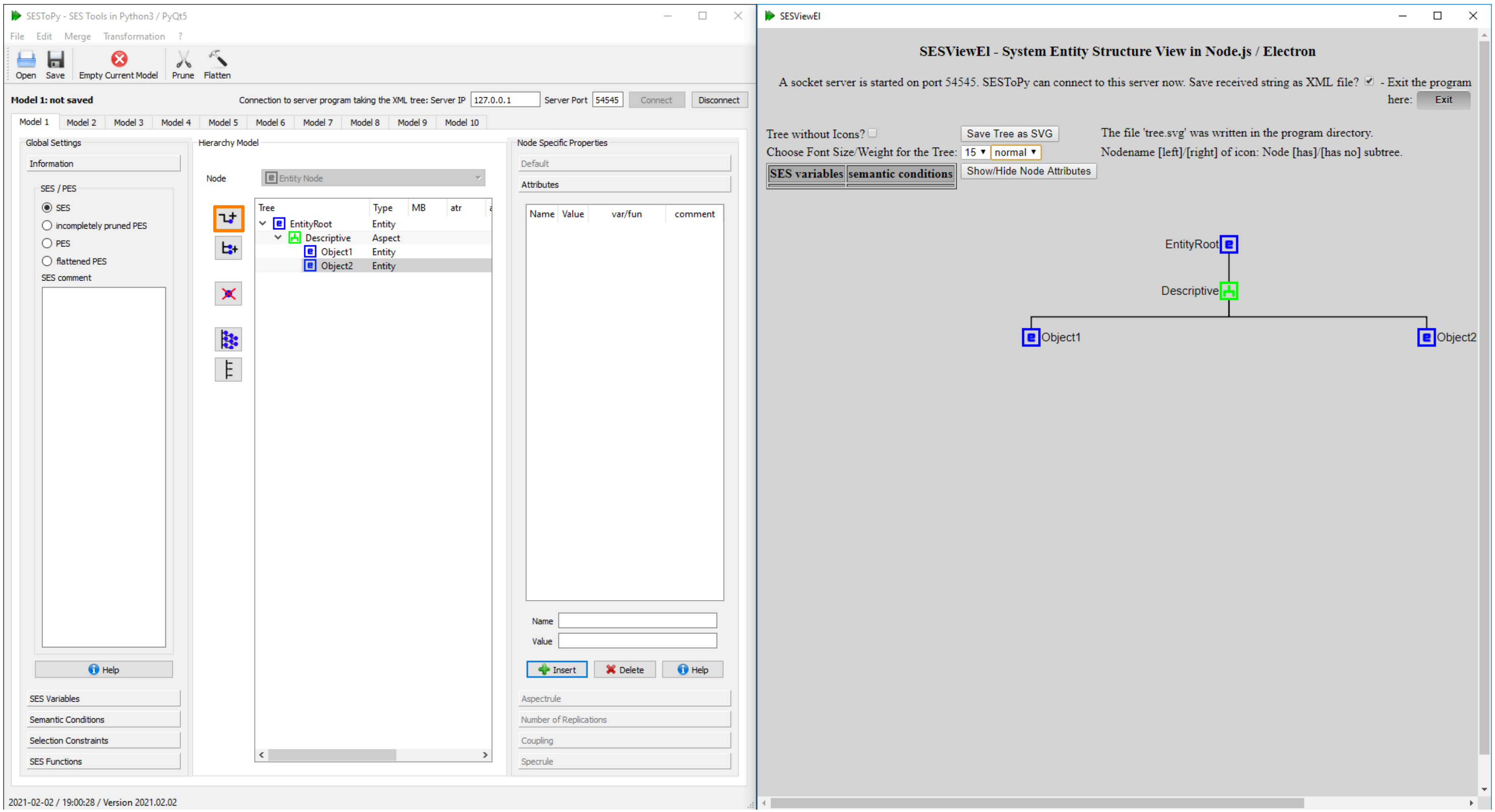
Connection of SESToPy and SESViewEl



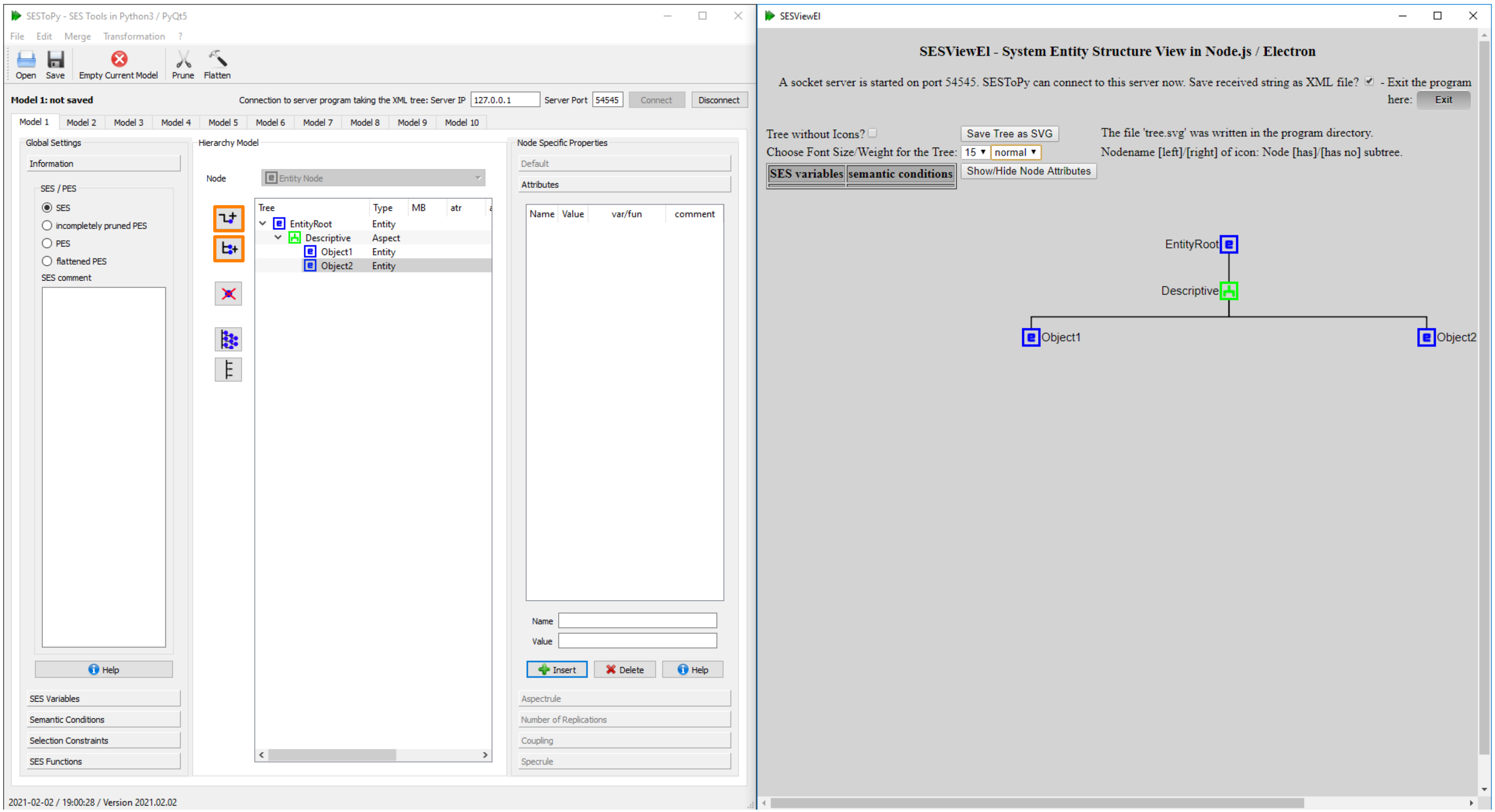
Create SES Tree with SESToPy



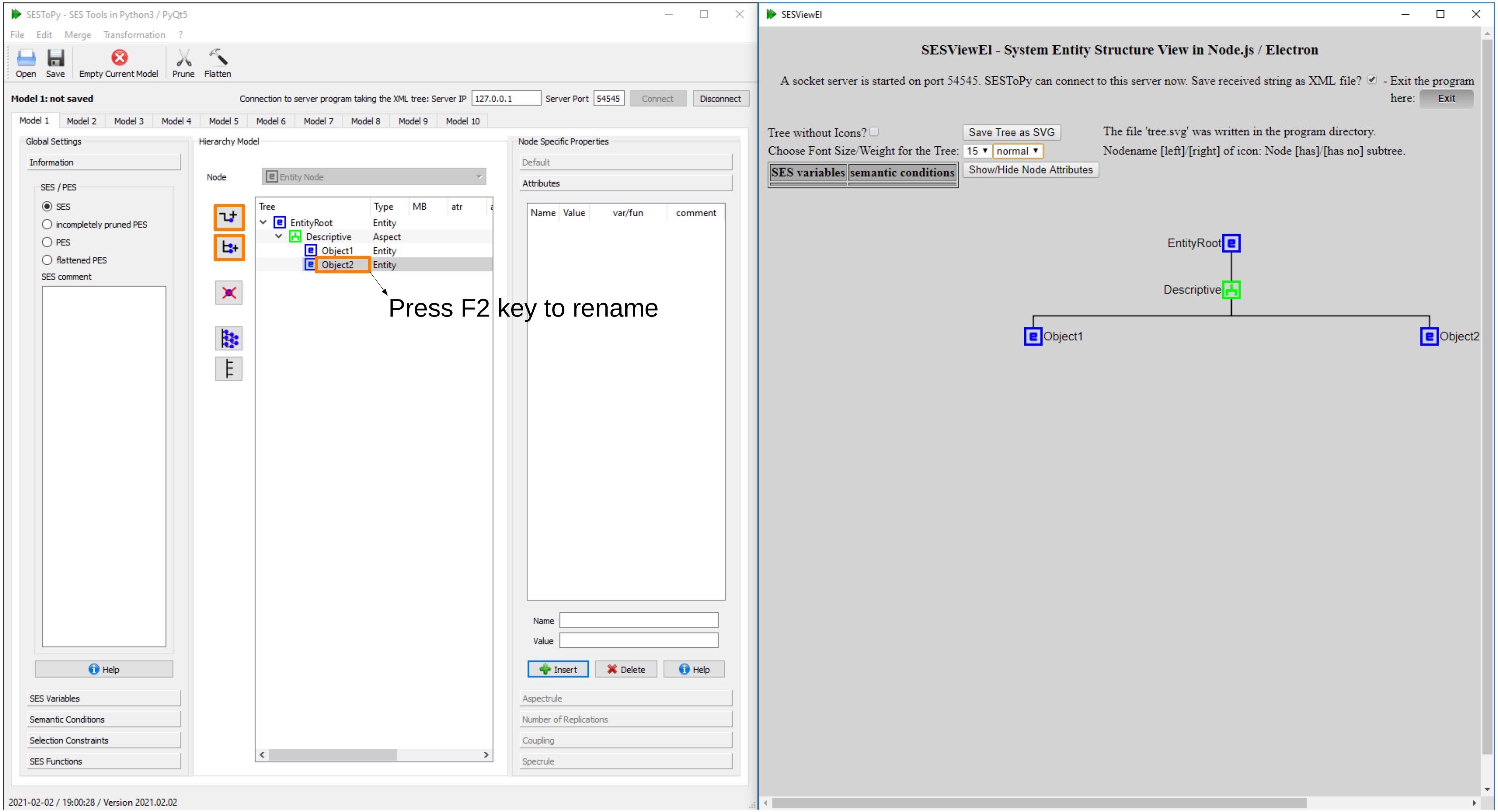
Create SES Tree with SESToPy



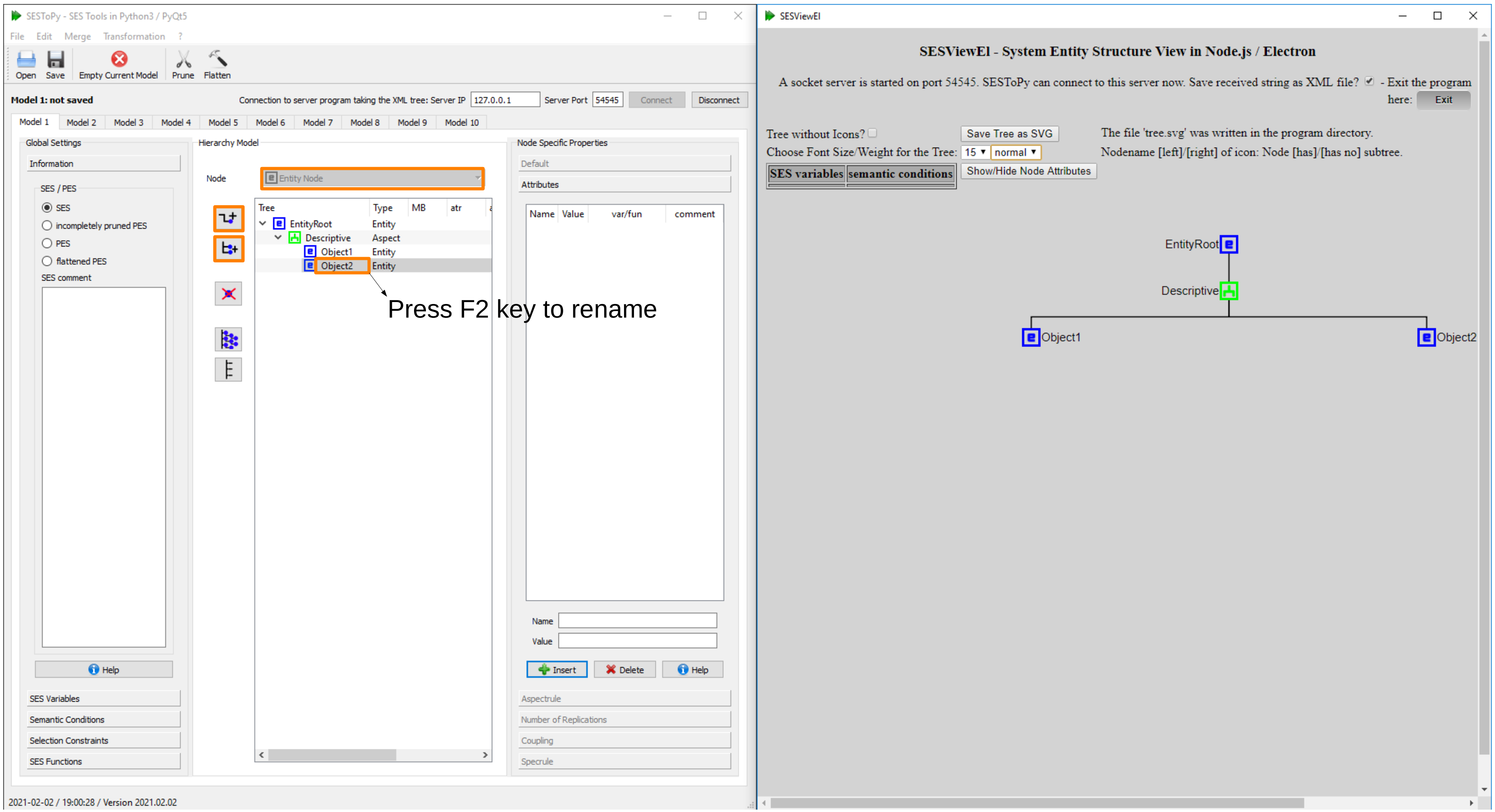
Create SES Tree with SESToPy



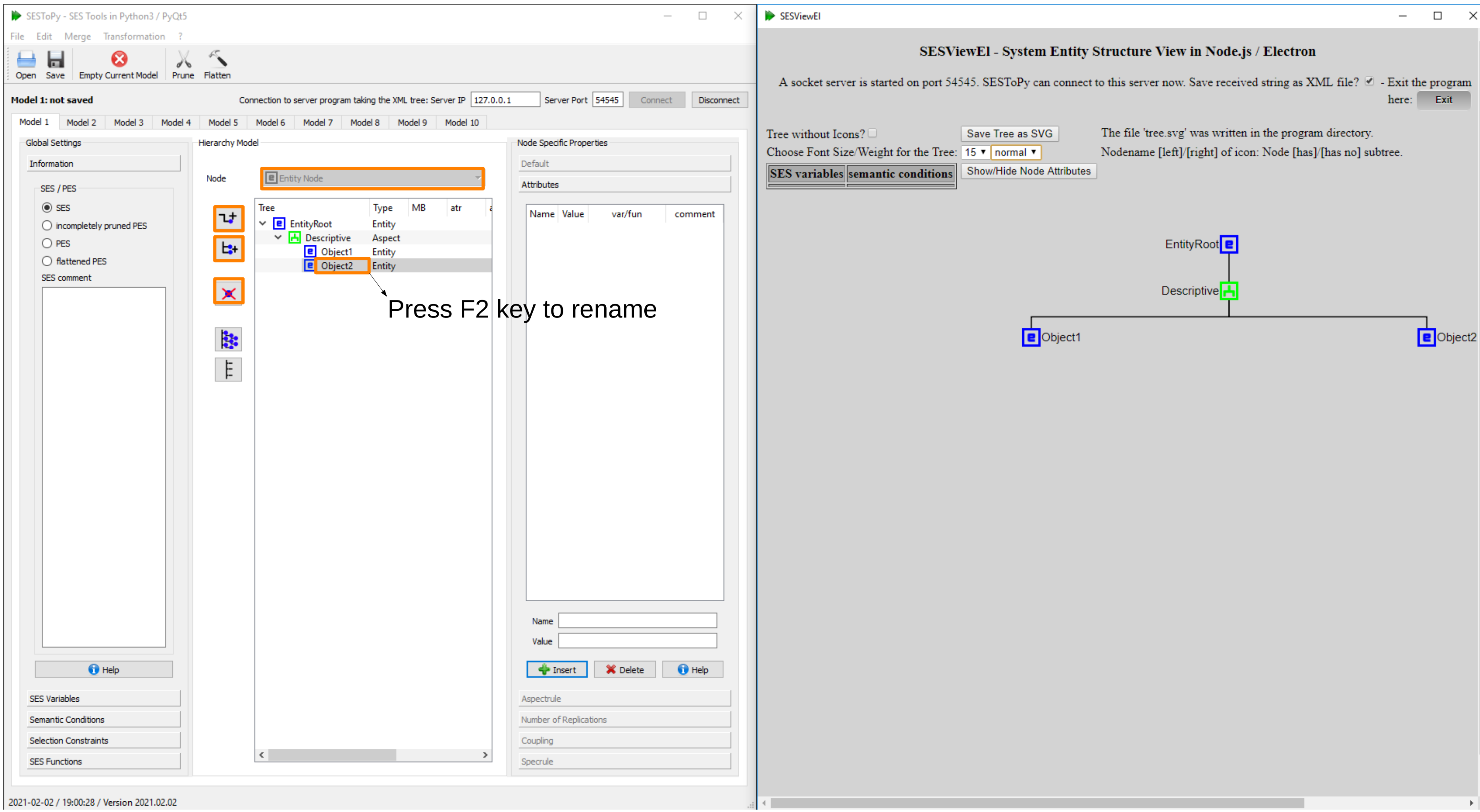
Create SES Tree with SESToPy



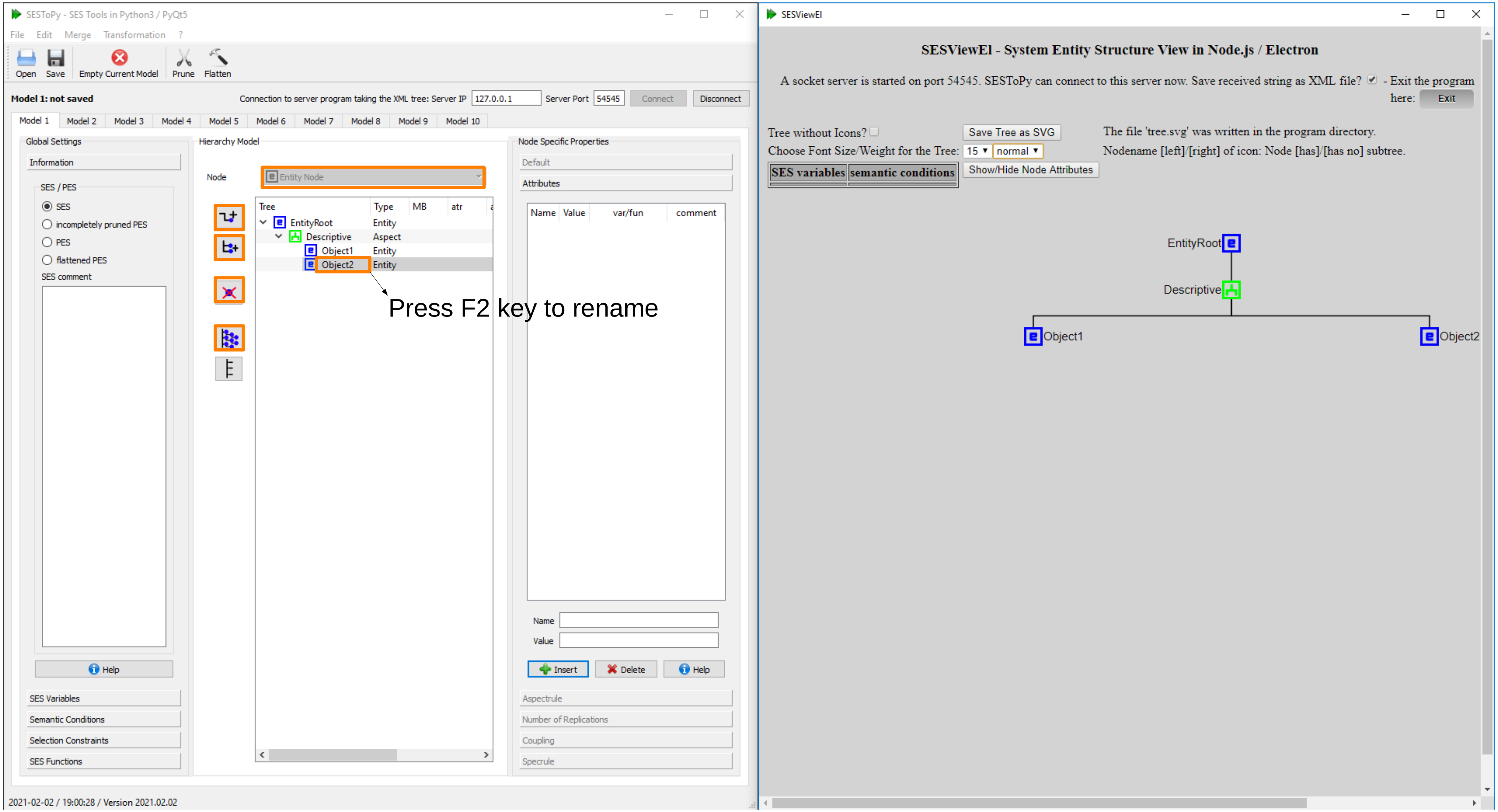
Create SES Tree with SESToPy



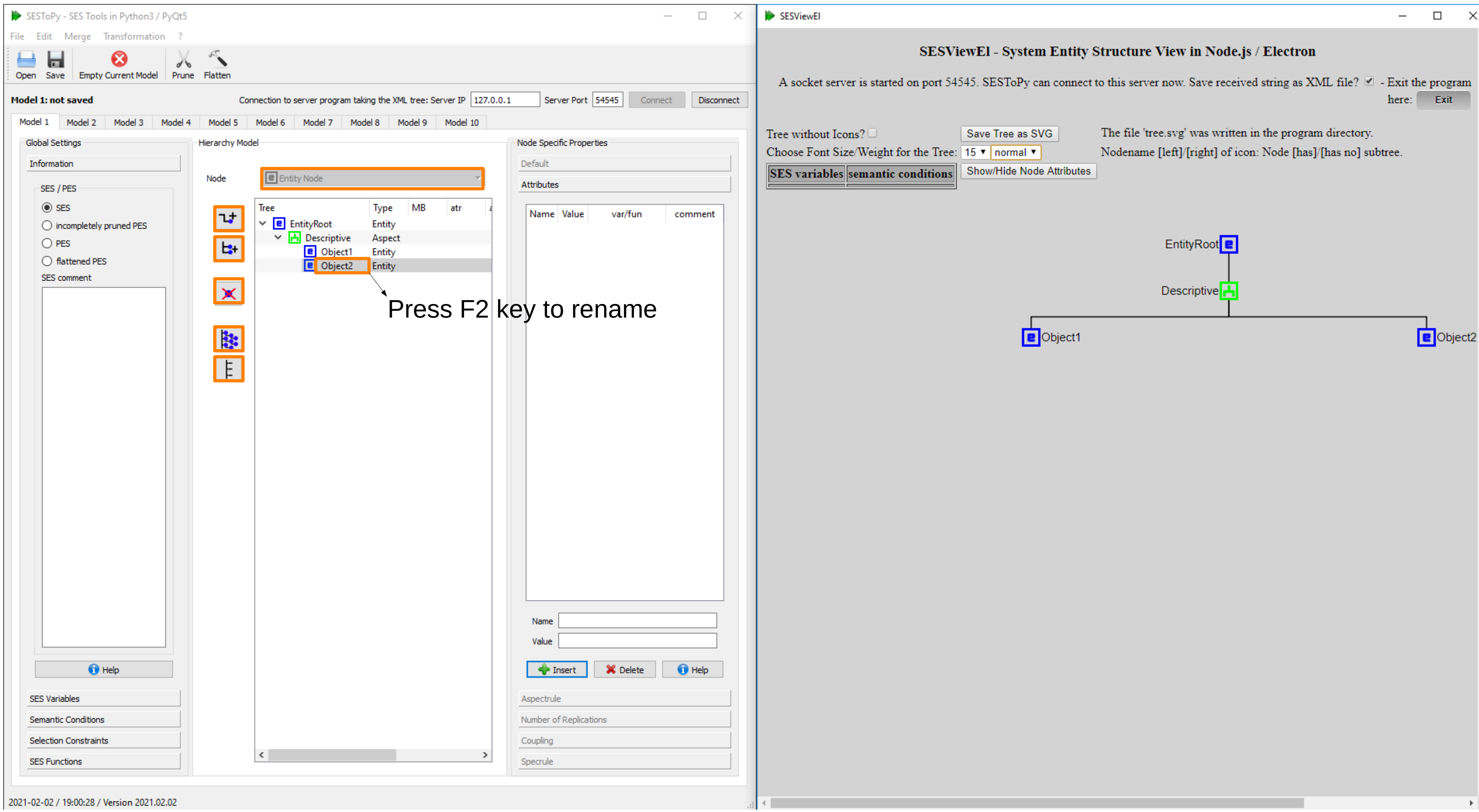
Create SES Tree with SESToPy



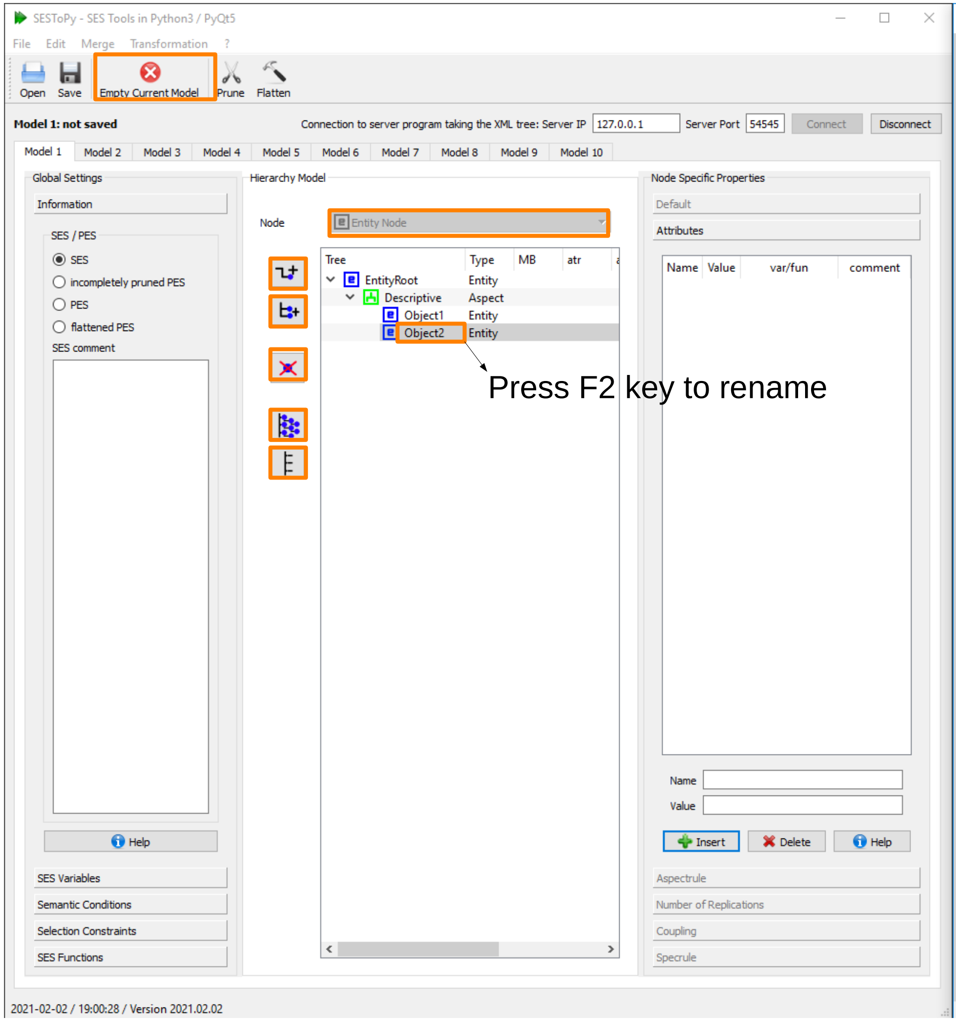
Create SES Tree with SESToPy

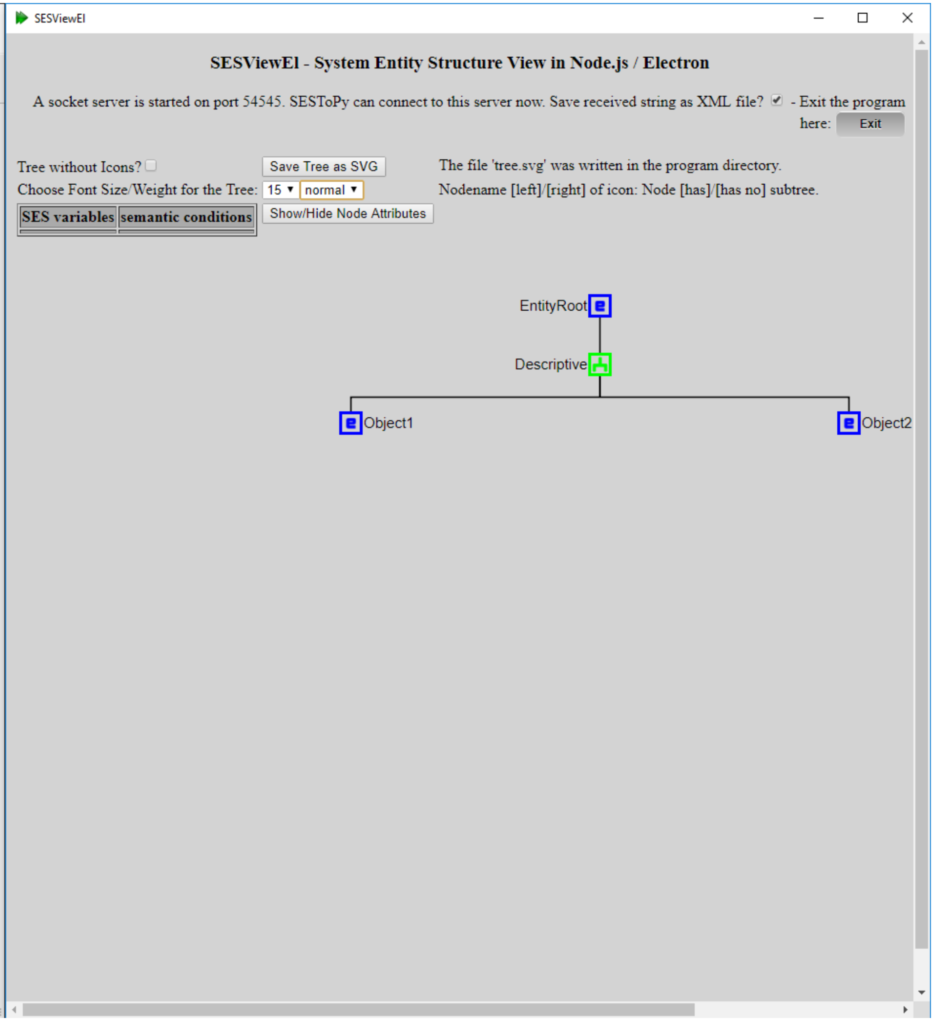


Create SES Tree with SESToPy

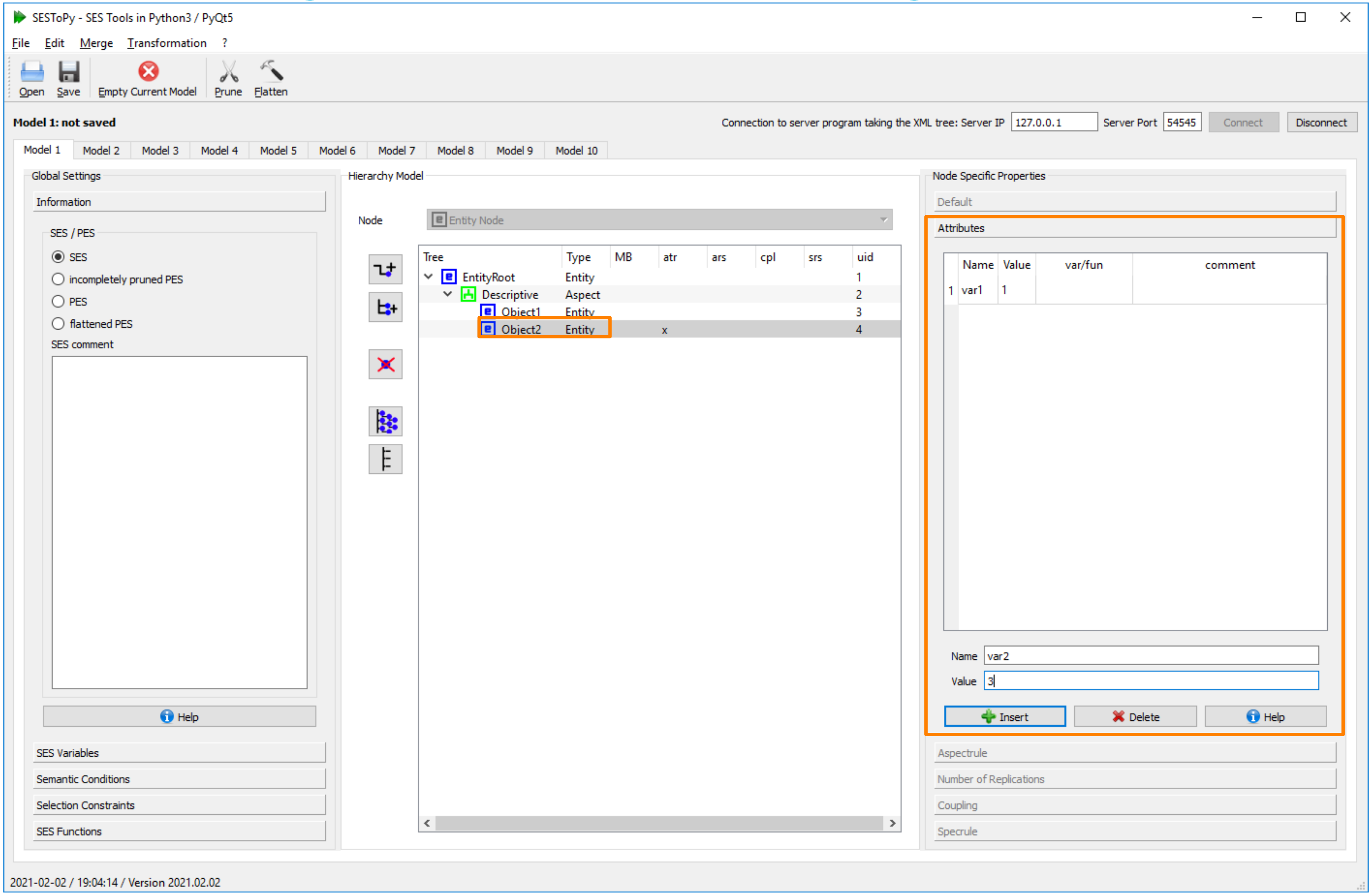


Create SES Tree with SESToPy

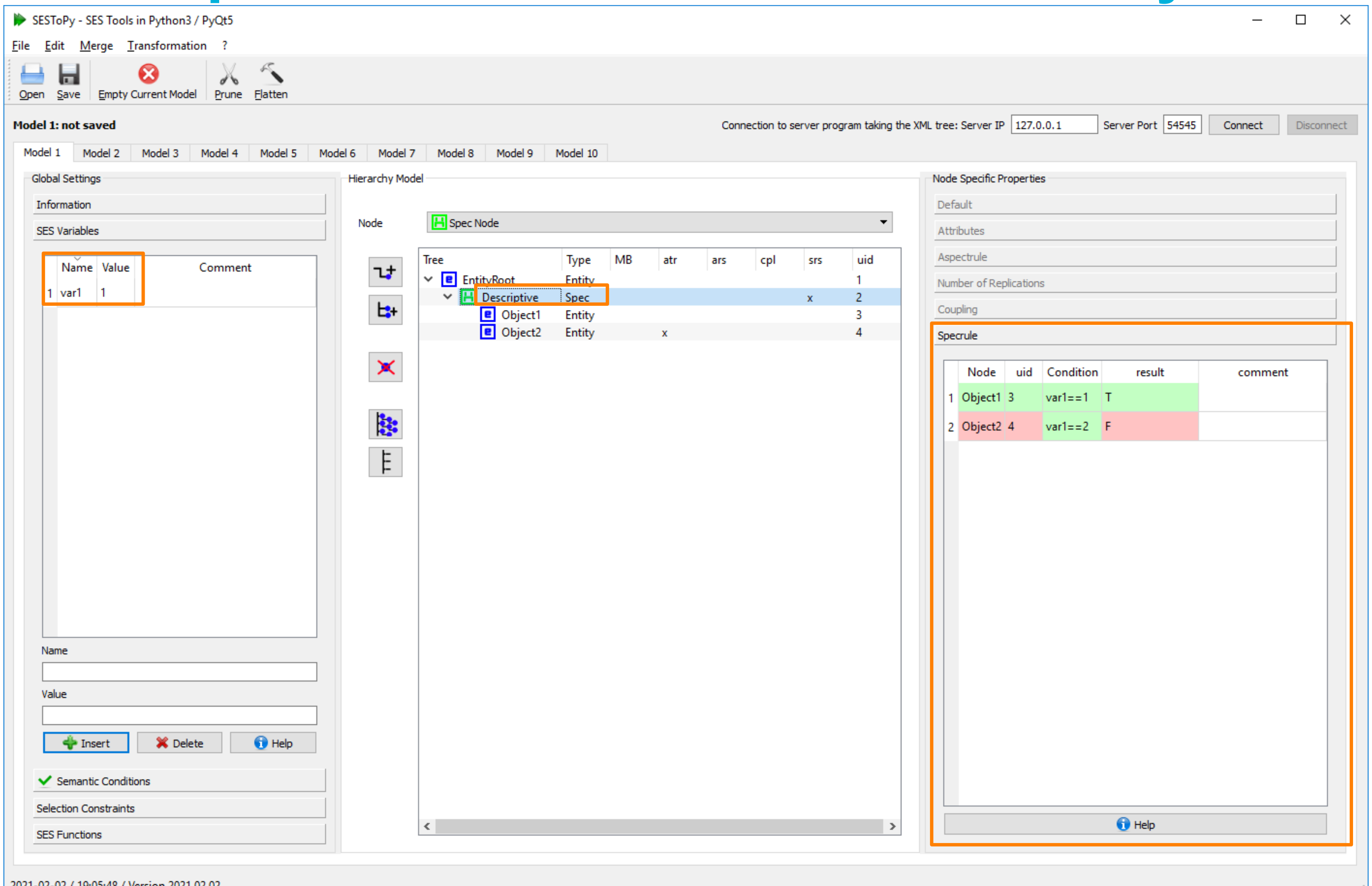




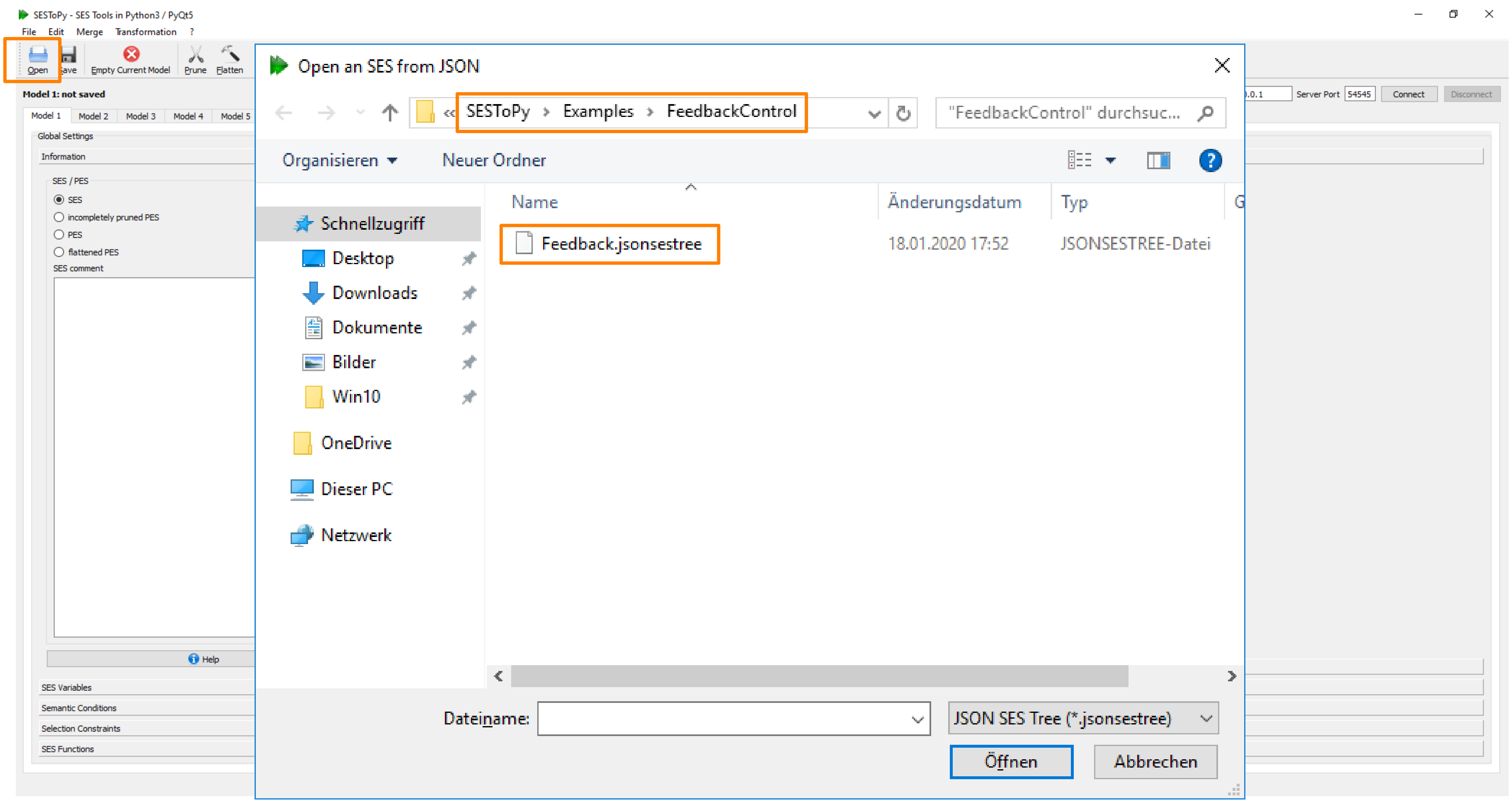
Edit Entity Node with SESToPy



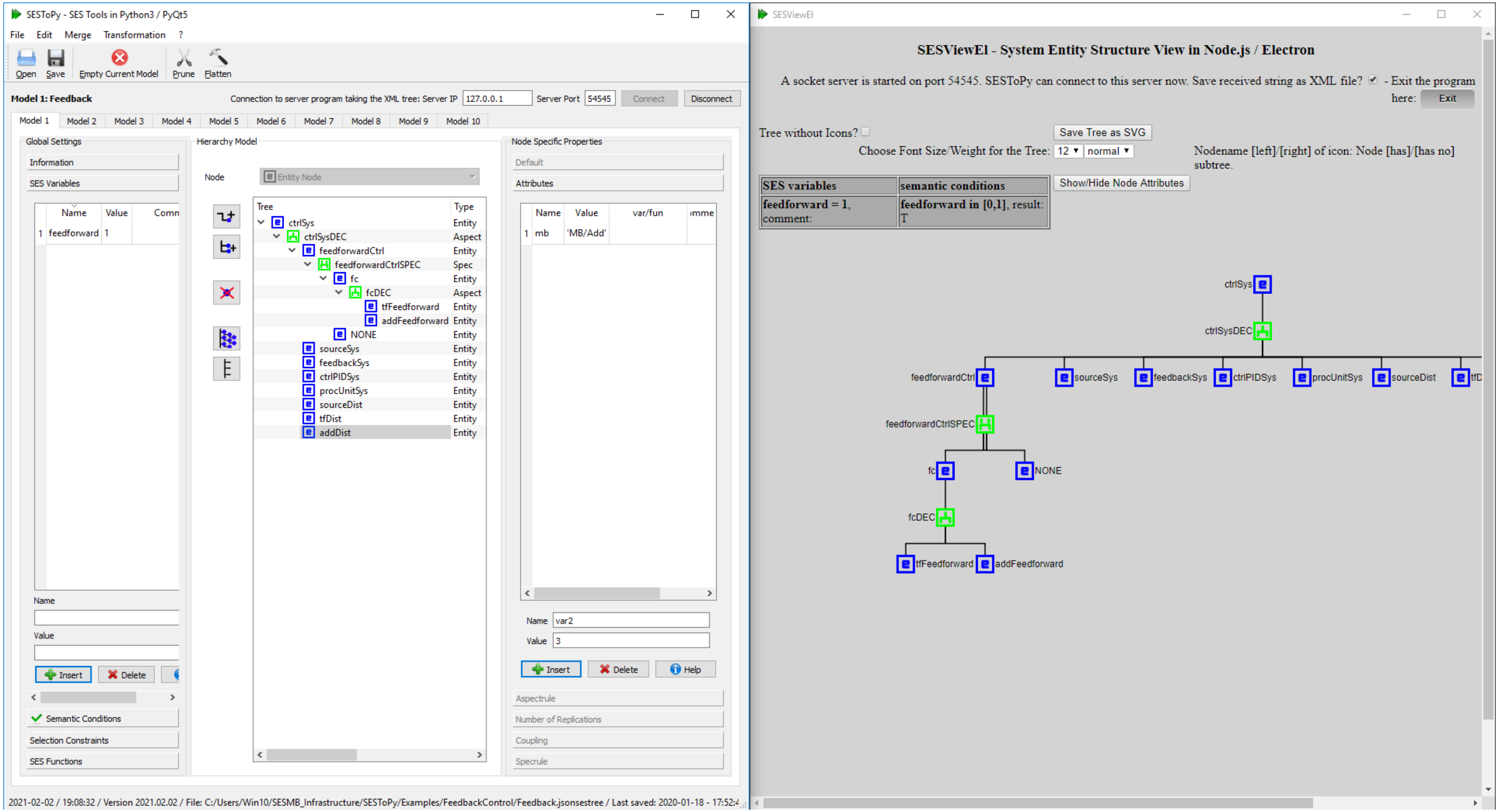
Edit Specialization Node with SESToPy



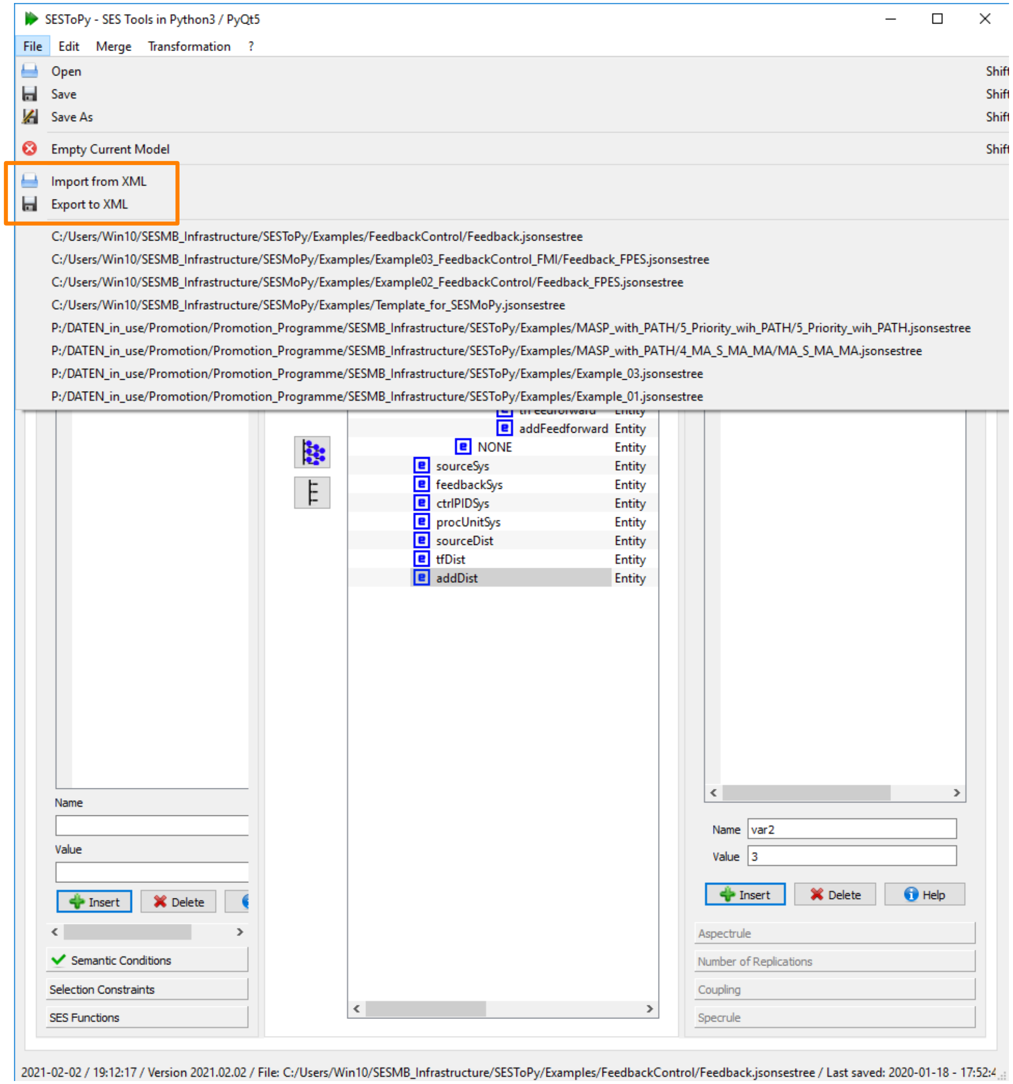
SESToPy Open the Feedback Example



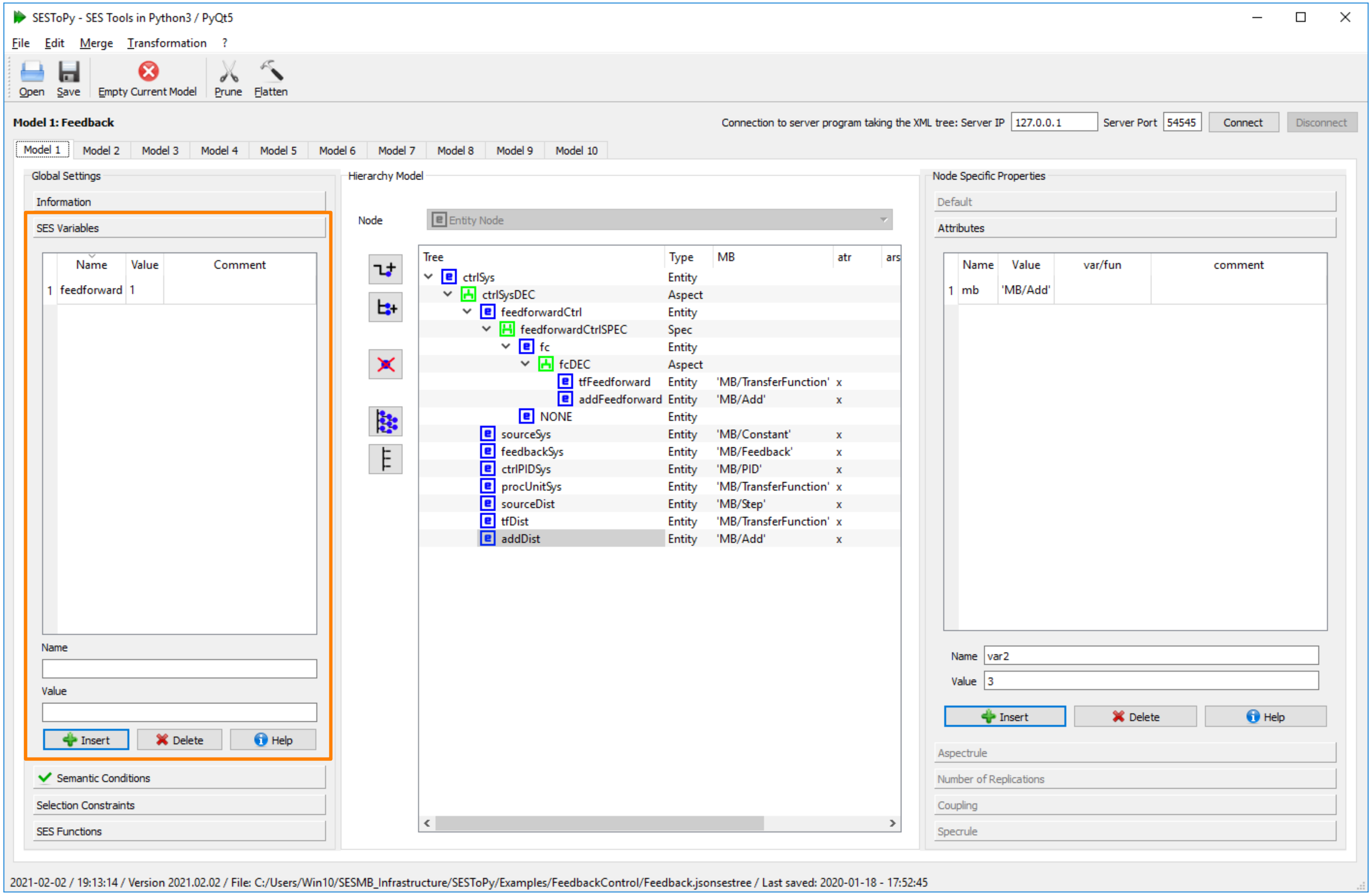
SESToPy Feedback Example



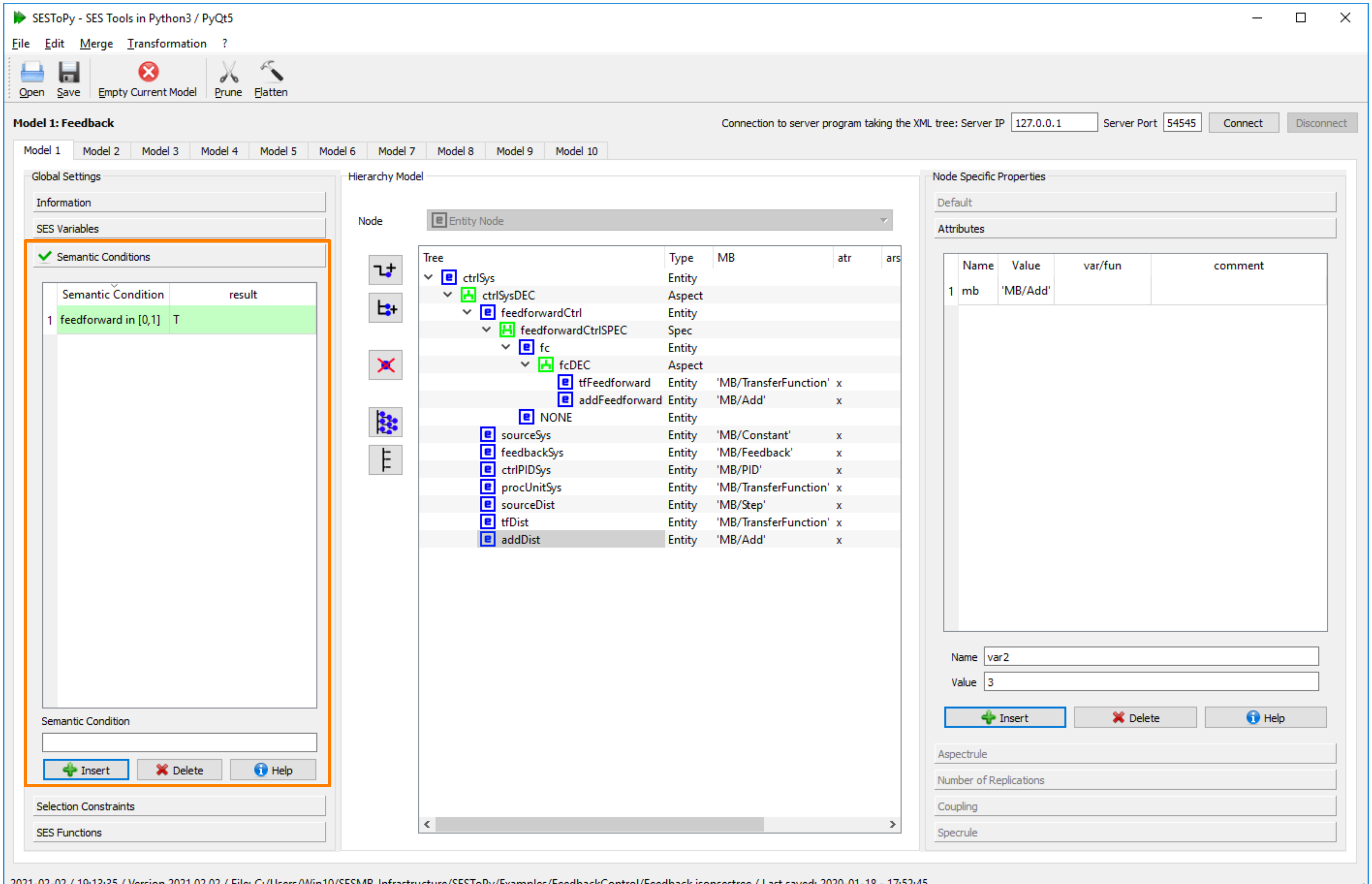
SESToPy XML Export / Import



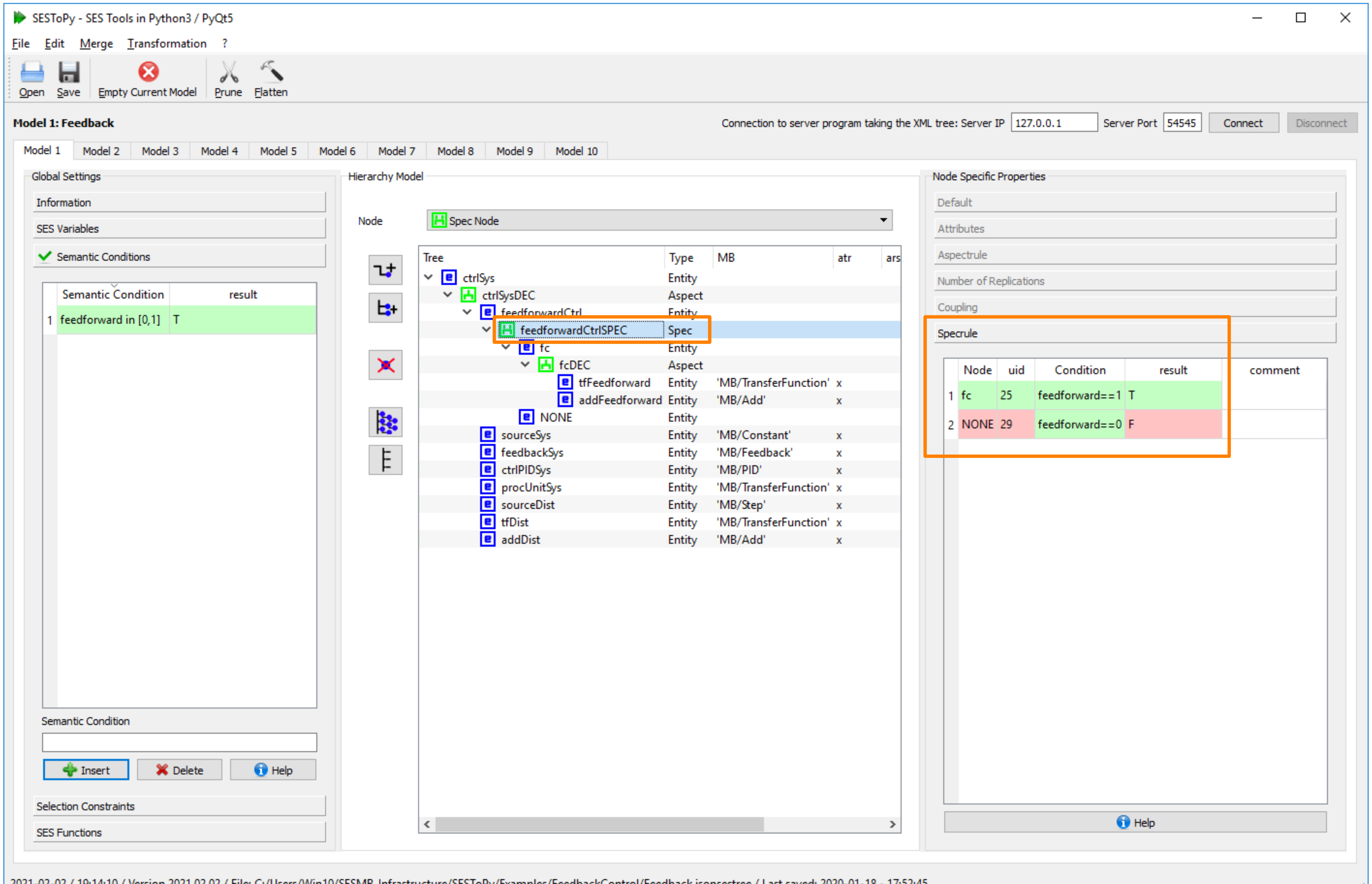
SESToPy SES Variables



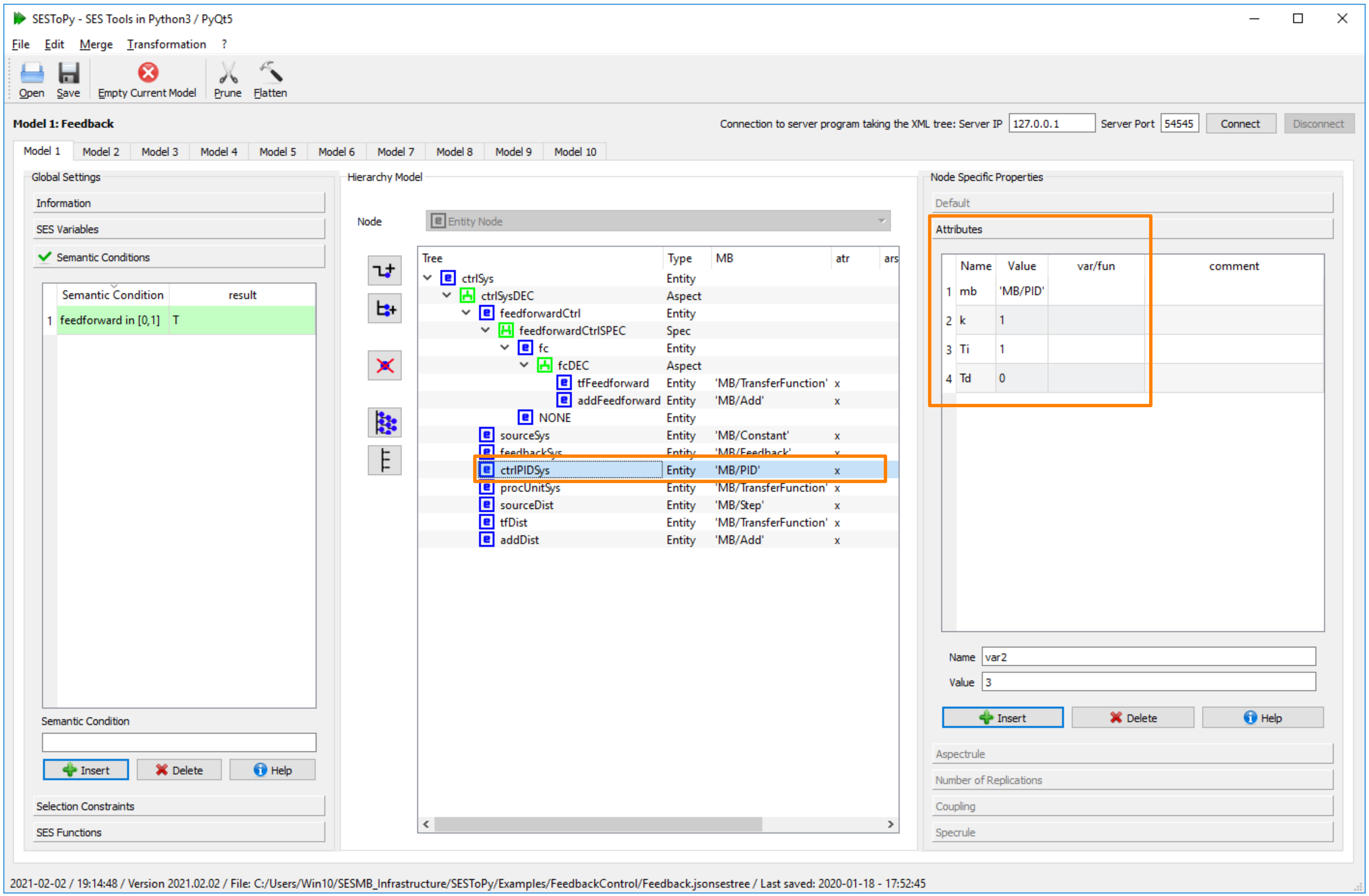
SESToPy Semantic Conditions



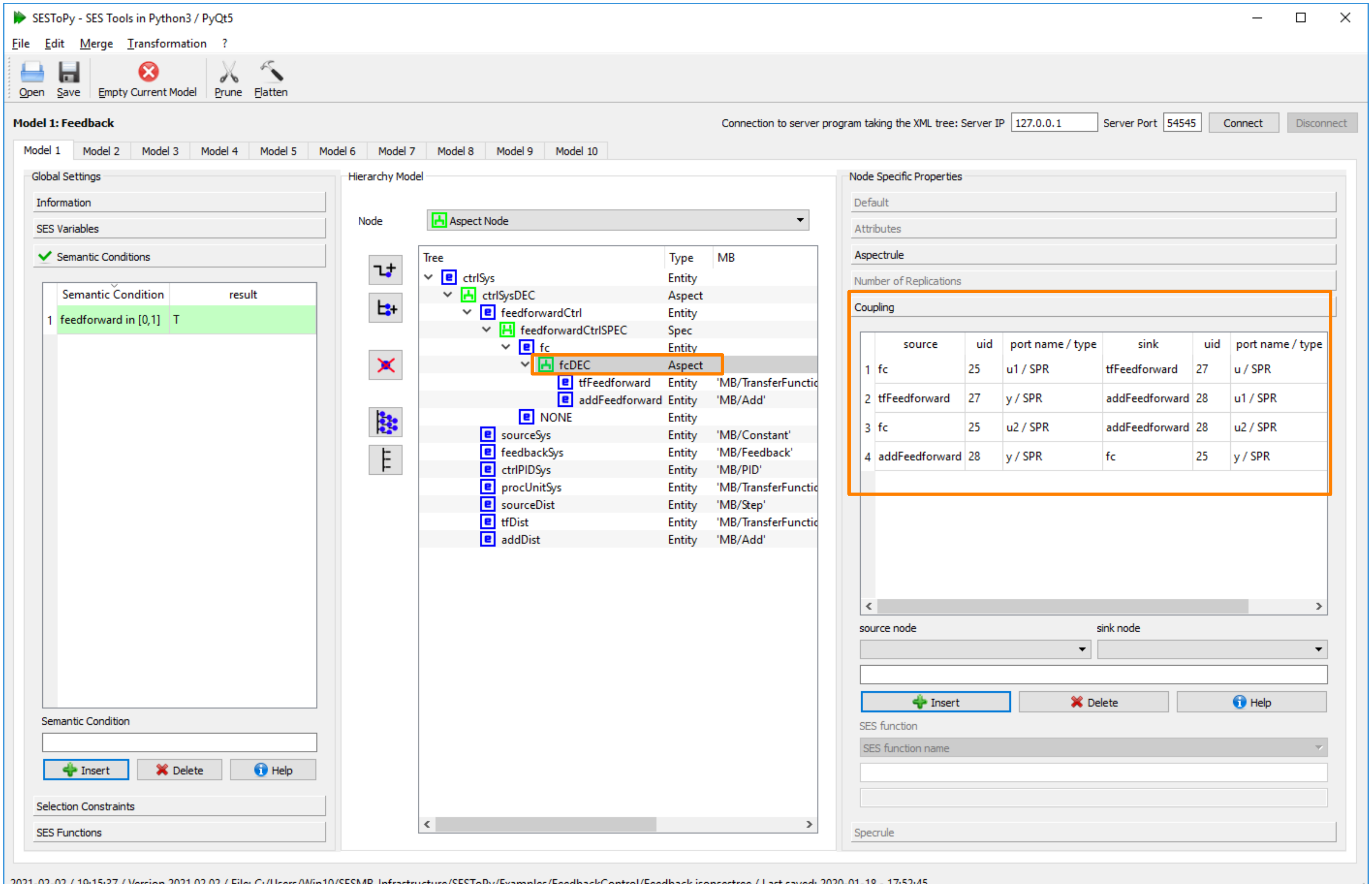
SESToPy Selection Rules (here Specrule)



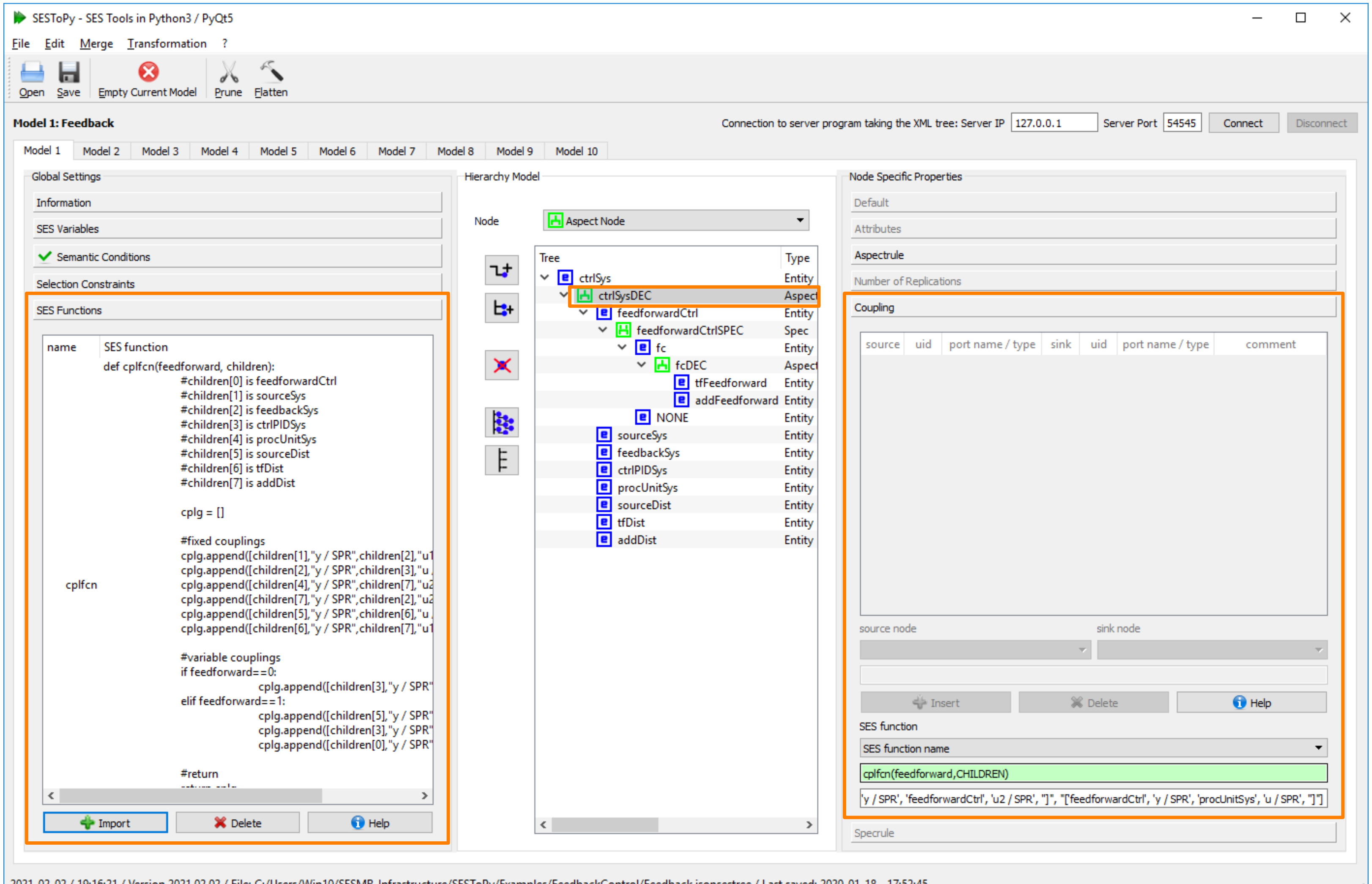
SESToPy Attributes



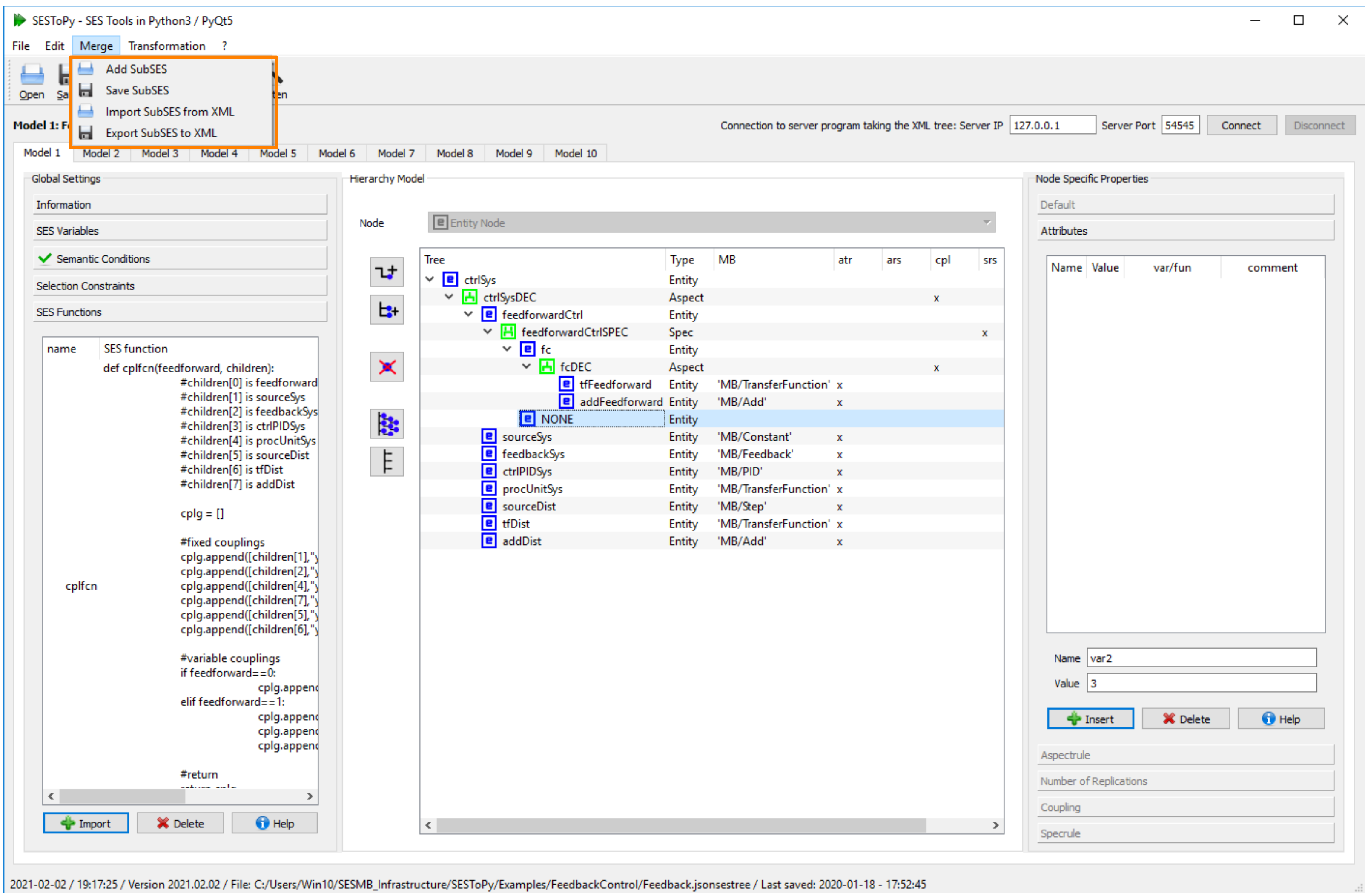
SESToPy Coupling List



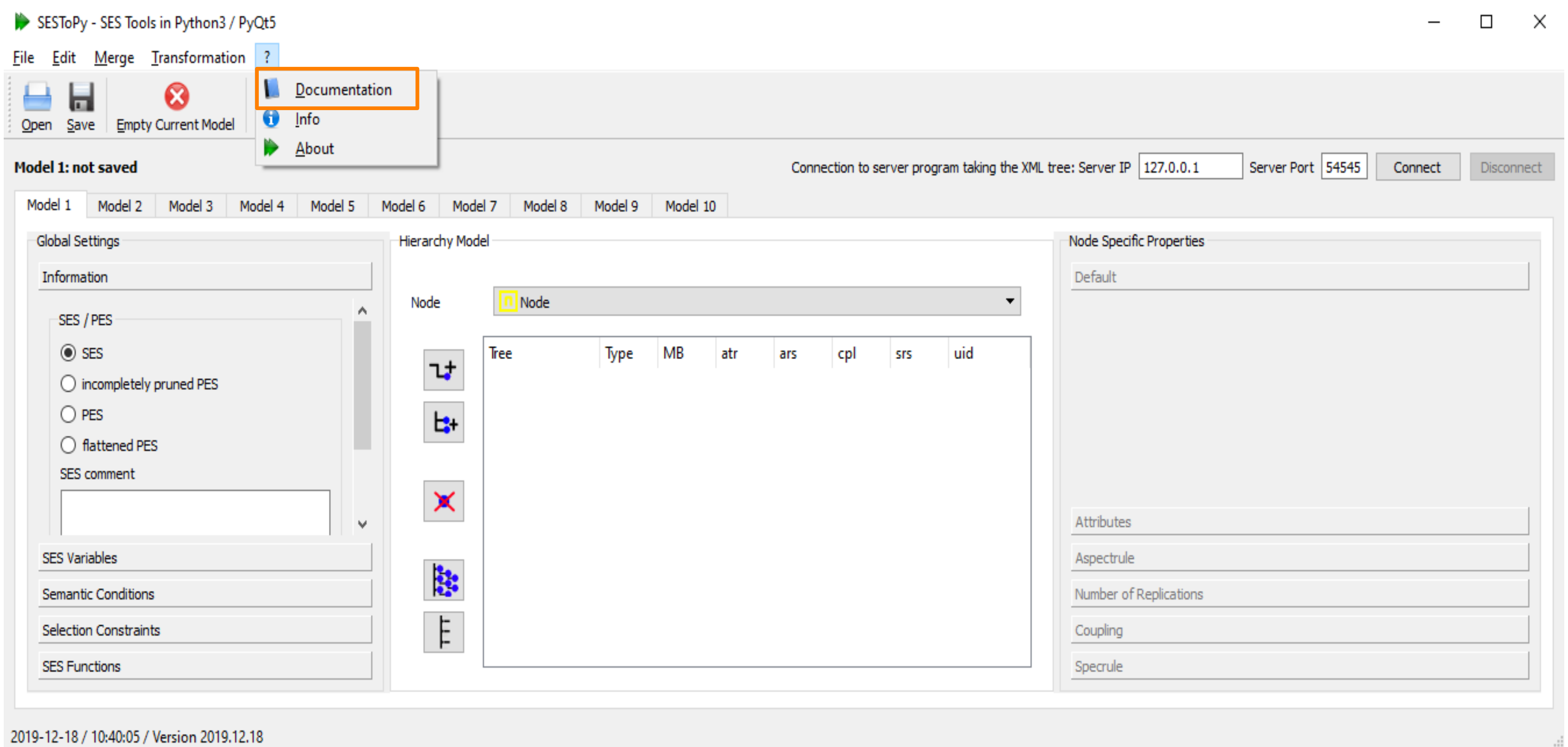
SESToPy SES Function / Coupling Function



SESToPy Merging



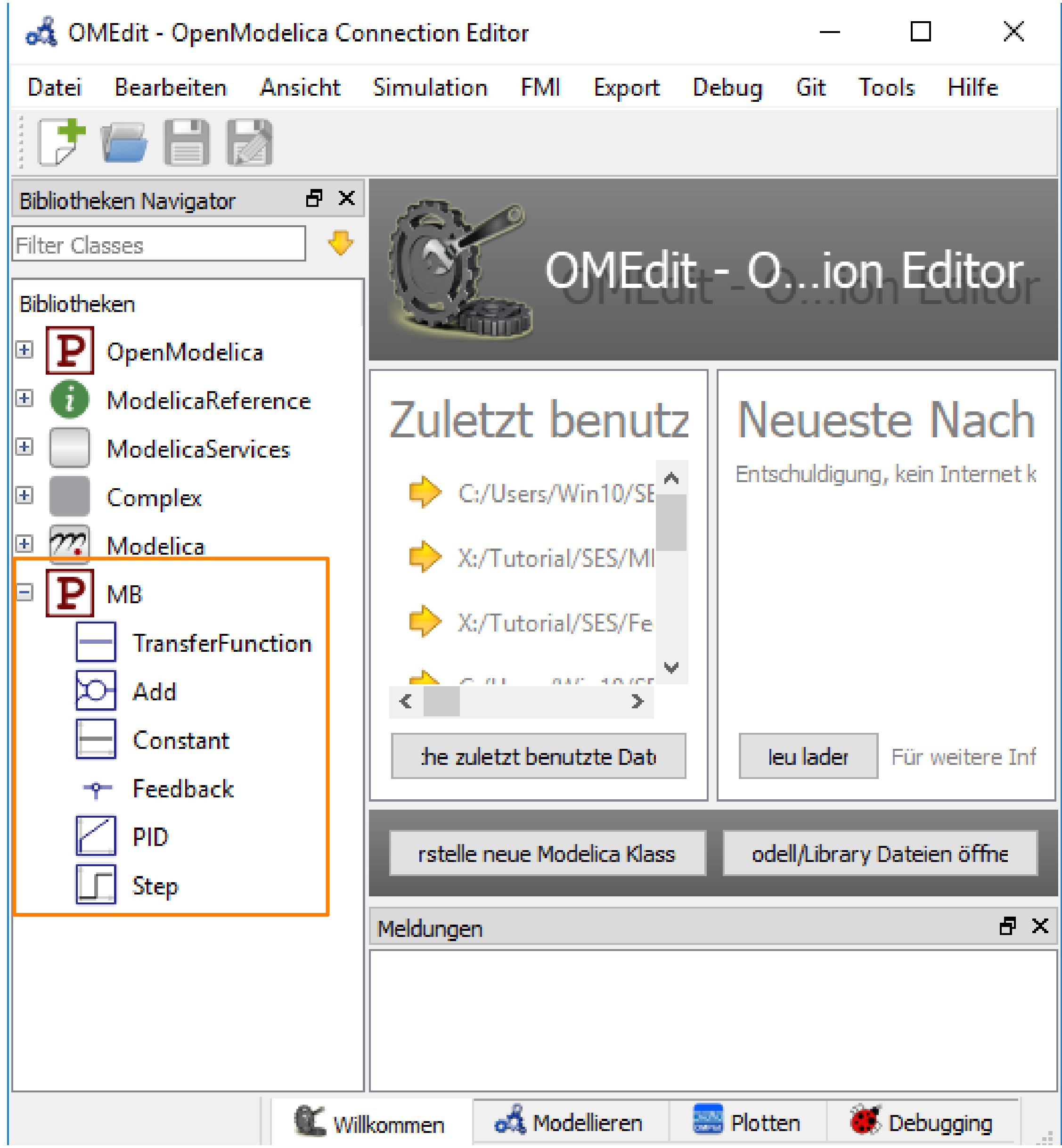
SESToPy Documentation



- See the documentation for more information

OpenModelica MB

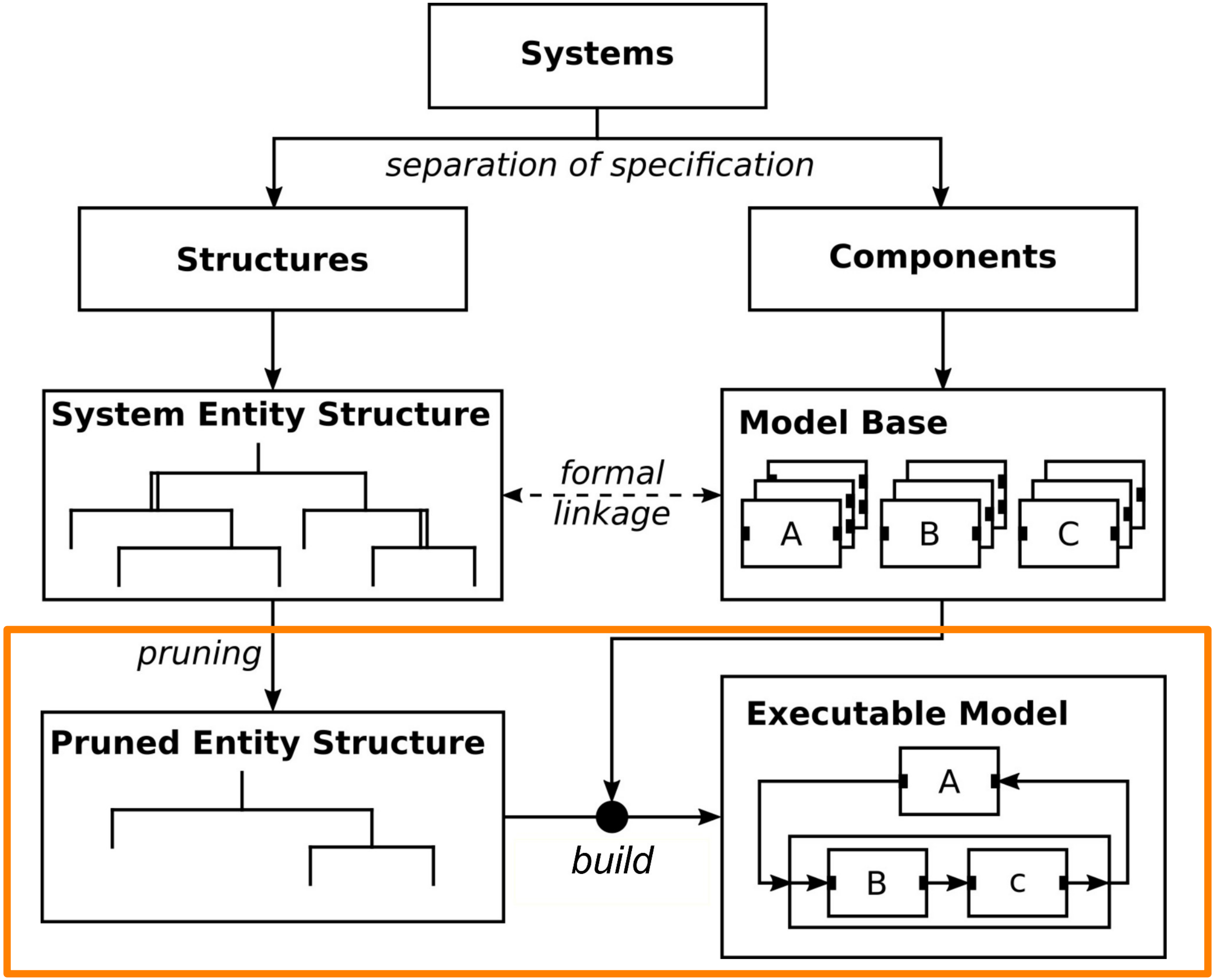
- MB built of Modelica basic models in a **Package**
- Save package as `MB.mo` file



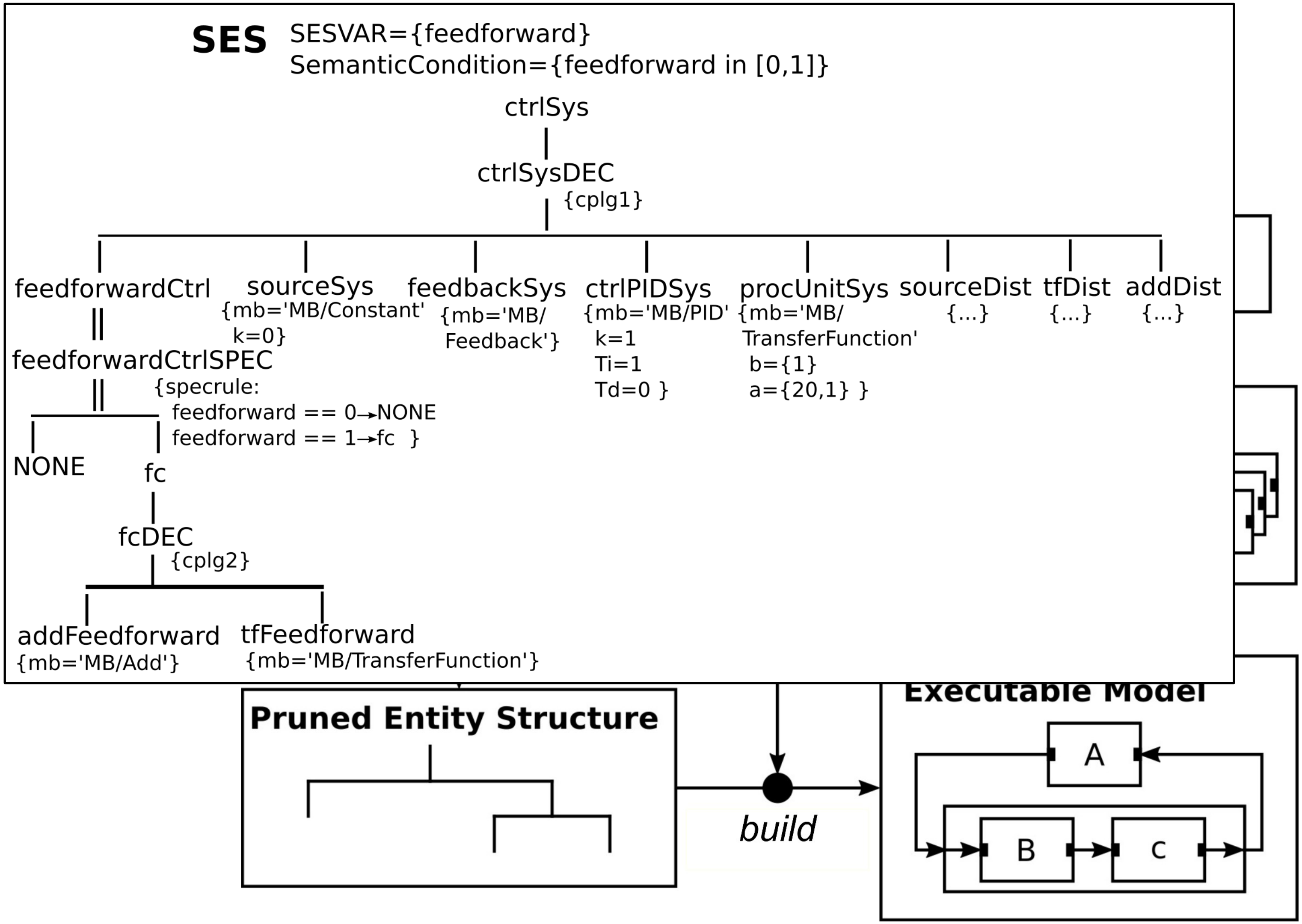
Outline

1. Case study
2. Implementation of the SES and an MB
3. **Model selection and model generation**
4. Organization of a simulator-independent MB
5. Full automation of simulation experiments
6. Summary

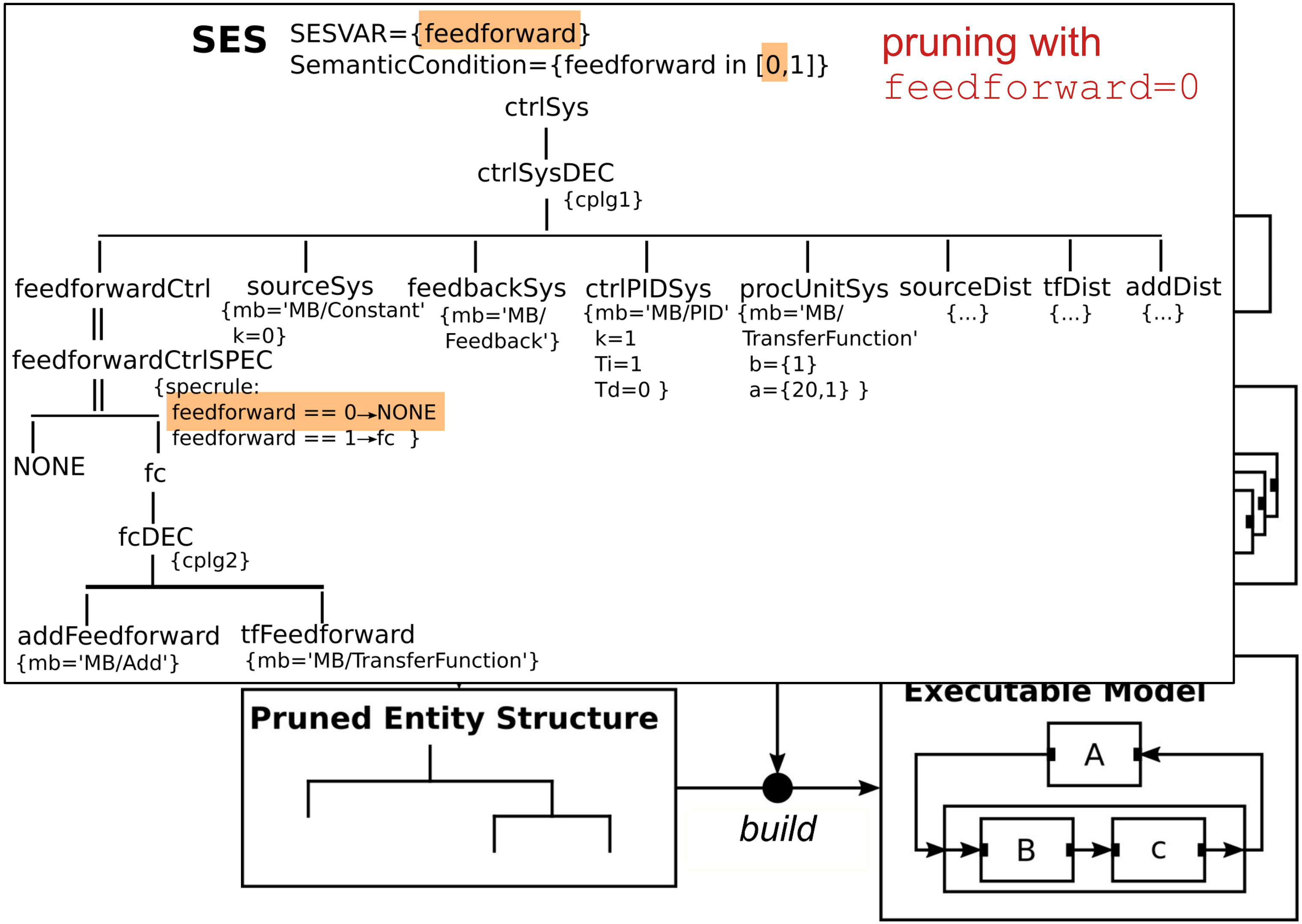
Model Selection and Generation



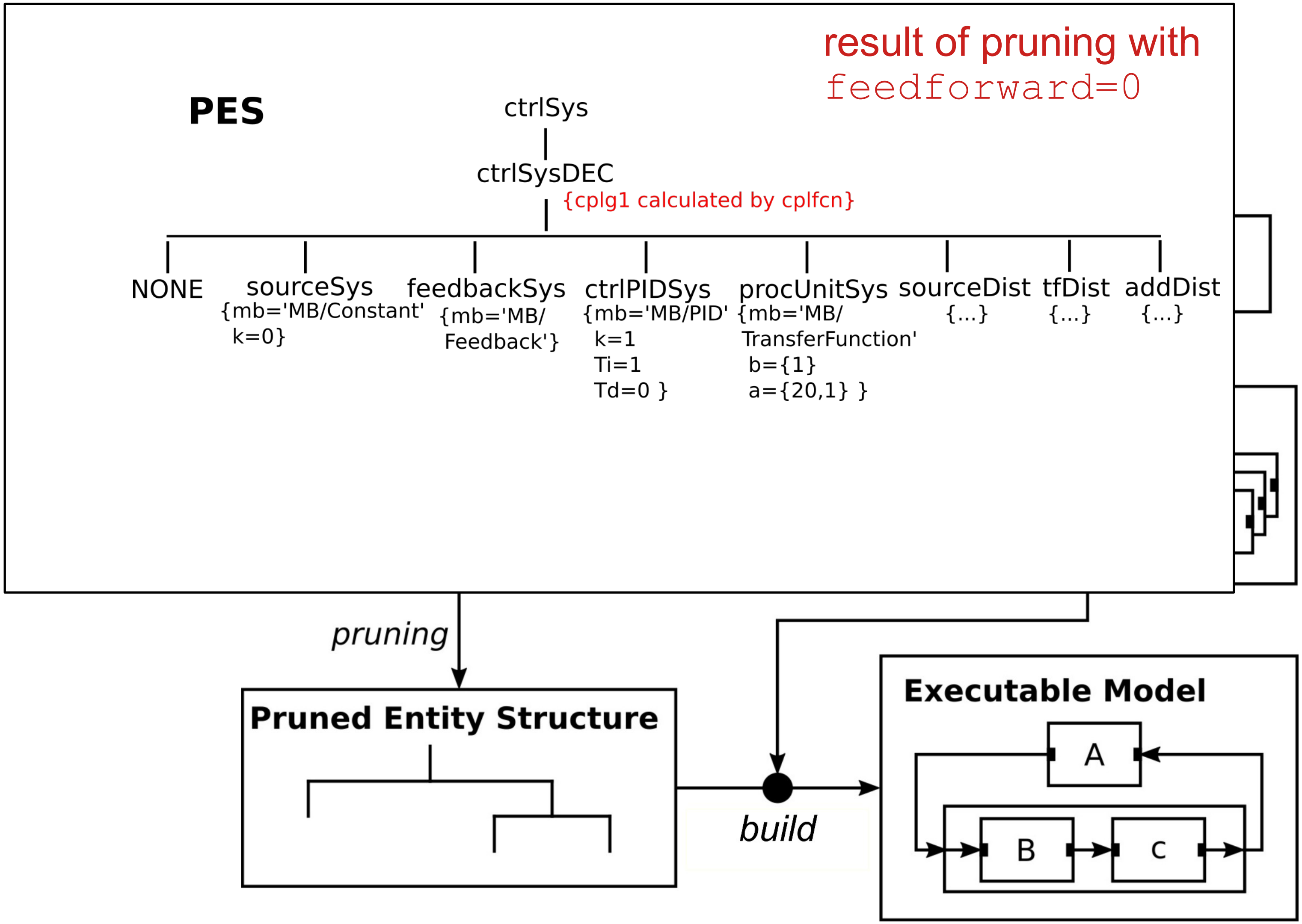
Model Selection and Generation of Variant #1



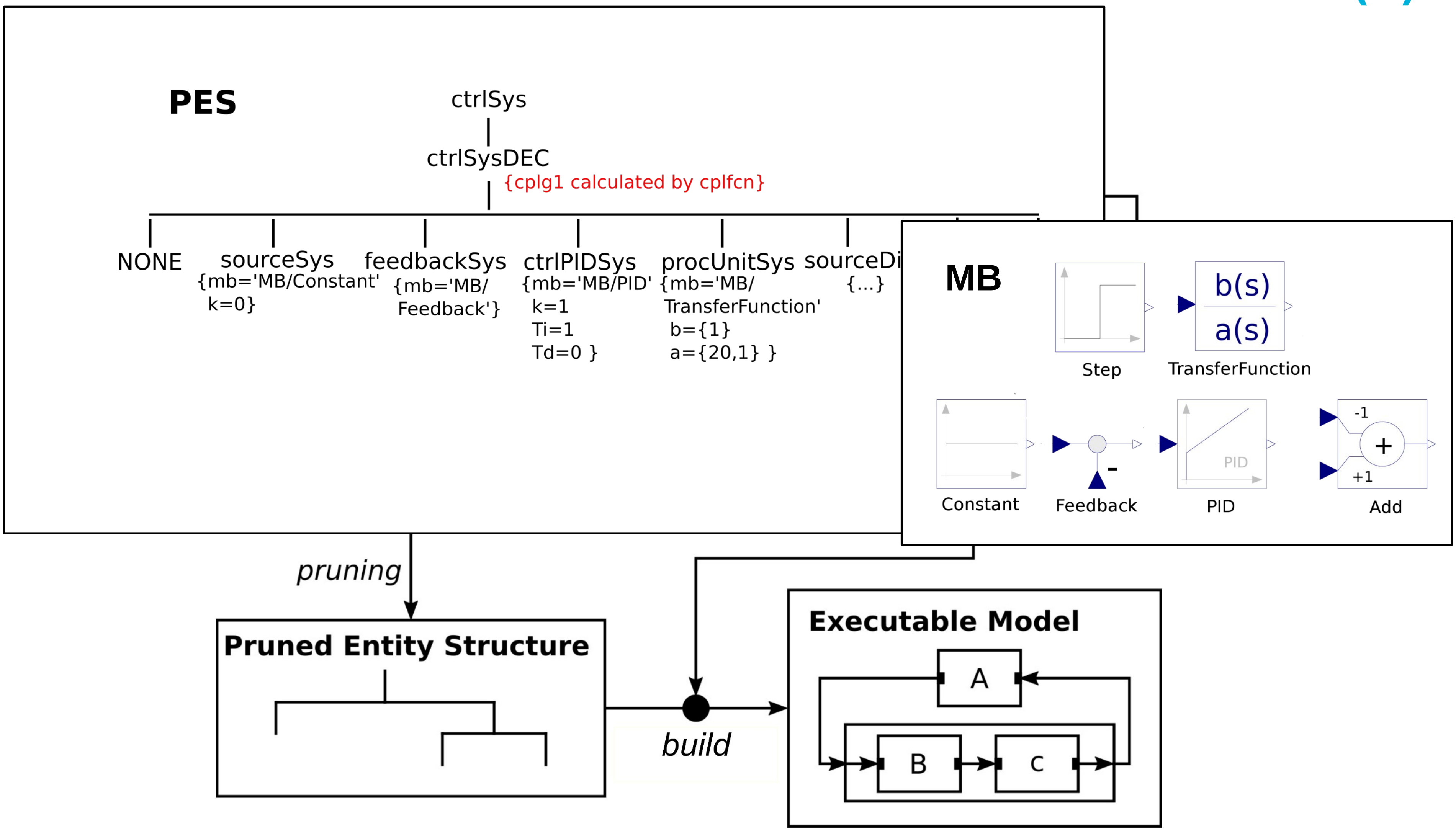
Model Selection and Generation of Variant #1



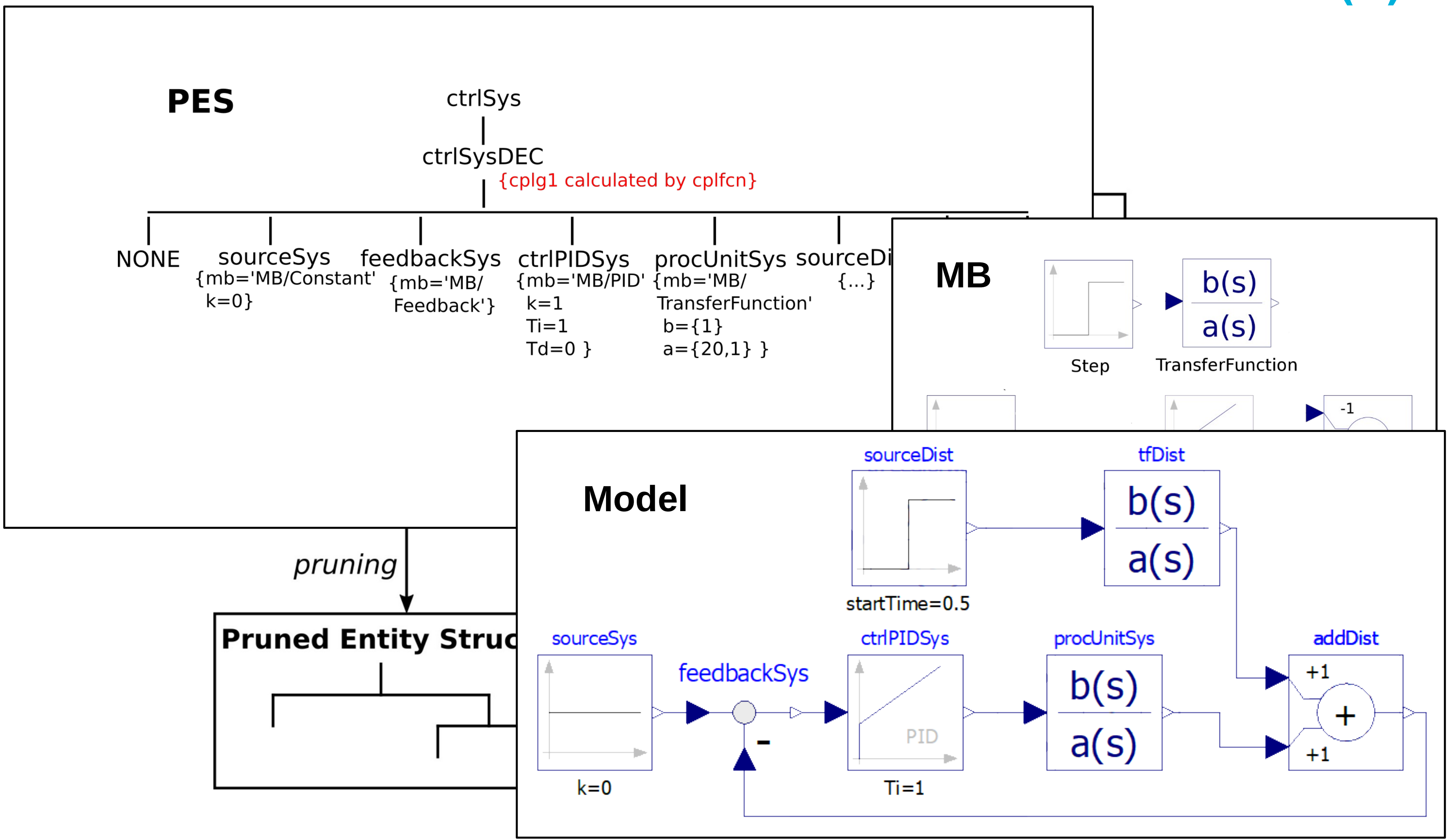
Model Selection and Generation of Variant #1 (2)



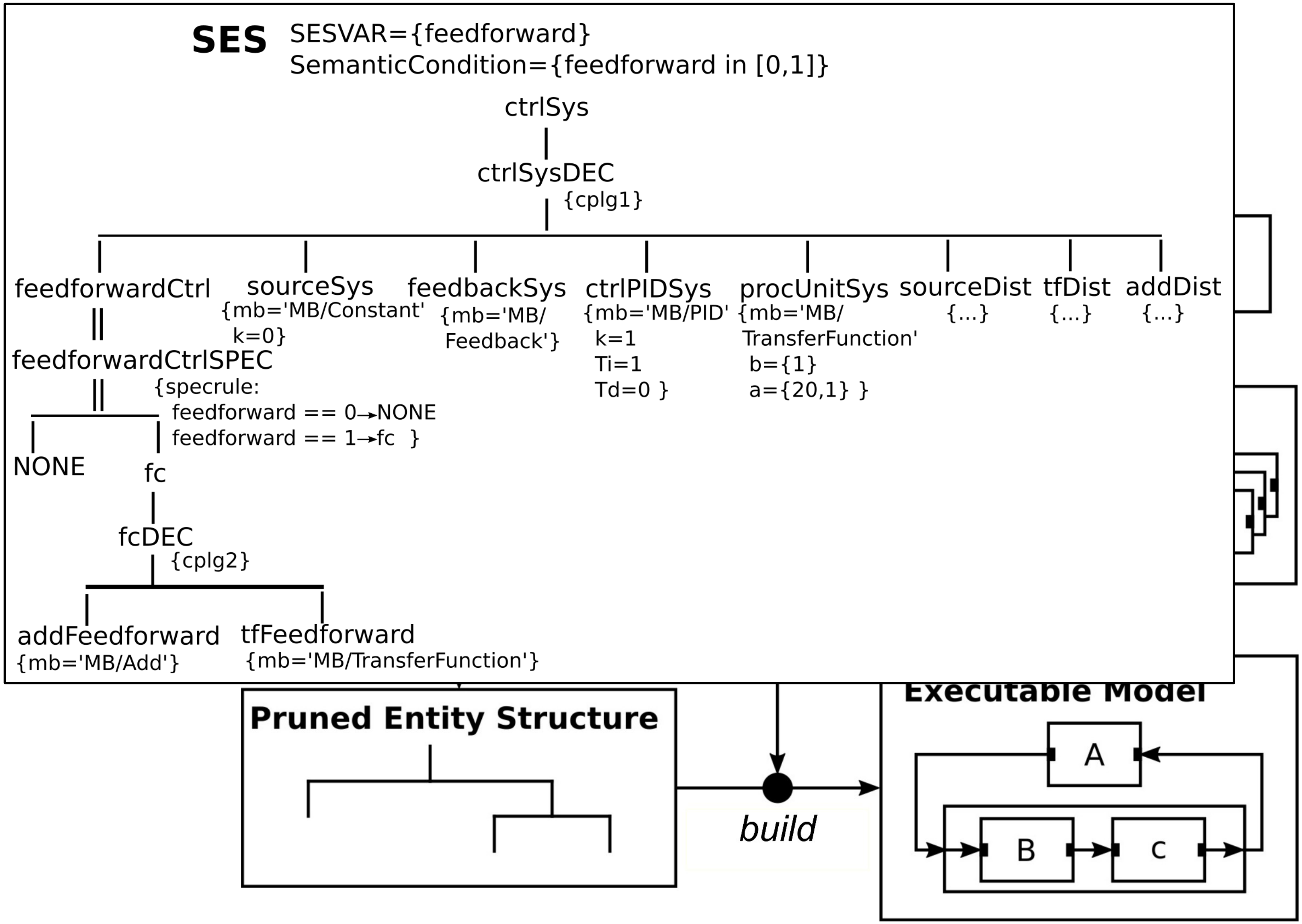
Model Selection and Generation of Variant #1 (3)



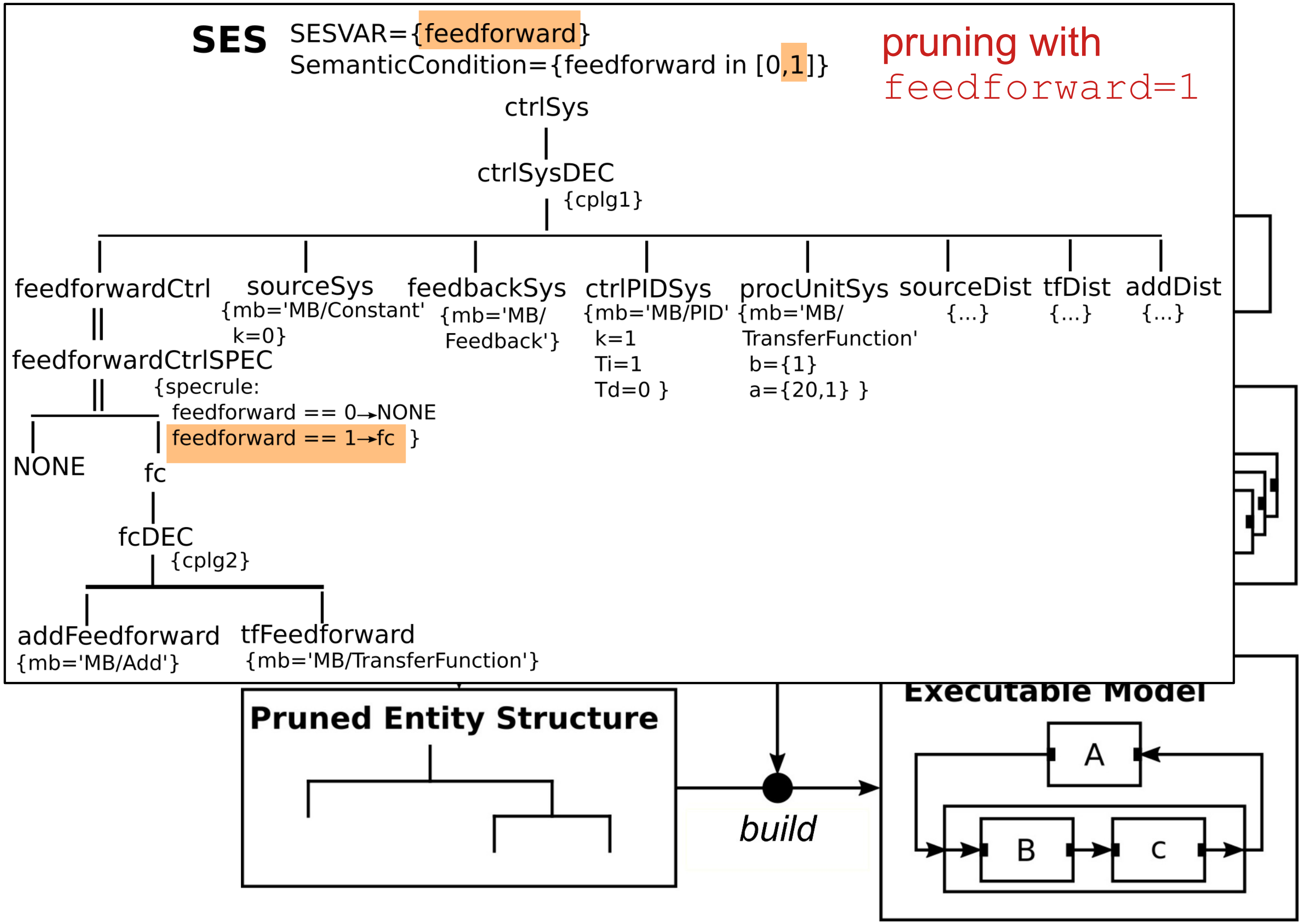
Model Selection and Generation of Variant #1 (4)



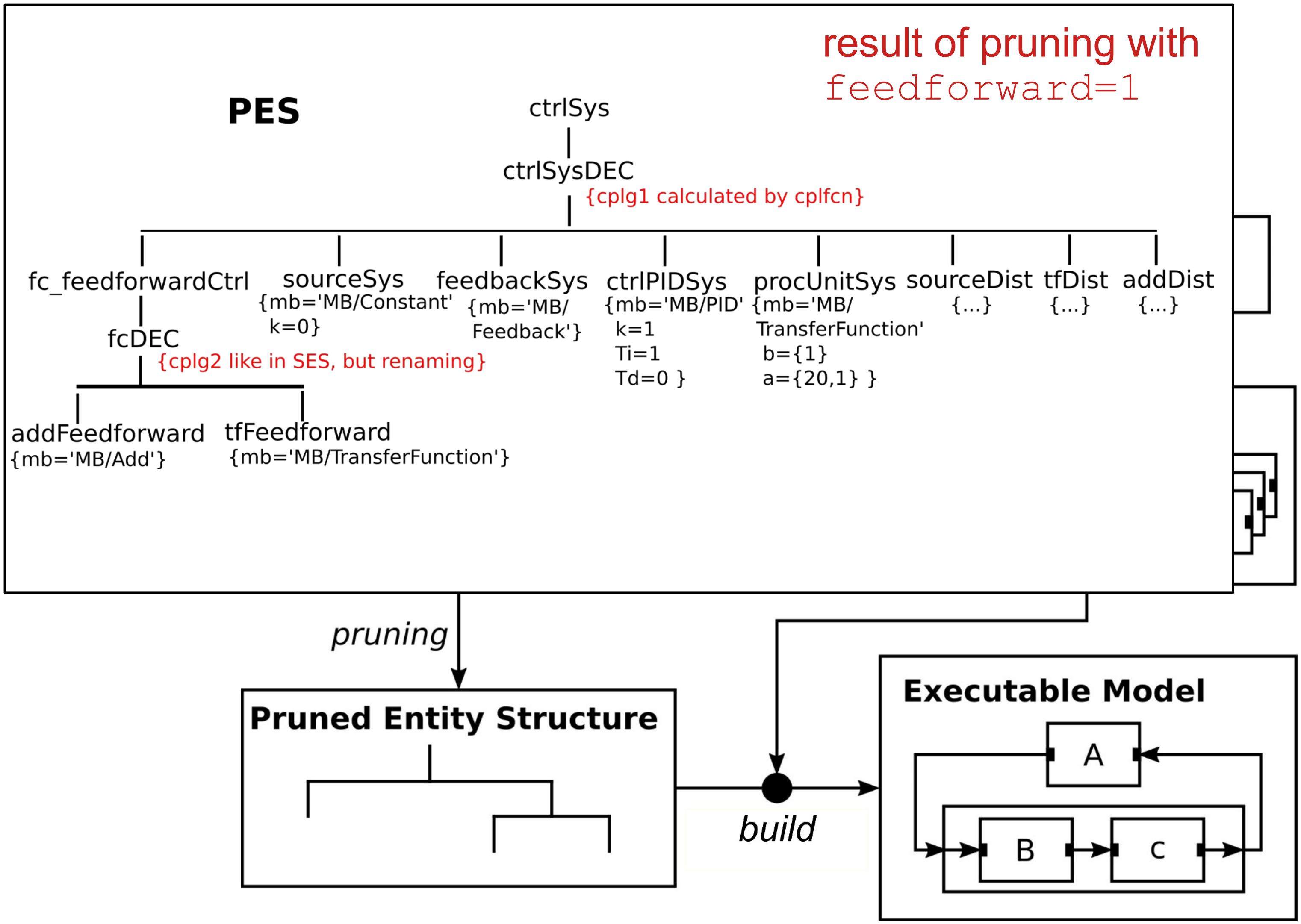
Model Selection and Generation of Variant #2



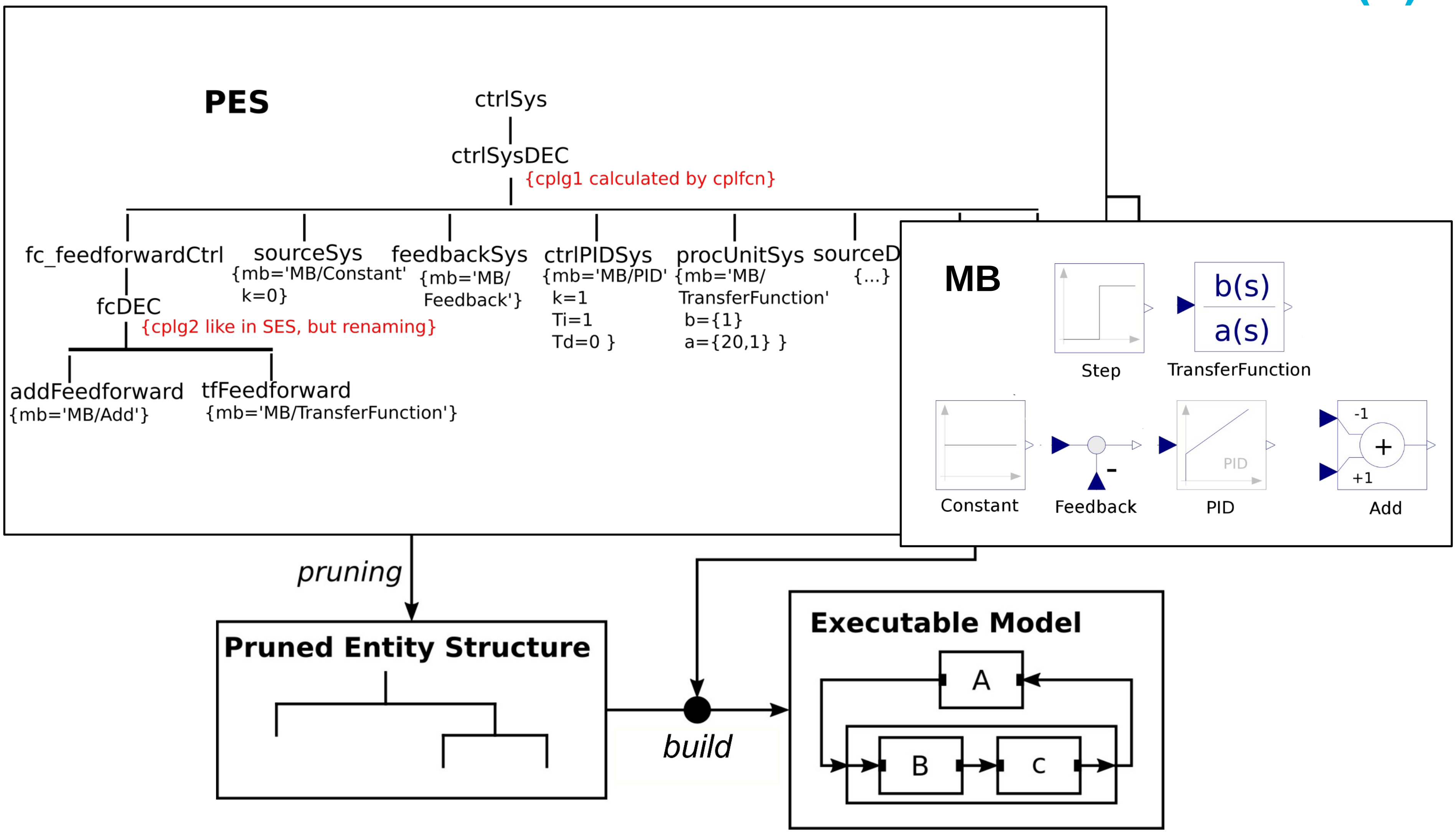
Model Selection and Generation of Variant #2



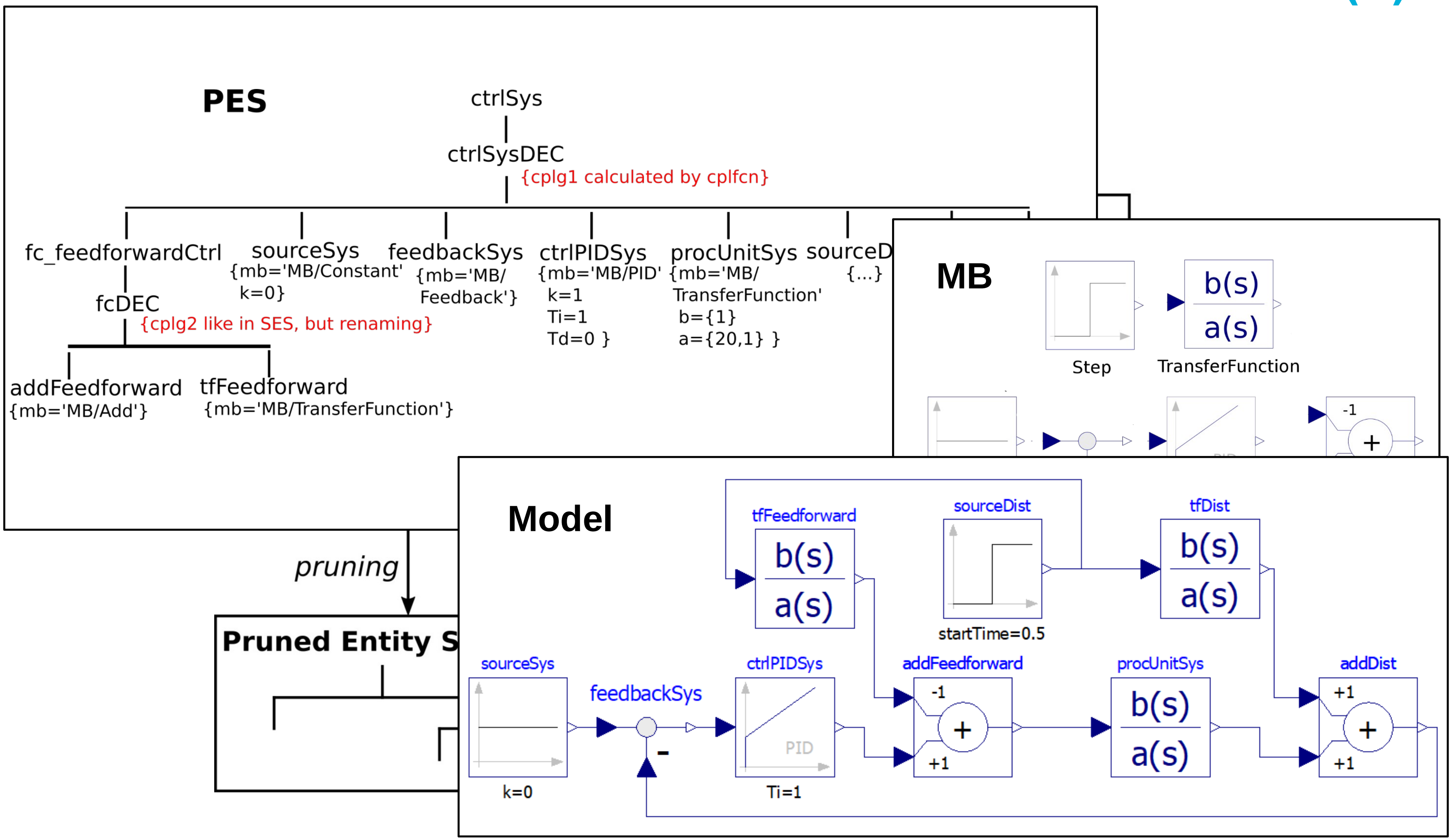
Model Selection and Generation of Variant #2 (2)



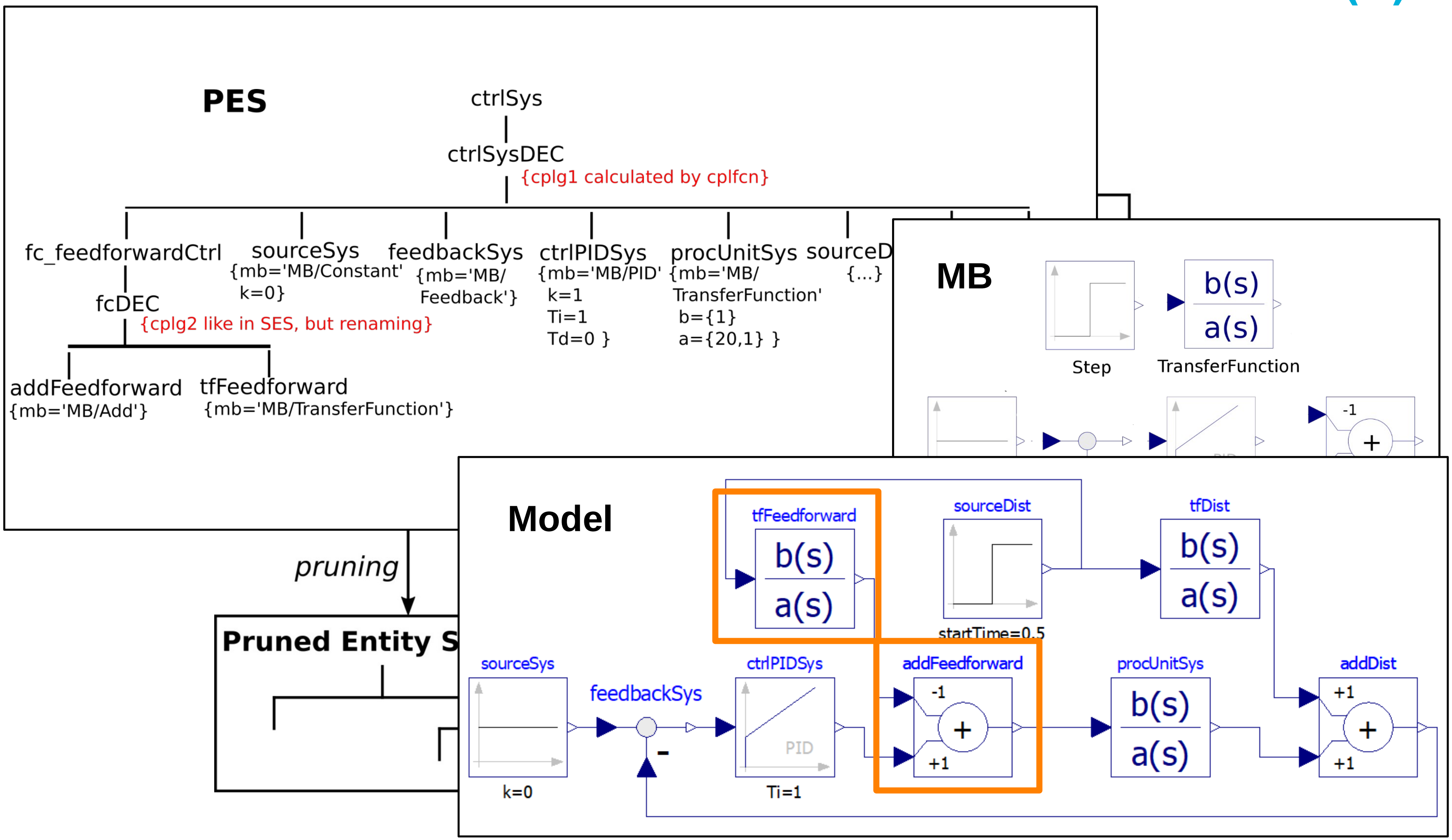
Model Selection and Generation of Variant #2 (3)



Model Selection and Generation of Variant #2 (4)



Model Selection and Generation of Variant #2 (4)



Python Toolset

- Available:
  `https://github.com/cea-wismar/SESMB_Inf_Python/`

- Tools

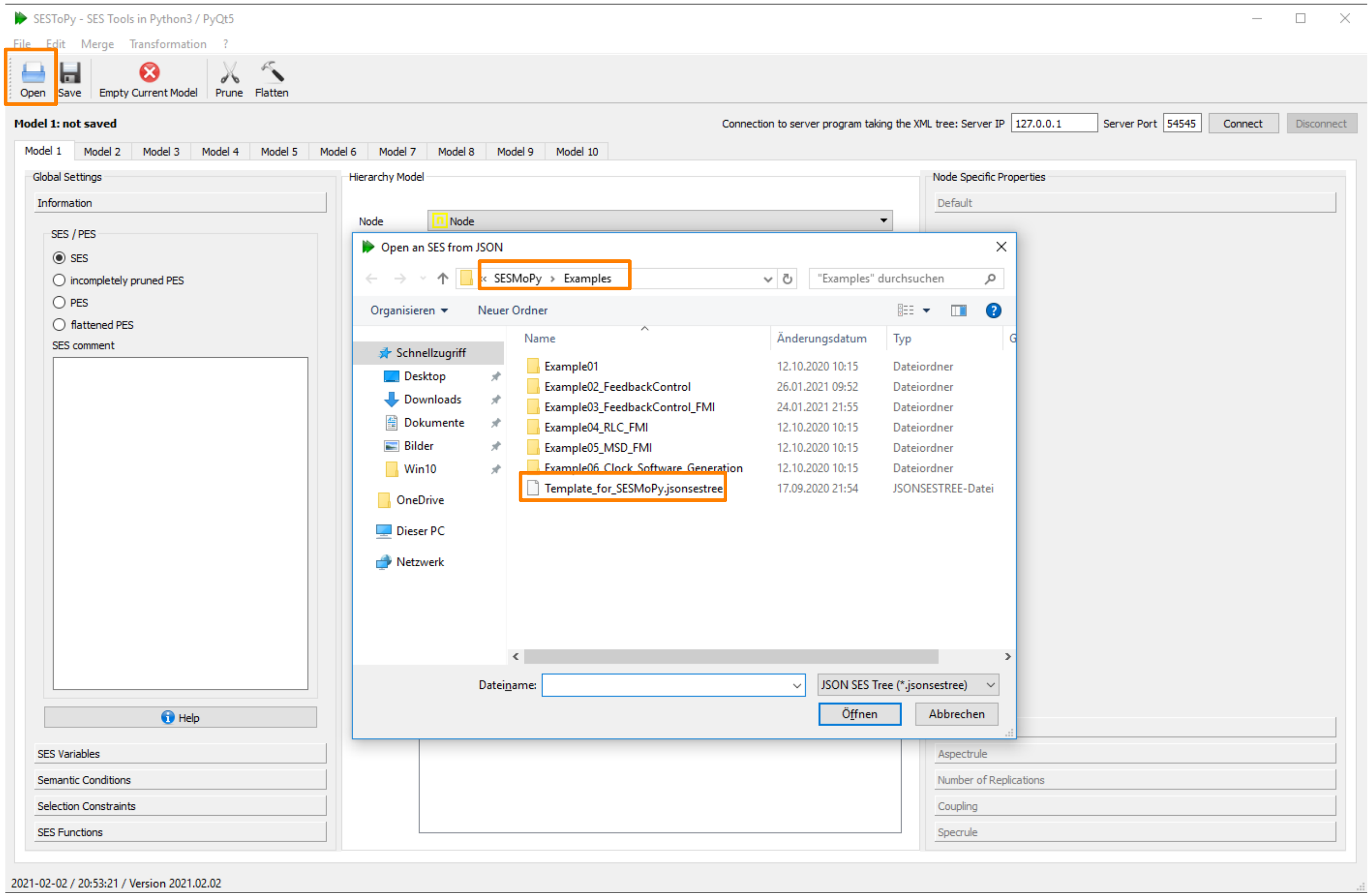
  - SESToPy → SES editor and IDE
  - SESViewEl → SES tree viewer
  - **SESMoPy** → Model builder → OpenModelica / Dymola / Simulink
  - SESEuPy
  - SESEcPy

Demonstration of SESMoPy (case study) → Screenshots on Next Slides

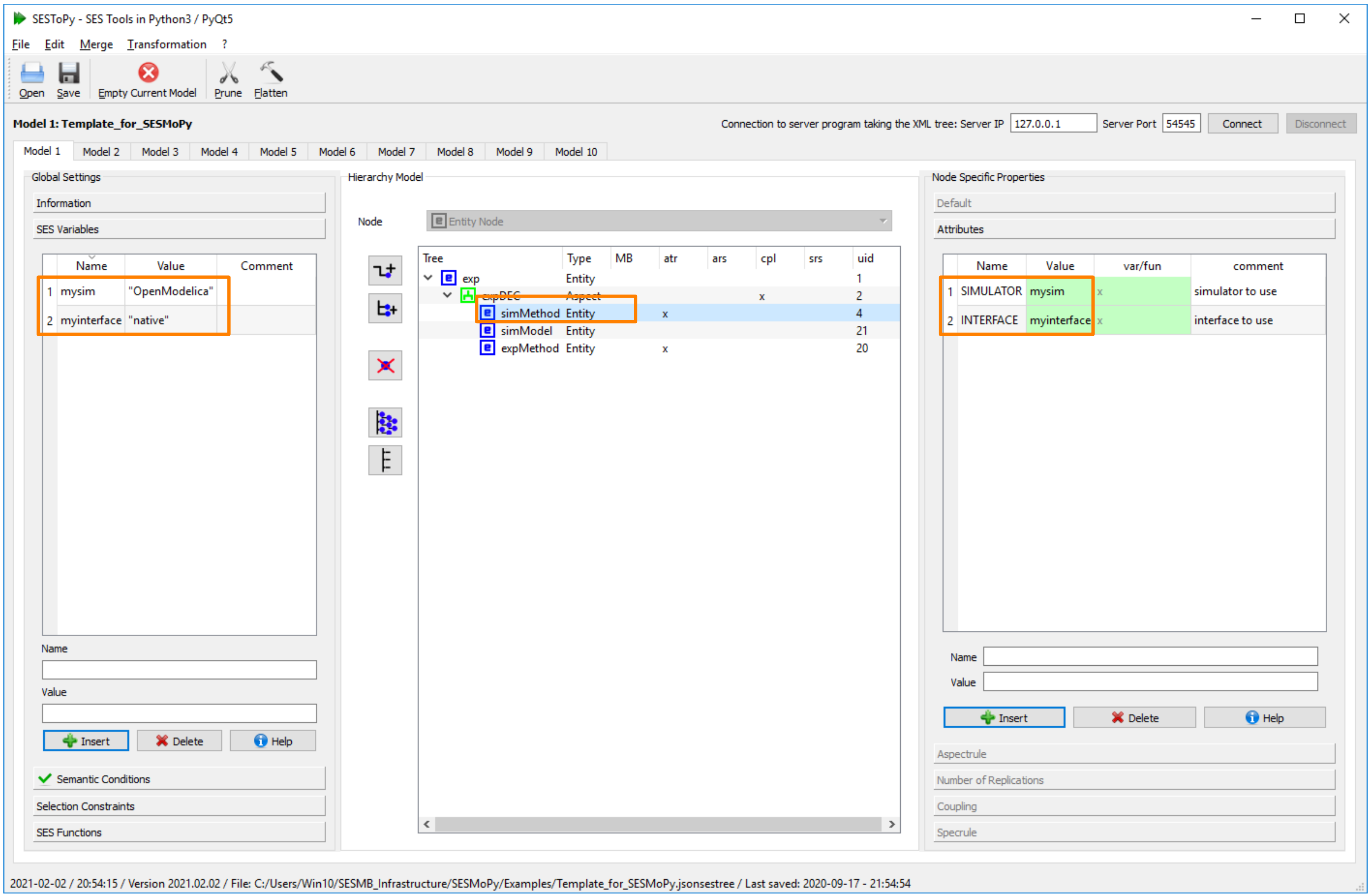
- Show provisional Experimental Frame from SESMoPy examples → `Template_for_SESMoPy`
- Show that different simulators can be set → here `OpenModelica`
- Show that two interfaces can be set → here `native`
- Merge Feedback SES from SESToPy examples to simModel → rename `simModel` to `ctrlSys` for merging.
- 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 → model not displayed (no annotation set) → click button `Text View`
- Execute simulation → set simulation time to 50 seconds
  → Signals of interest (setpoint, disturbance, controlled variable):
  `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

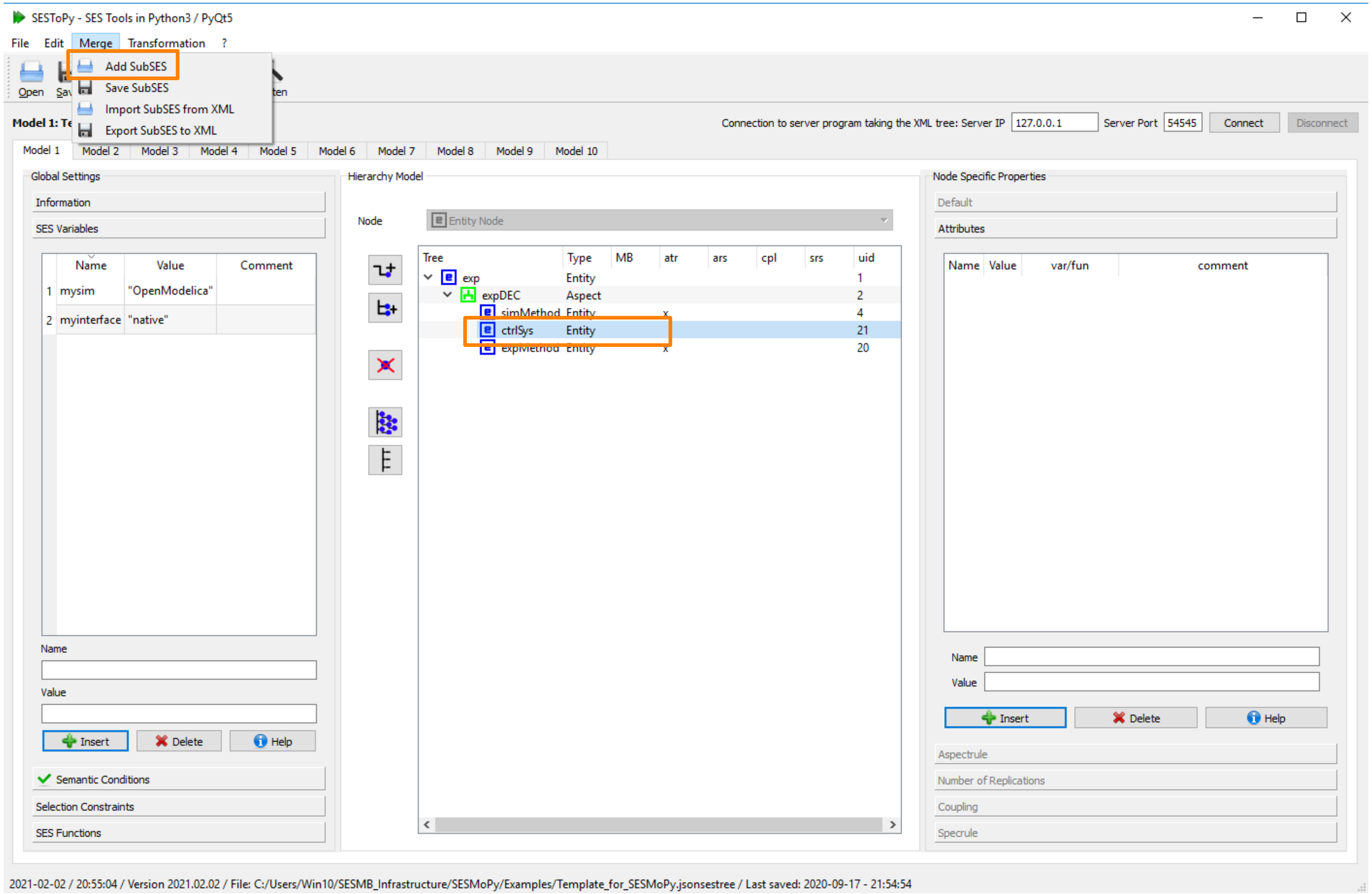
SESMoPy's Provisional Experimental Frame Show in SESToPy



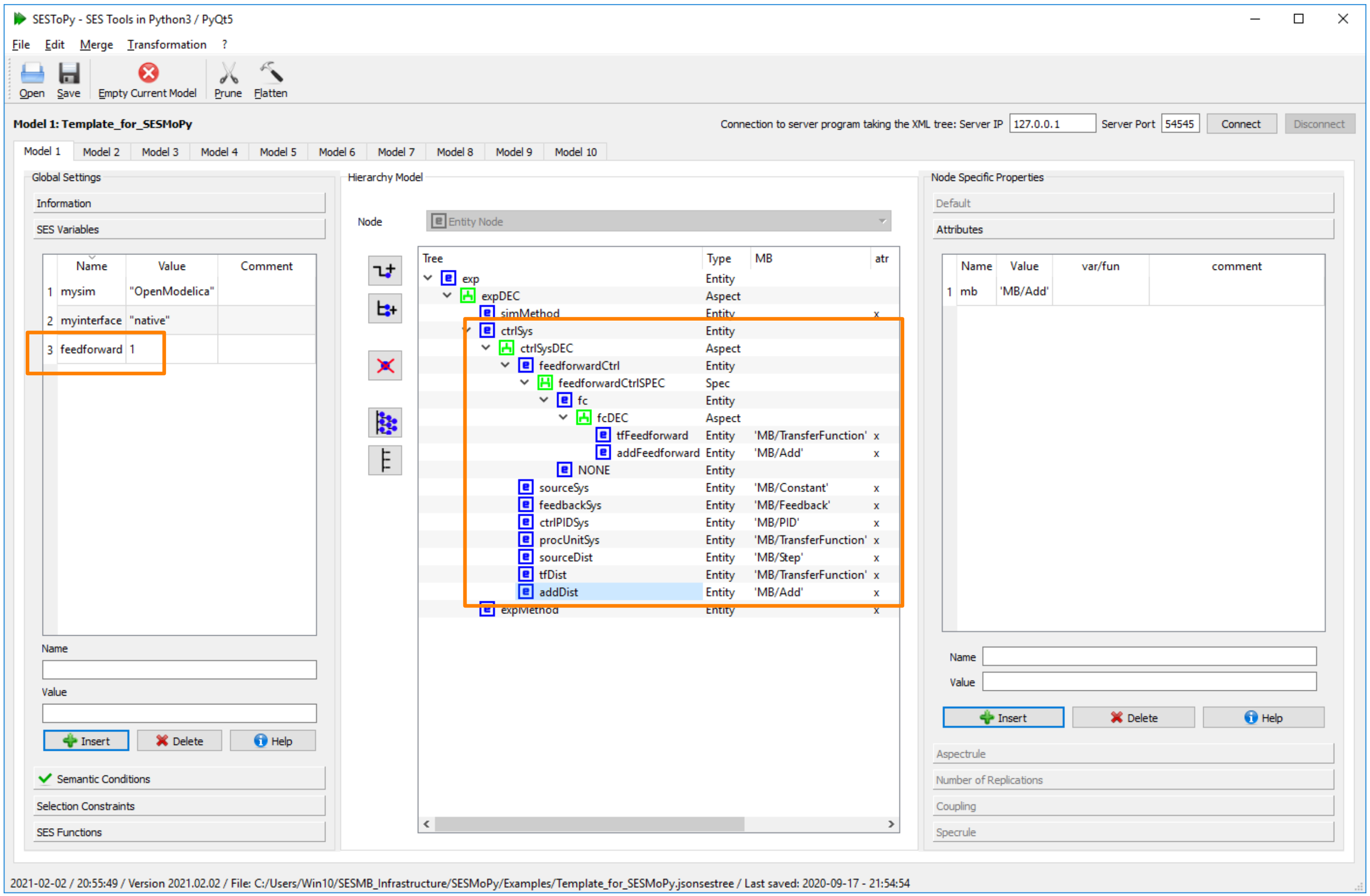
Different Simulators & Interfaces can be set



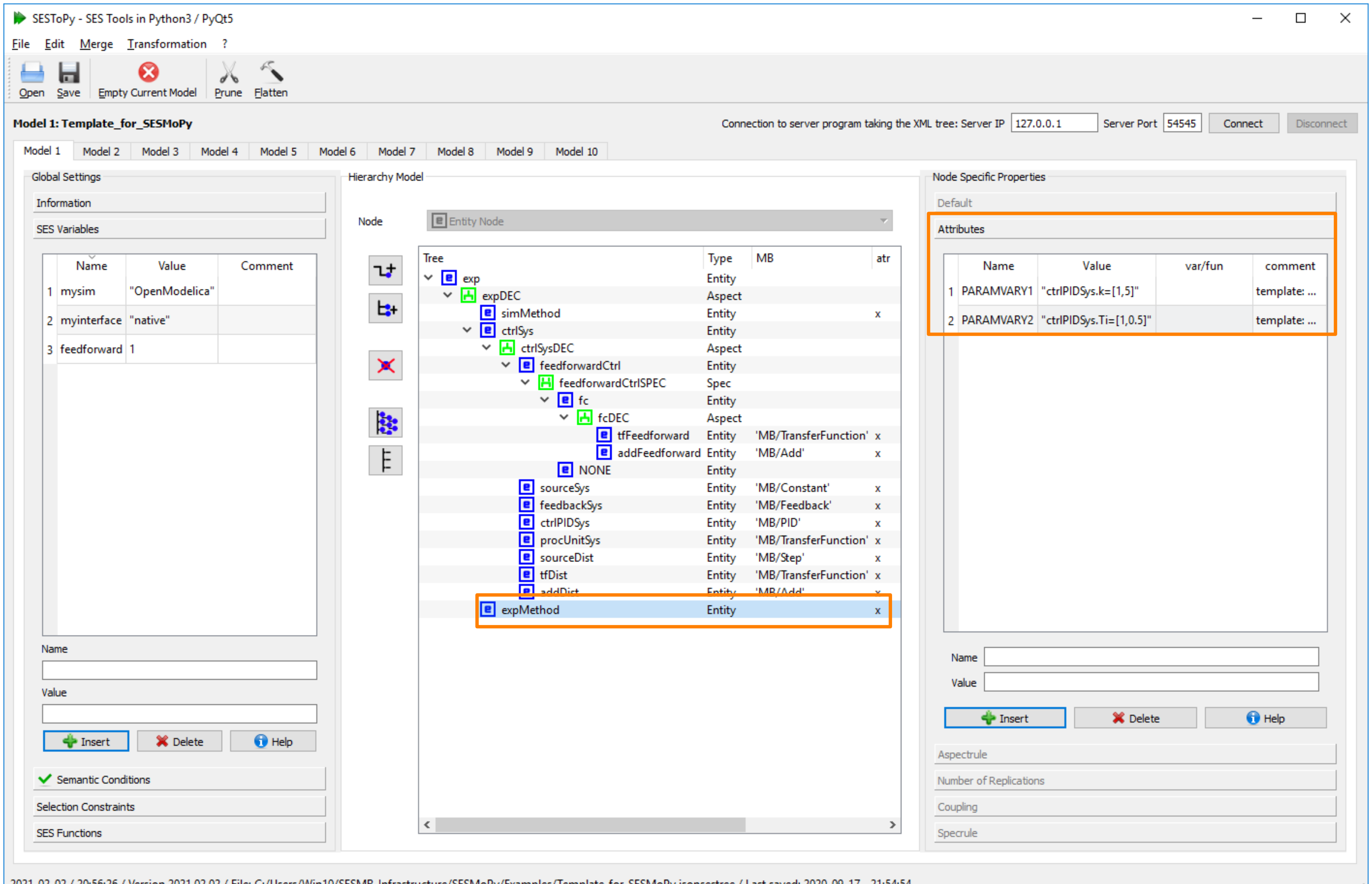
Merge Feedback SES to Provisional Experimental Frame



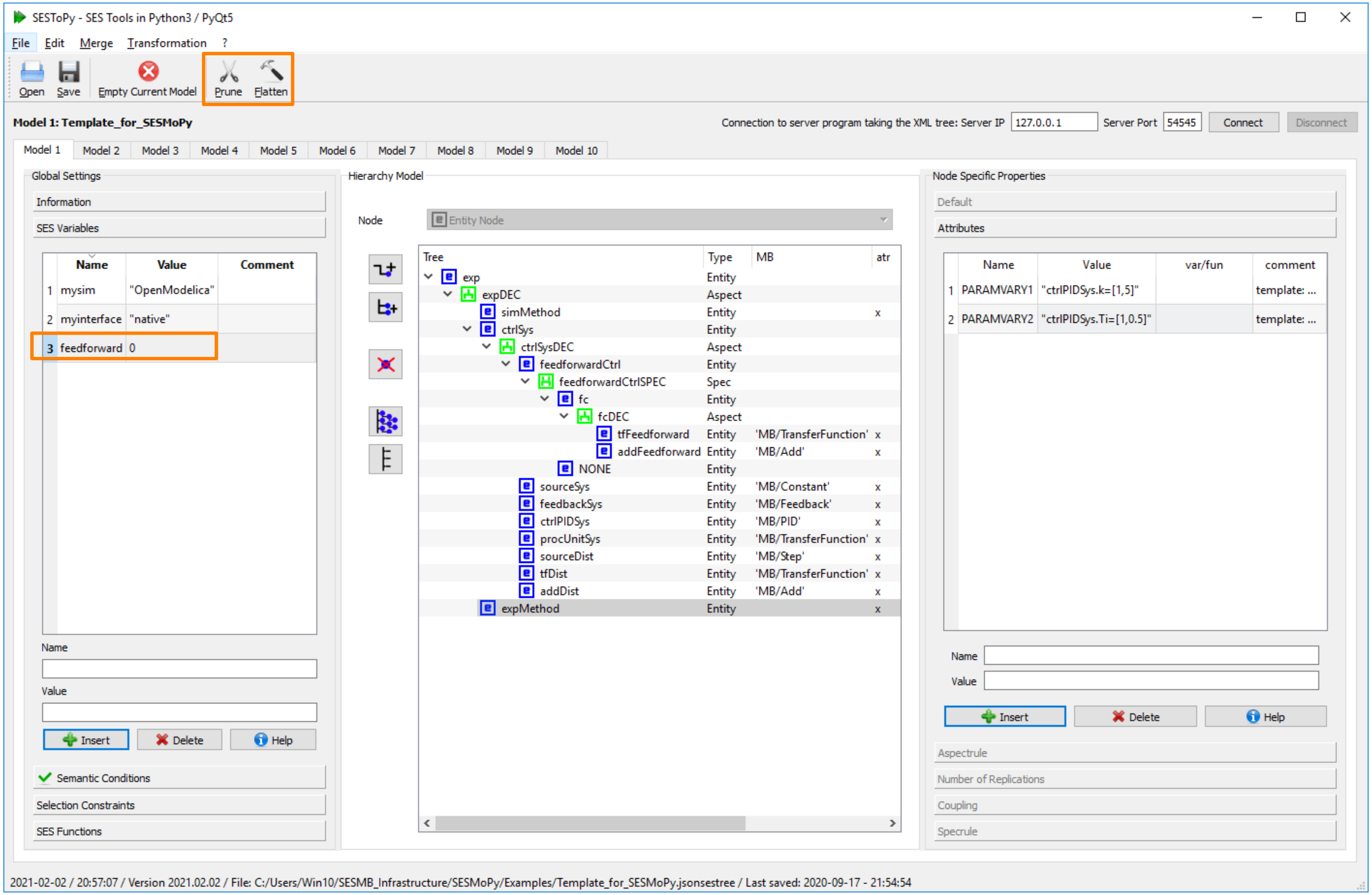
Merge Feedback SES to Provisional Experimental Frame (2)



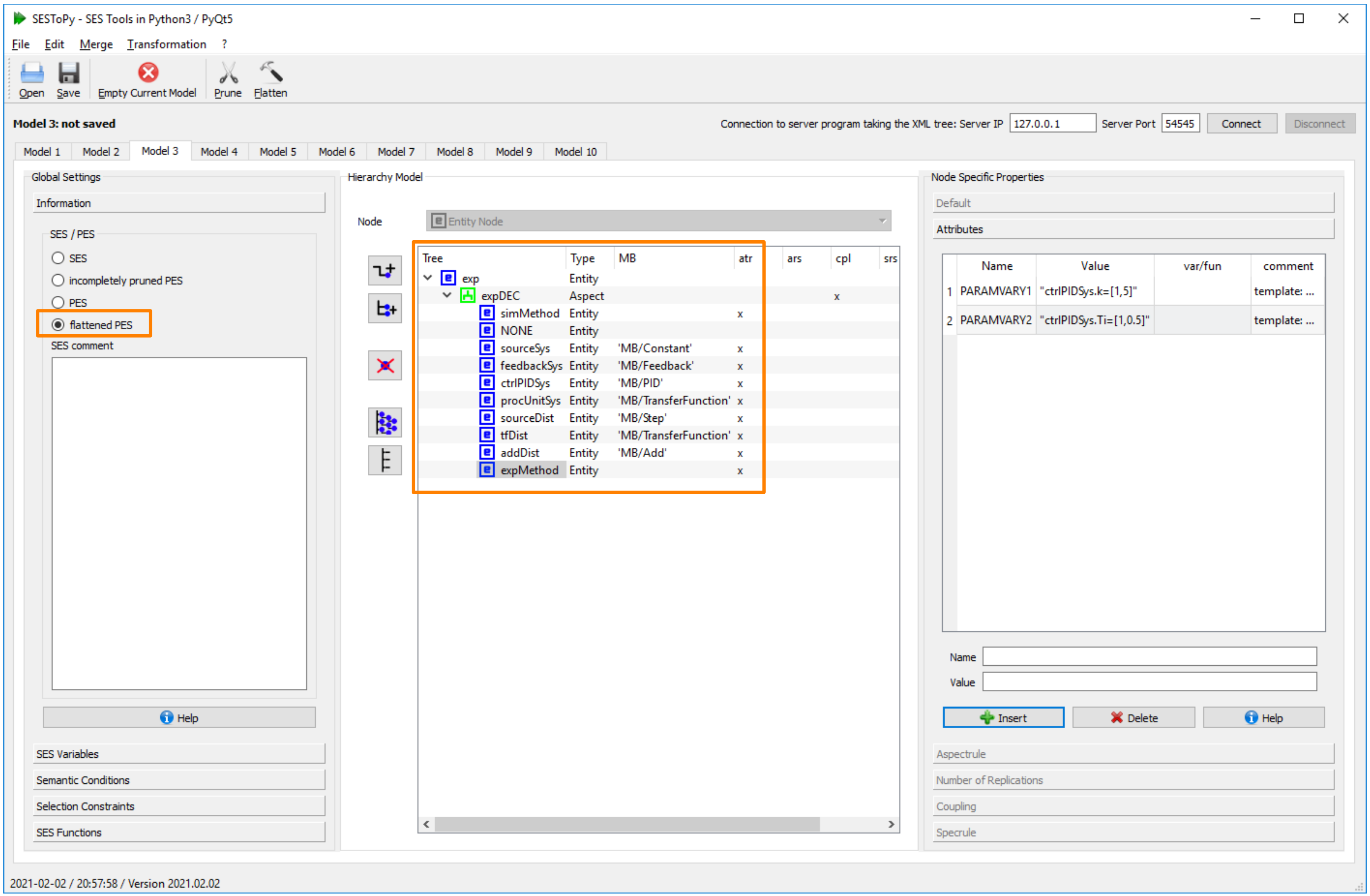
Different Parameters can be set



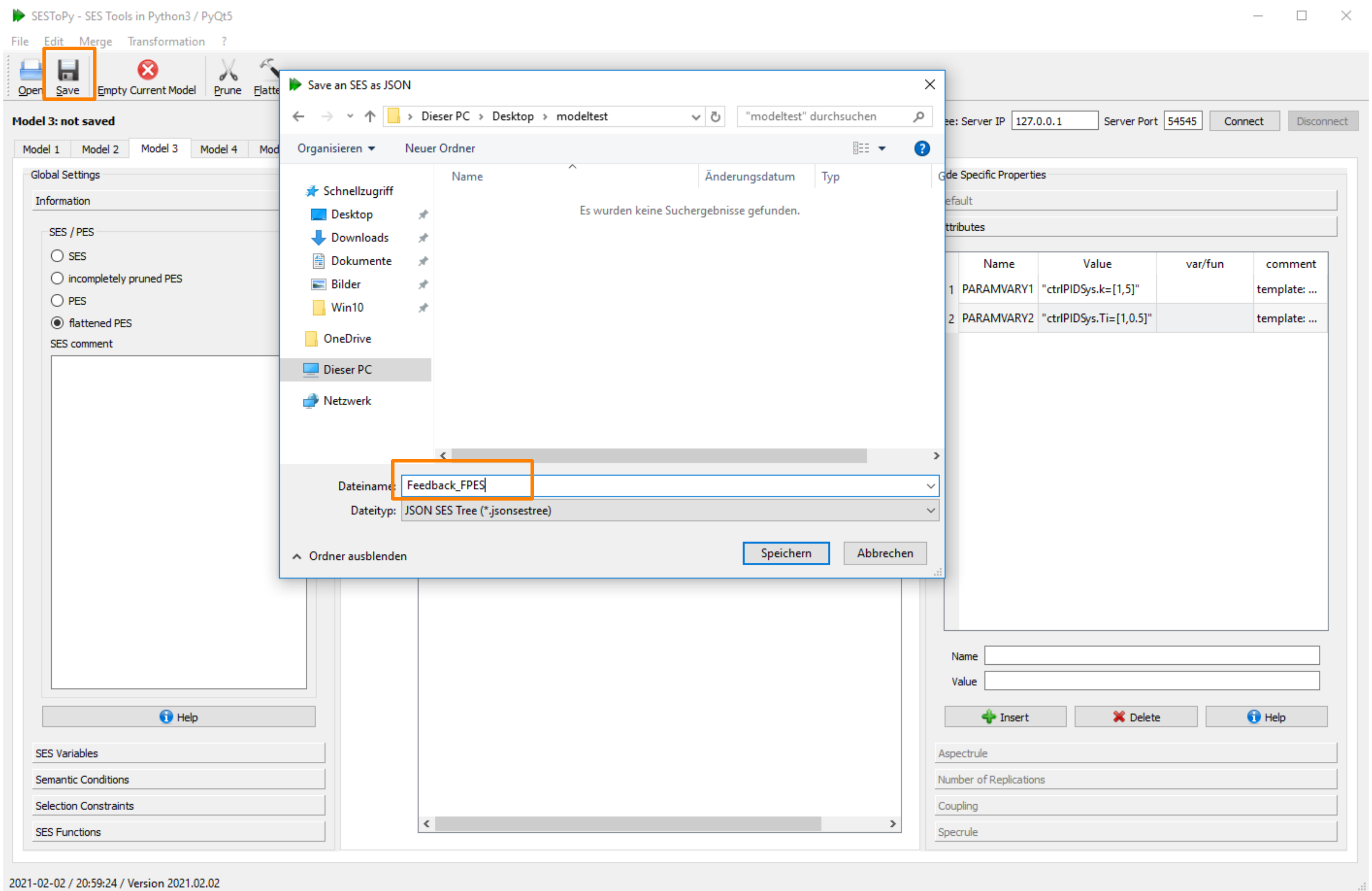
Prune & Flatten for feedforward=0



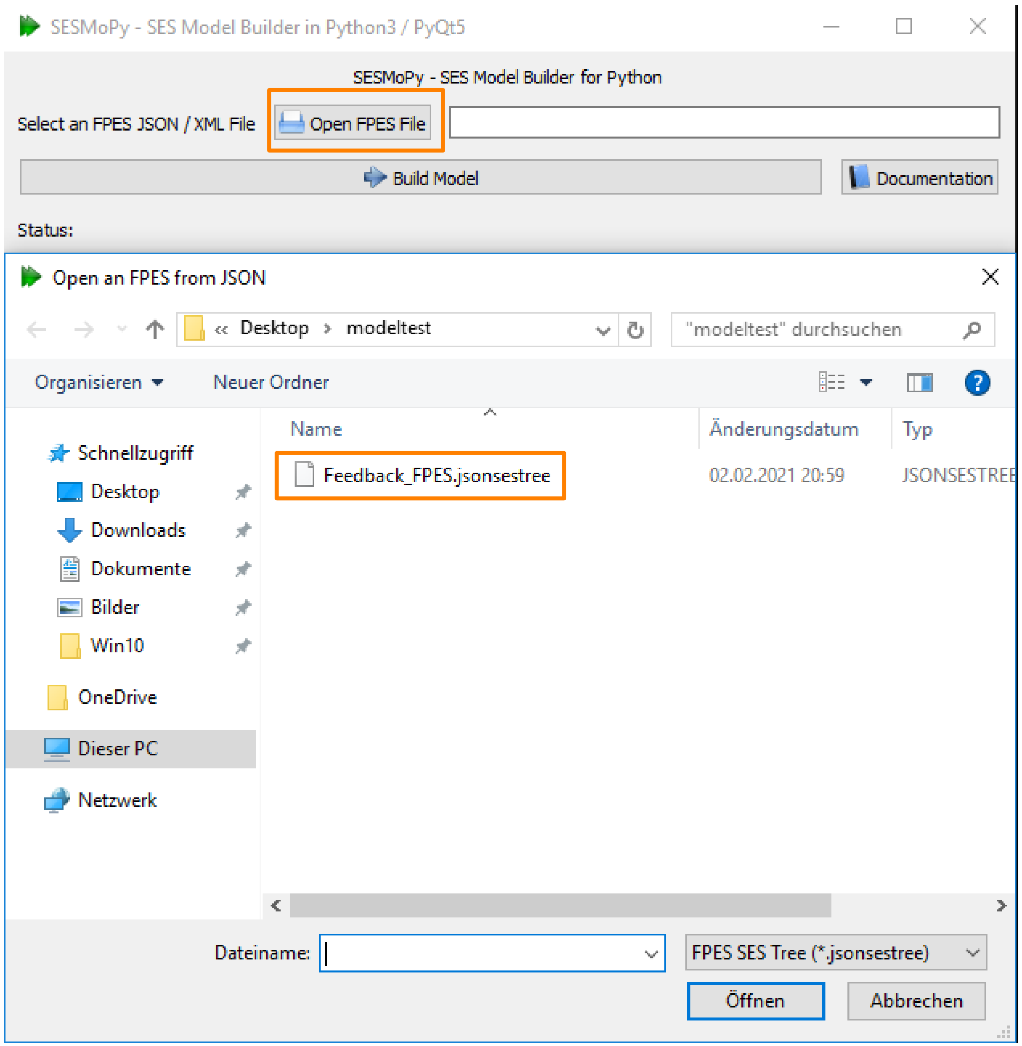
Flattened PES for feedforward=0



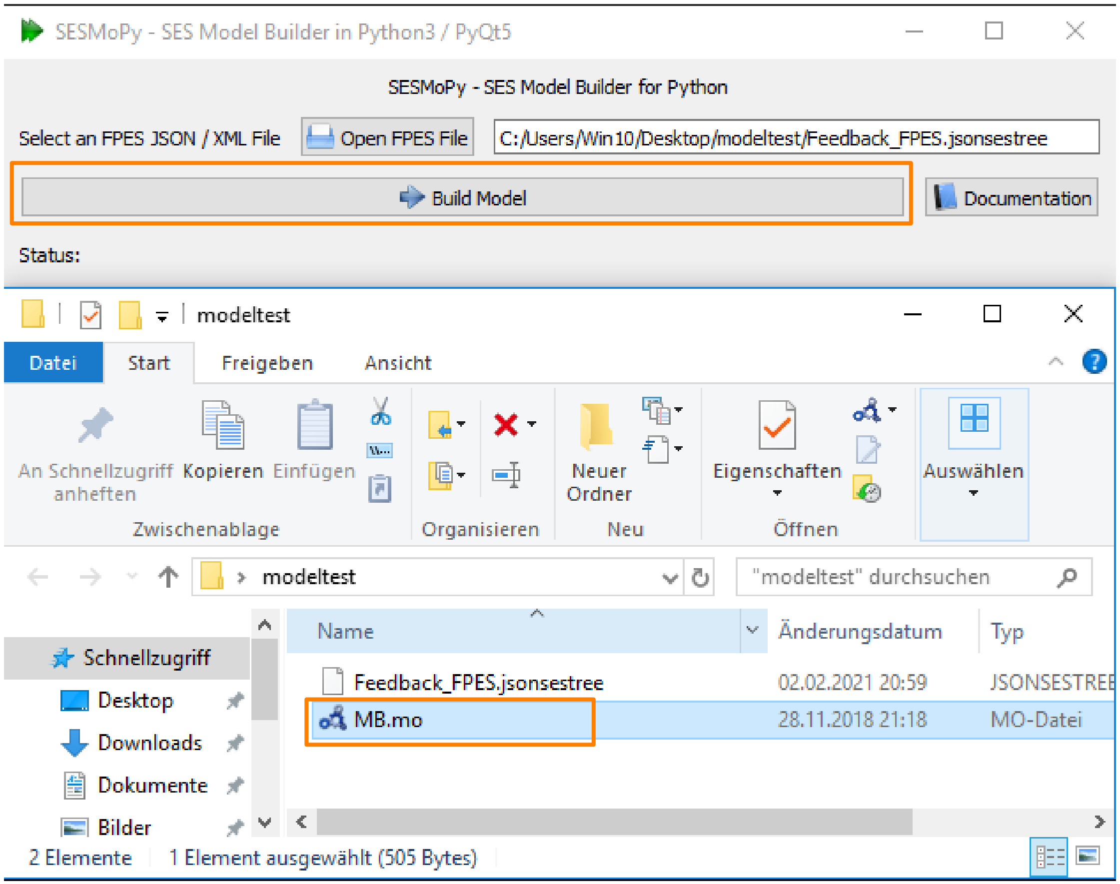
Save Flattened PES for feedforward=0



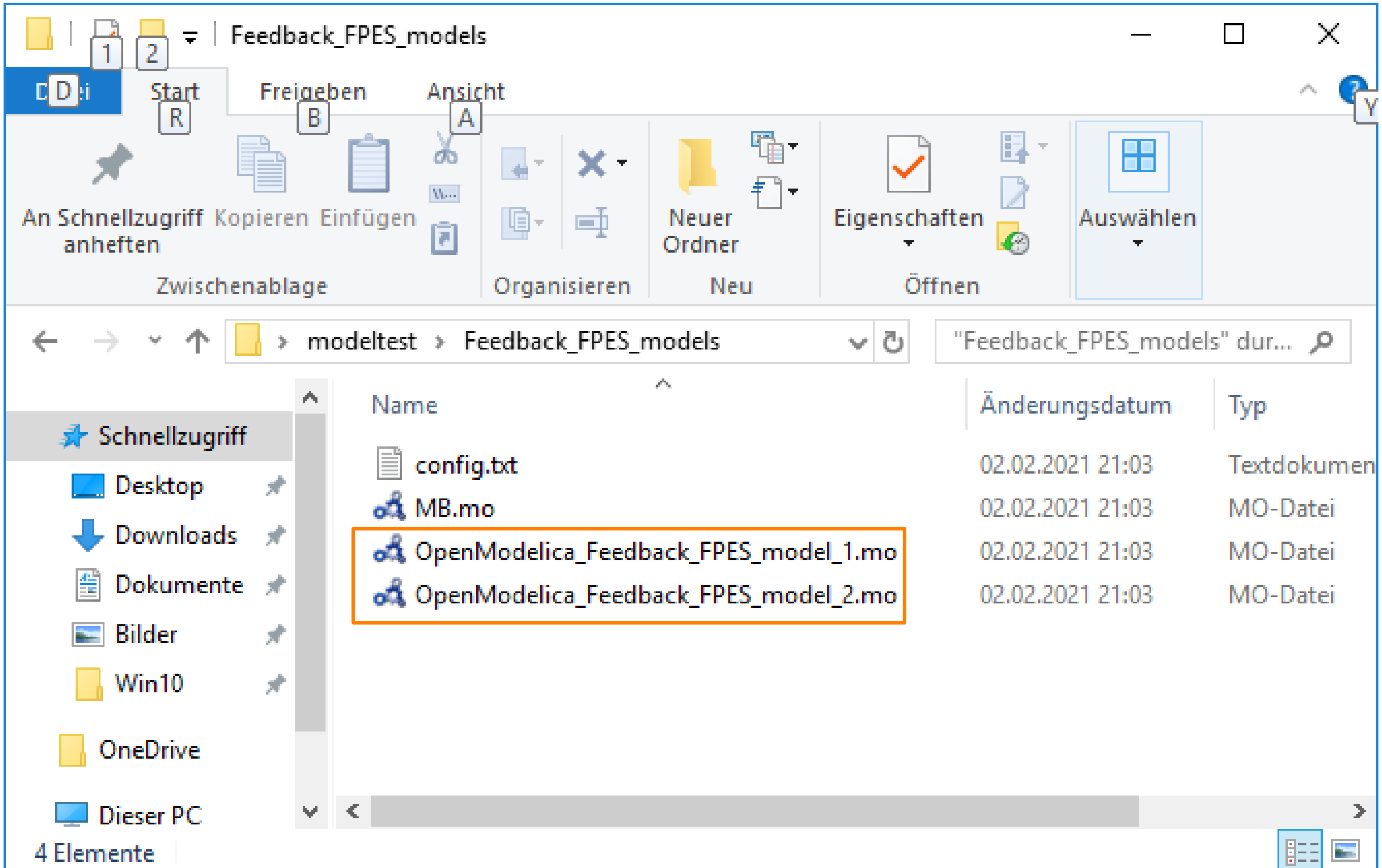
Open Flattened PES in SESMoPy



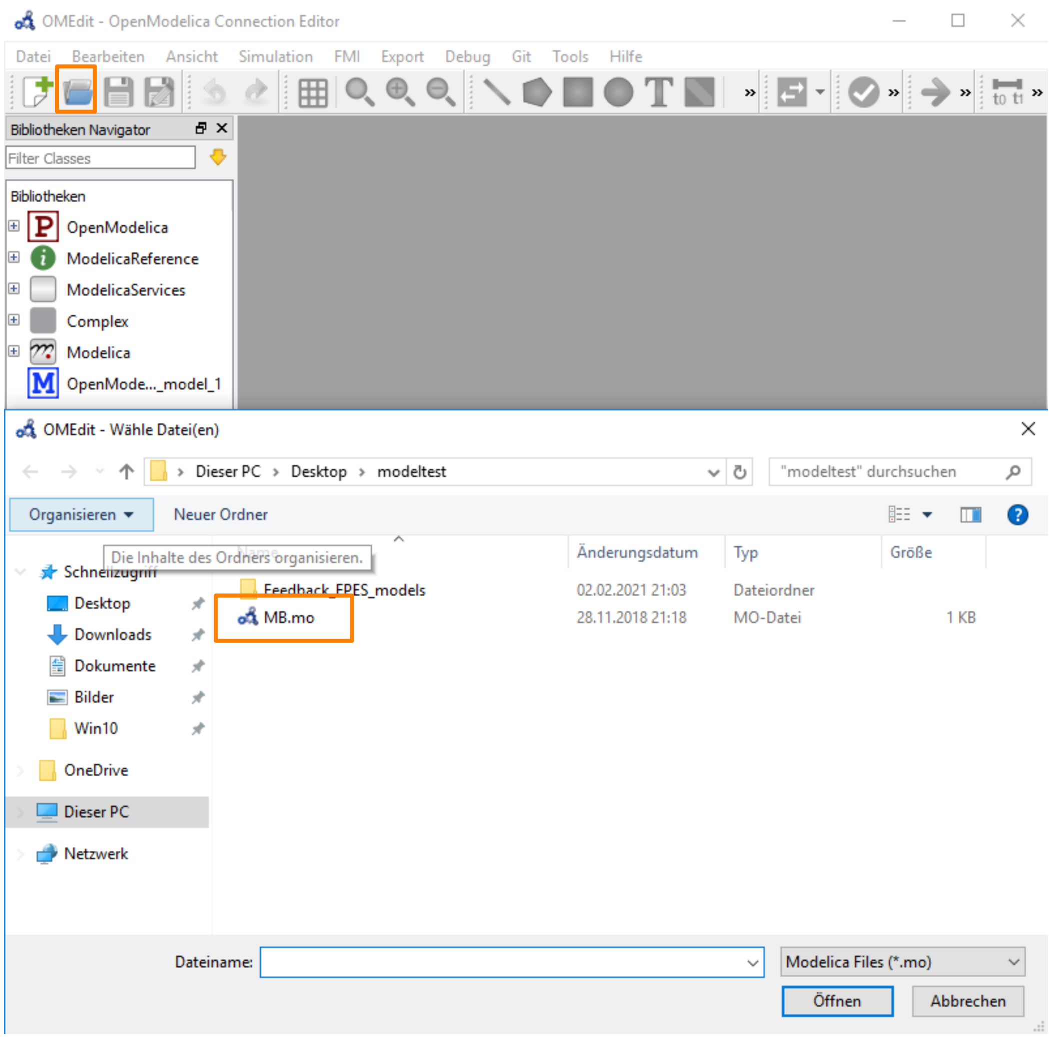
Add MB in Modelfolder & Build Model



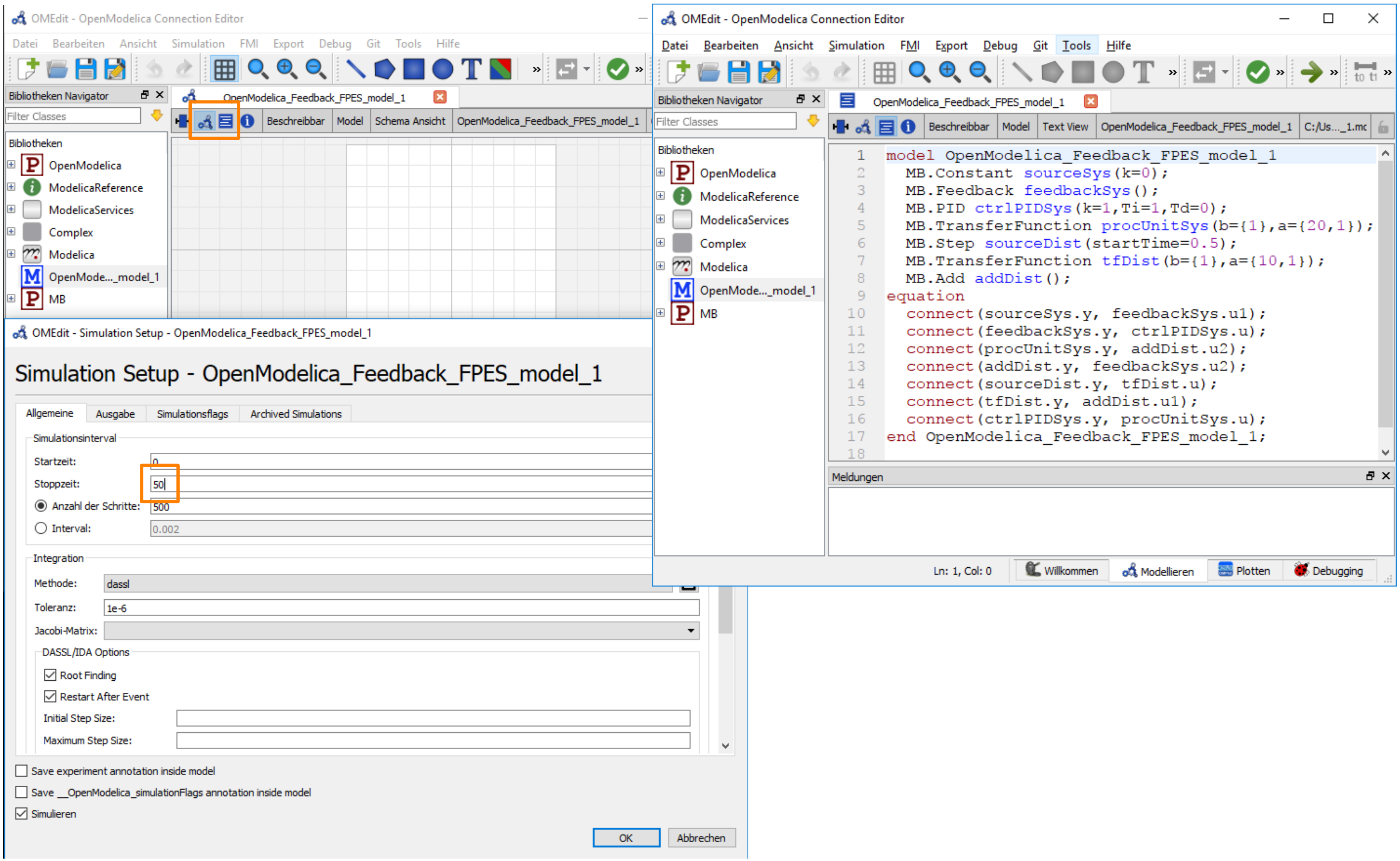
Created Models



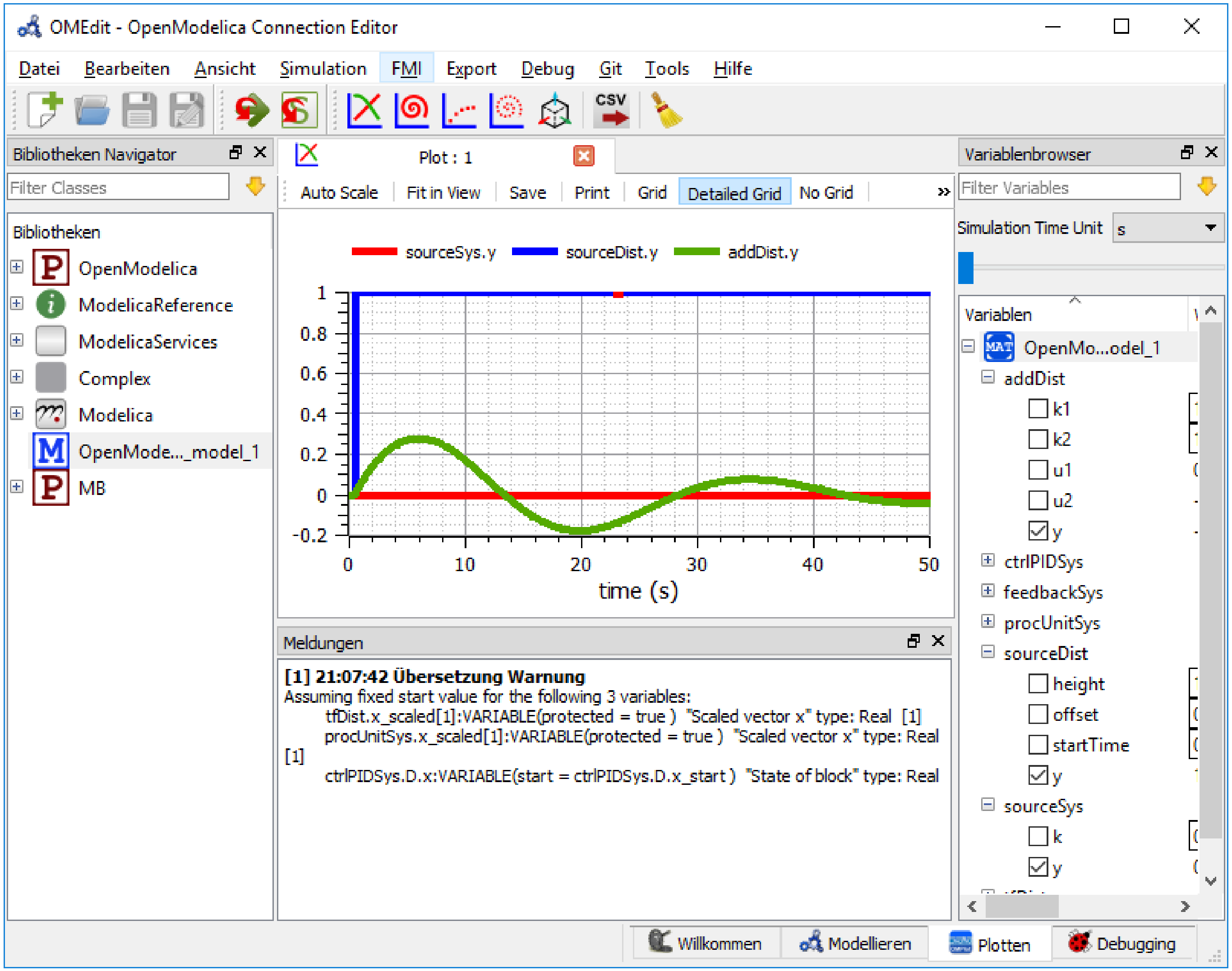
Open Created Models in OpenModelica & Load MB



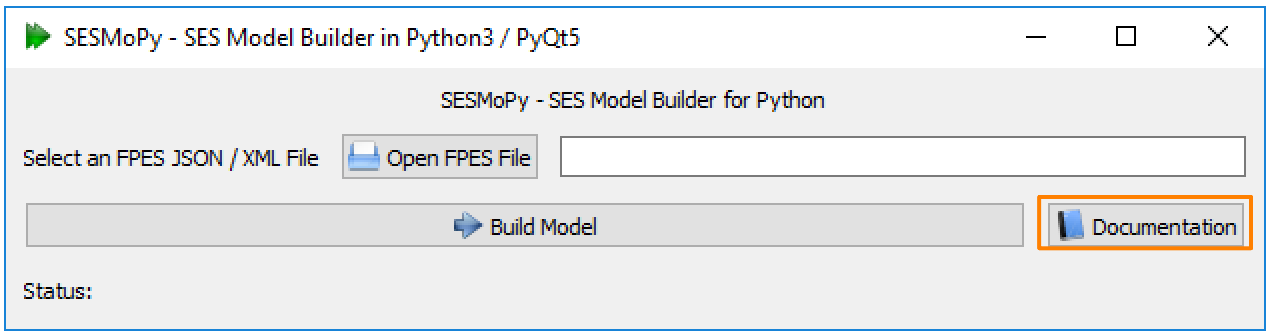
Model & Setup Simulation in OpenModelica



OpenModelica Simulation Results



SESMoPy Documentation



- See the documentation for more information

Outline

1. Case study
2. Implementation of the SES and an MB
3. Model selection and model generation
4. **Organization of a simulator-independent MB**
5. Full automation of simulation experiments
6. Summary

Model Building – Support Different Simulators Facts / Challenges

Model Building – Support Different Simulators Facts / Challenges

- System models should be executable with different simulators
  - Simulators are domain specific
  - Verify simulator correctness

Model Building – Support Different Simulators Facts / Challenges

- System models should be executable with different simulators
  - Simulators are domain specific
  - Verify simulator correctness

- **SES is independent of simulator**

Model Building – Support Different Simulators Facts / Challenges

- System models should be executable with different simulators
  - Simulators are domain specific
  - Verify simulator correctness
- **SES is independent of simulator**
- **Native model building using a simulator dependent MB**
  - Needs **one** MB for **each** simulator (error prone and costly to maintain)
  - Needs **specific model builders**, because simulators are different (syntax and semantics such as port names, block parameters, ...)

Model Building – Support Different Simulators Facts / Challenges

- System models should be executable with different simulators
  - Simulators are domain specific
  - Verify simulator correctness
- **SES is independent of simulator**
- **Native model building using a simulator dependent MB**
  - Needs **one** MB for **each** simulator (error prone and costly to maintain)
  - Needs **specific model builders**, because simulators are different (syntax and semantics such as port names, block parameters, ...)
- **Goal: One** (simple) MB and model builder for **all** simulators

Functional Mock-up Interface (FMI) [1,2]

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

[2]Blochwitz 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)

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

[2]Blochwitz 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)
  - Reuse of components
    - (i) For **model exchange**
    - (ii) For co-simulation

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

[2]Blochwitz 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)
  - Reuse of components
    - (i) For **model exchange**
    - (ii) For co-simulation
  - Based on C code or binaries

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

[2]Blochwitz 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)
  - Reuse of components
    - (i) For **model exchange**
    - (ii) For co-simulation
  - Based on C code or binaries
  - Many simulators support FMI

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

[2]Blochwitz 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)
  - Reuse of components
    - (i) For **model exchange**
    - (ii) For co-simulation
  - Based on C code or binaries
  - Many simulators support FMI
  - Still problems for discrete event models

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

[2] Blochwitz 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)

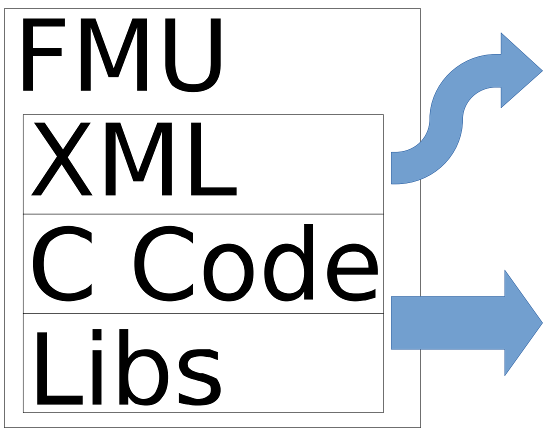
Component implementing FMI = Functional Mock-up Unit (FMU)

a zipped file with fileextension .fmu

Functional Mock-up Unit (FMU)

Component implementing FMI = Functional Mock-up Unit (FMU)

a zipped file with fileextension .fmu



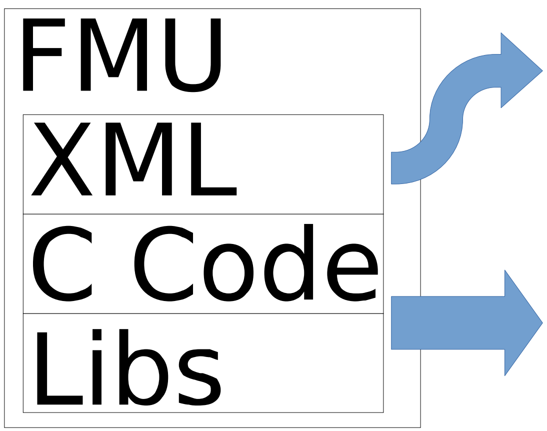
Description of interface data (variables, ...)

Functionality (dynamics) in C source or binary form for different platforms

Functional Mock-up Unit (FMU)

Component implementing FMI = Functional Mock-up Unit (FMU)

a zipped file with fileextension .fmu



Description of interface data (variables, ...)

Functionality (dynamics) in C source or binary form for different platforms

Model Exchange

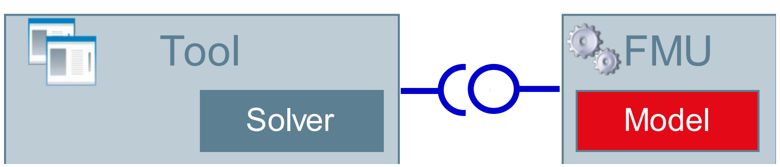


Figure taken from the FMI presentation on the website www.fmi-standard.org.

Model Building – Support Different Simulators Using FMI

Model Building – Support Different Simulators Using FMI

- **Idea: Using FMI for model exchange**

Model Building – Support Different Simulators Using FMI

- **Idea: Using FMI for model exchange**
  - Export basic models as FMUs from any simulator to create simulator-independent MB

Model Building – Support Different Simulators Using FMI

- **Idea: Using FMI for model exchange**
  - Export basic models as FMUs from any simulator to create simulator-independent MB
  - Model generation (build):
    - Import and configure FMUs from MB and create couplings in target simulator

Model Building – Support Different Simulators Using FMI

- **Idea: Using FMI for model exchange**
  - Export basic models as FMUs from any simulator to create simulator-independent MB
  - Model generation (build):
    - Import and configure FMUs from MB and create couplings in target simulator
  - **Problems:**
    - Some simulators do not support configuration of FMUs (basic models) and creation of couplings
    - FMU import is time consuming → slow model building

Model Building – Support Different Simulators Using FMI

- **Idea: Using FMI for model exchange**
  - Export basic models as FMUs from any simulator to create simulator-independent MB
  - Model generation (build):
    - Import and configure FMUs from MB and create couplings in target simulator
  - **Problems:**
    - Some simulators do not support configuration of FMUs (basic models) and creation of couplings
    - FMU import is time consuming → slow model building
- **Workaround: Using FMI for model exchange and OpenModelica**

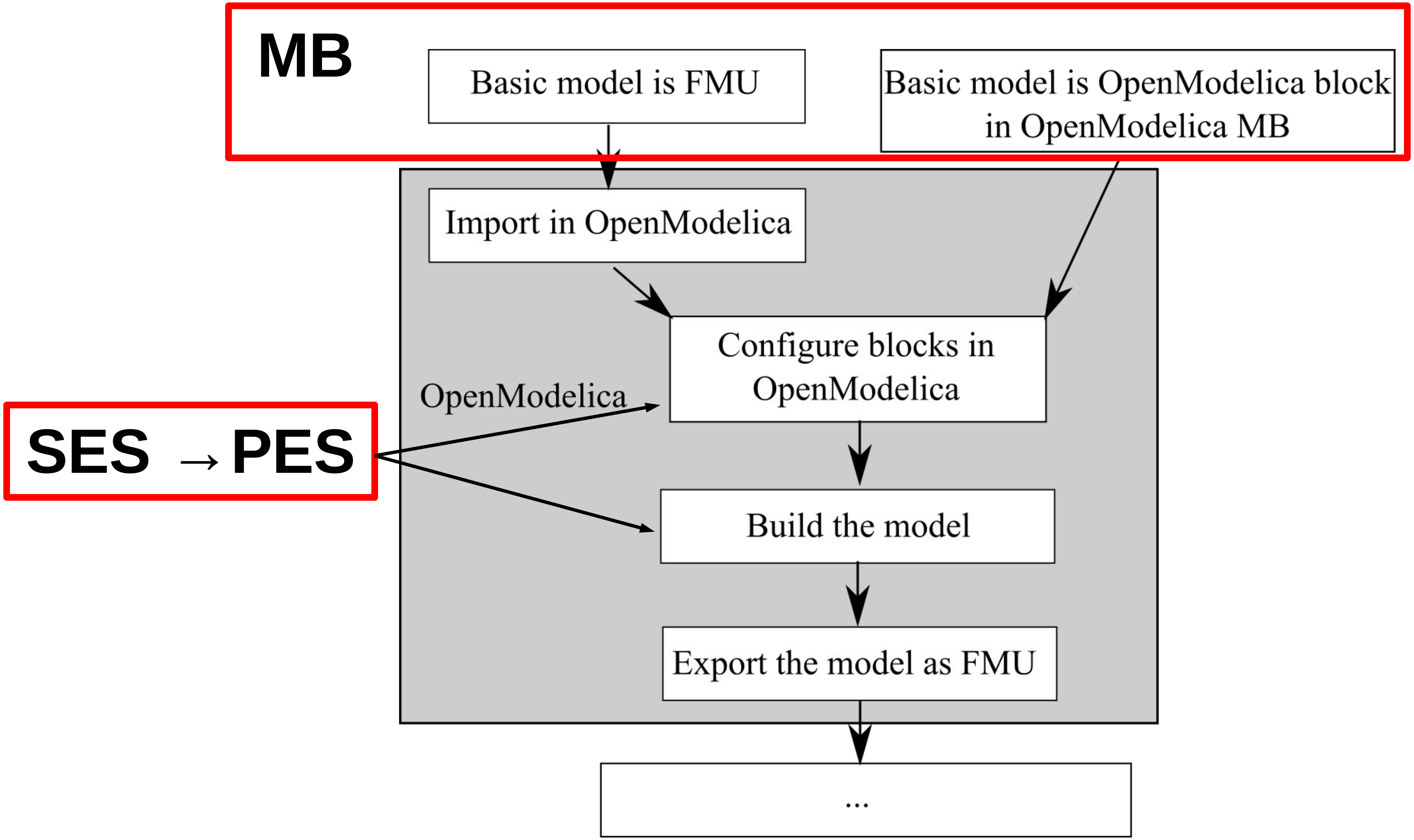
Model Building – Support Different Simulators Using FMI

- **Idea: Using FMI for model exchange**
  - Export basic models as FMUs from any simulator to create simulator-independent MB
  - Model generation (build):
    - Import and configure FMUs from MB and create couplings in target simulator
  - **Problems:**
    - Some simulators do not support configuration of FMUs (basic models) and creation of couplings
    - FMU import is time consuming → slow model building
- **Workaround: Using FMI for model exchange and OpenModelica**
  - Export basic models as FMUs from any simulator to create simulator-independent MB

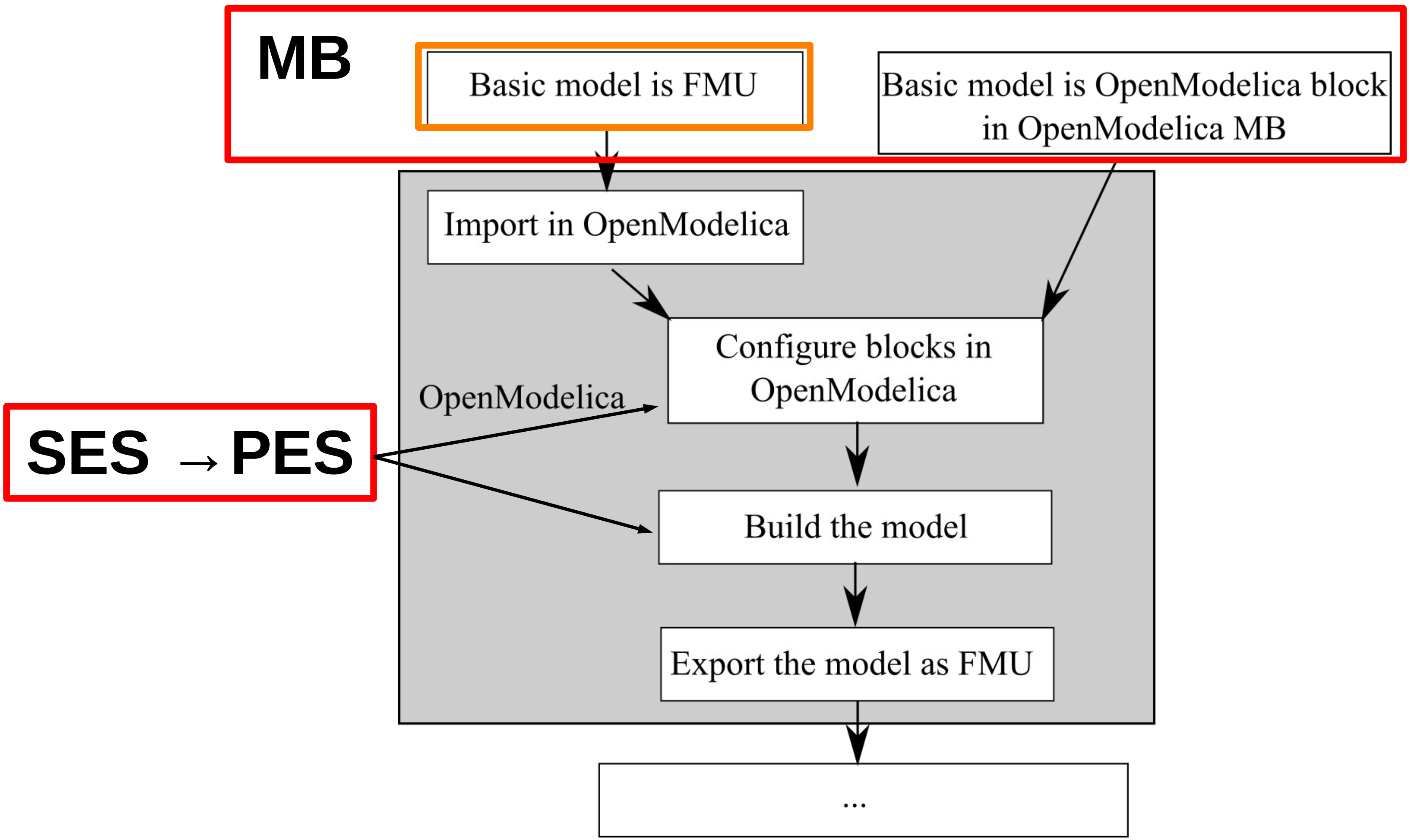
Model Building – Support Different Simulators Using FMI

- **Idea: Using FMI for model exchange**
  - Export basic models as FMUs from any simulator to create simulator-independent MB
  - Model generation (build):
    - Import and configure FMUs from MB and create couplings in target simulator
  - **Problems:**
    - Some simulators do not support configuration of FMUs (basic models) and creation of couplings
    - FMU import is time consuming → slow model building
- **Workaround: Using FMI for model exchange and OpenModelica**
  - Export basic models as FMUs from any simulator to create simulator-independent MB
  - Model generation (build):
    - Import and configure FMUs from MB and create couplings in OpenModelica
    - Export the configured model as one FMU
    - Import model FMU in the target simulator

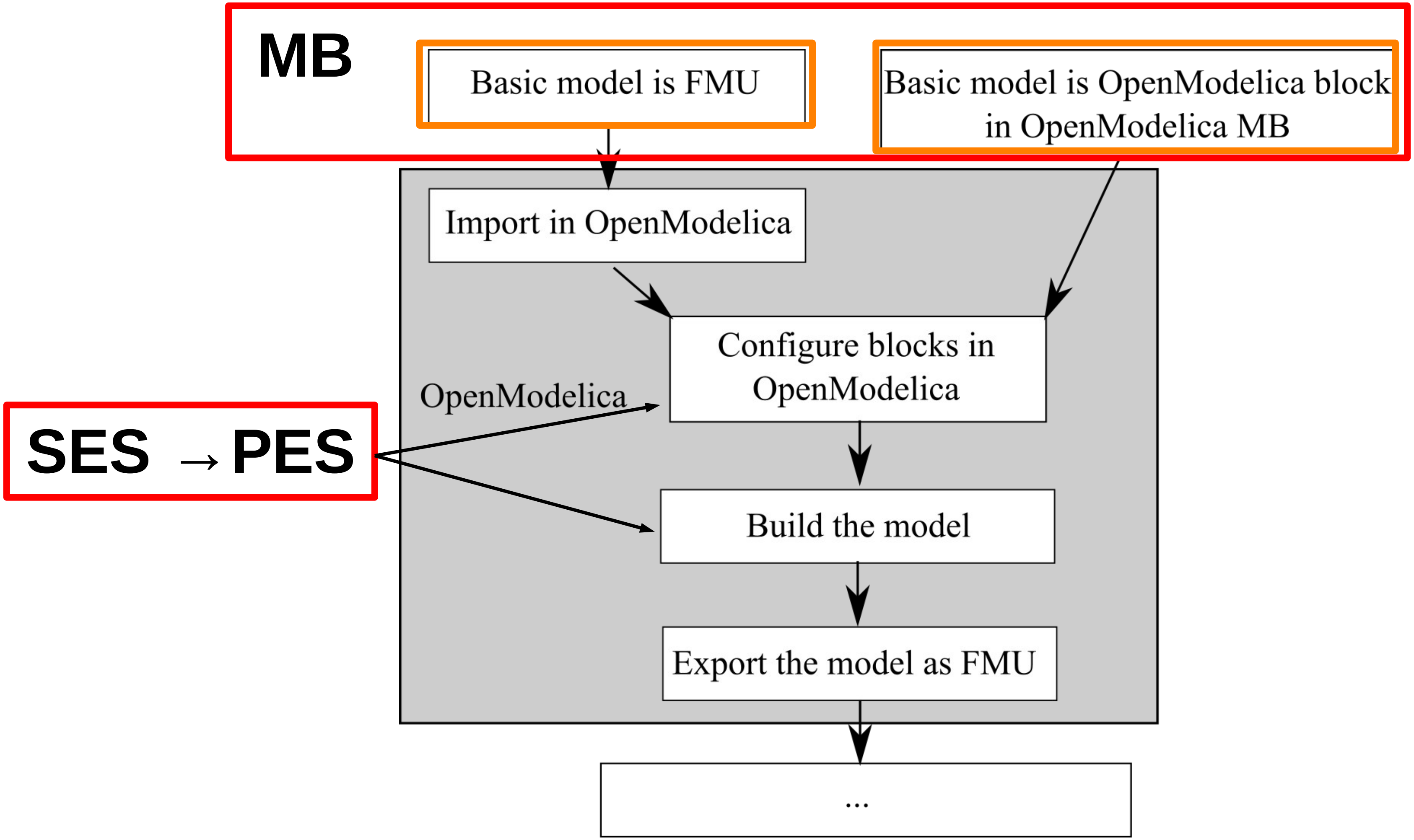
Model Building Using FMI



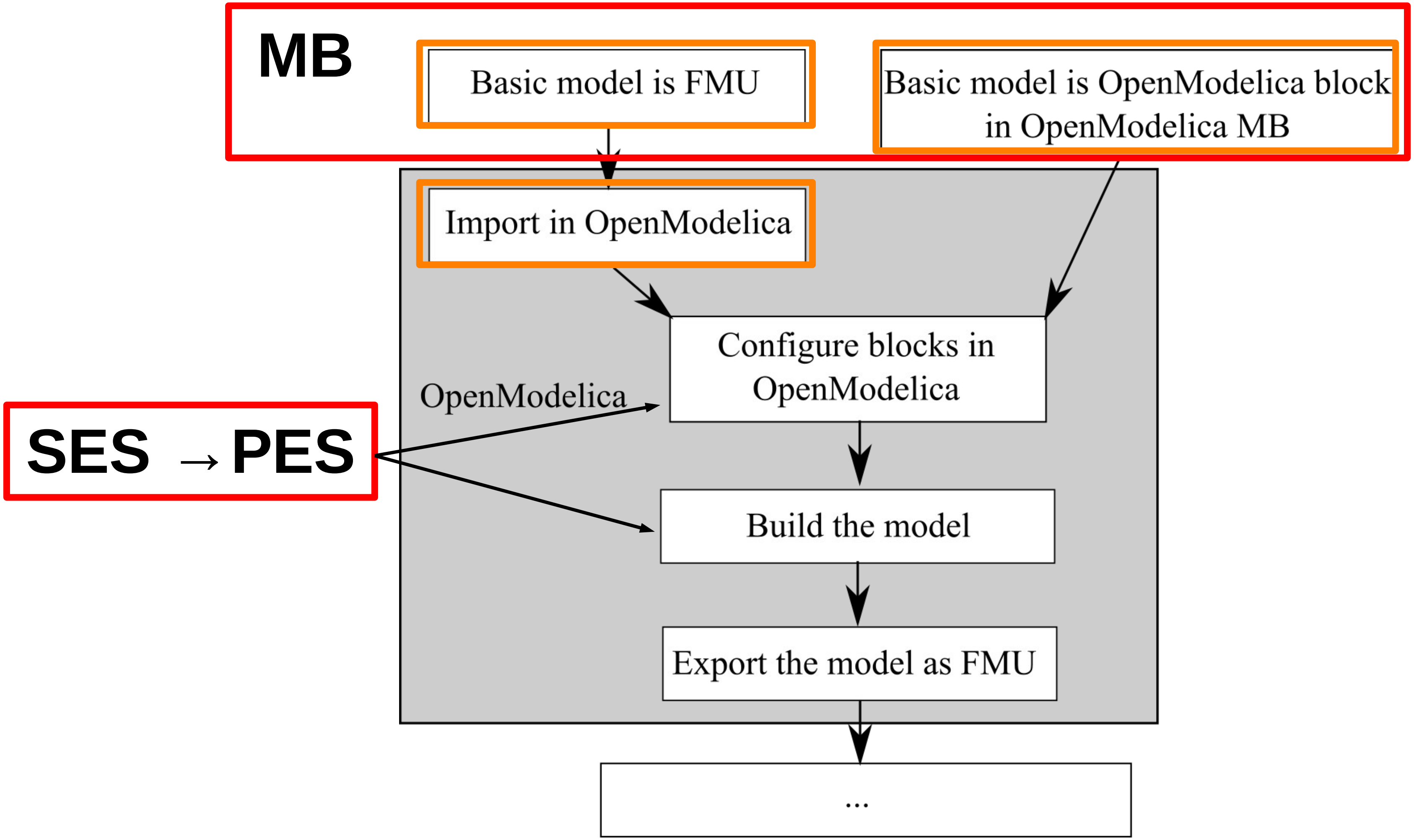
Model Building Using FMI



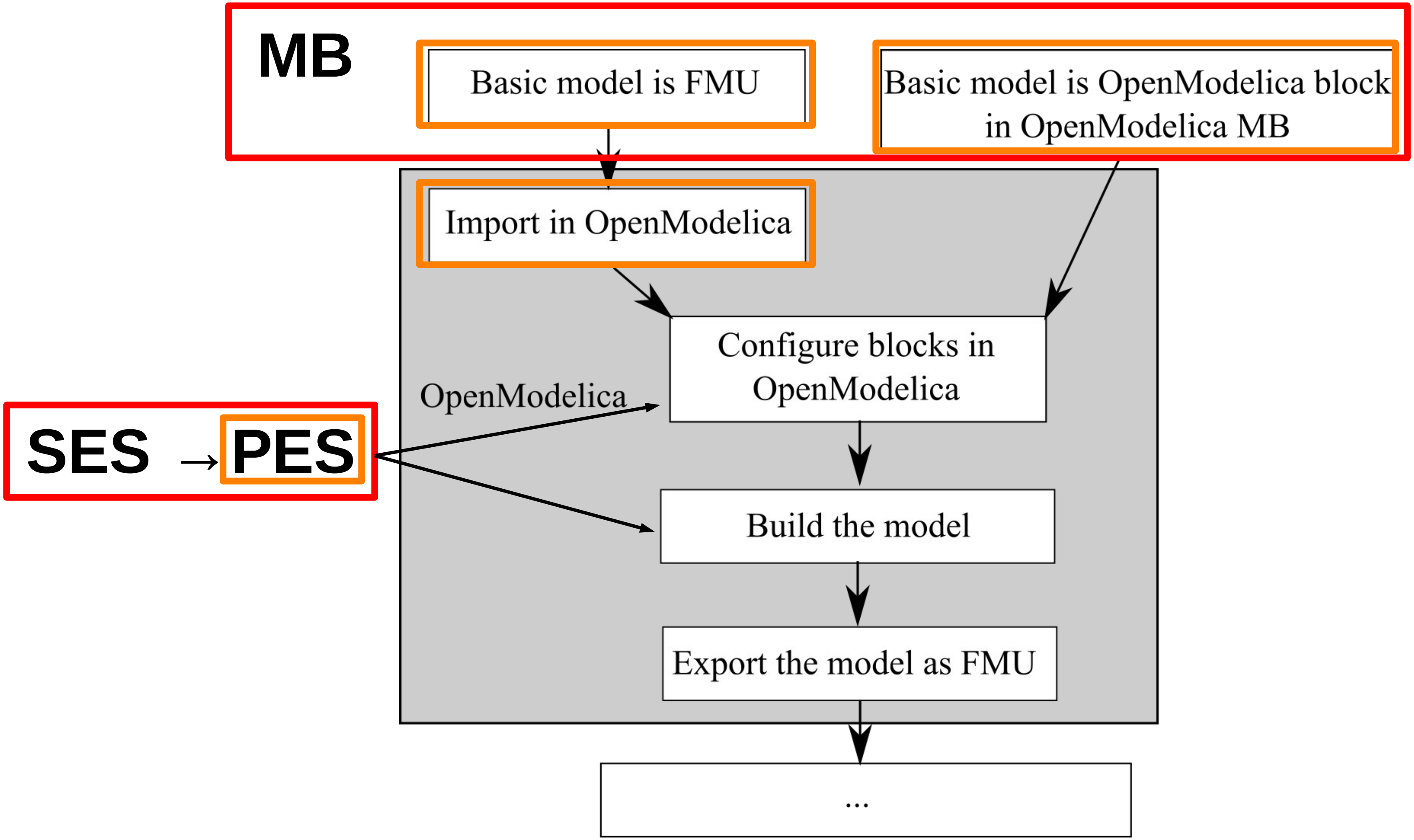
Model Building Using FMI



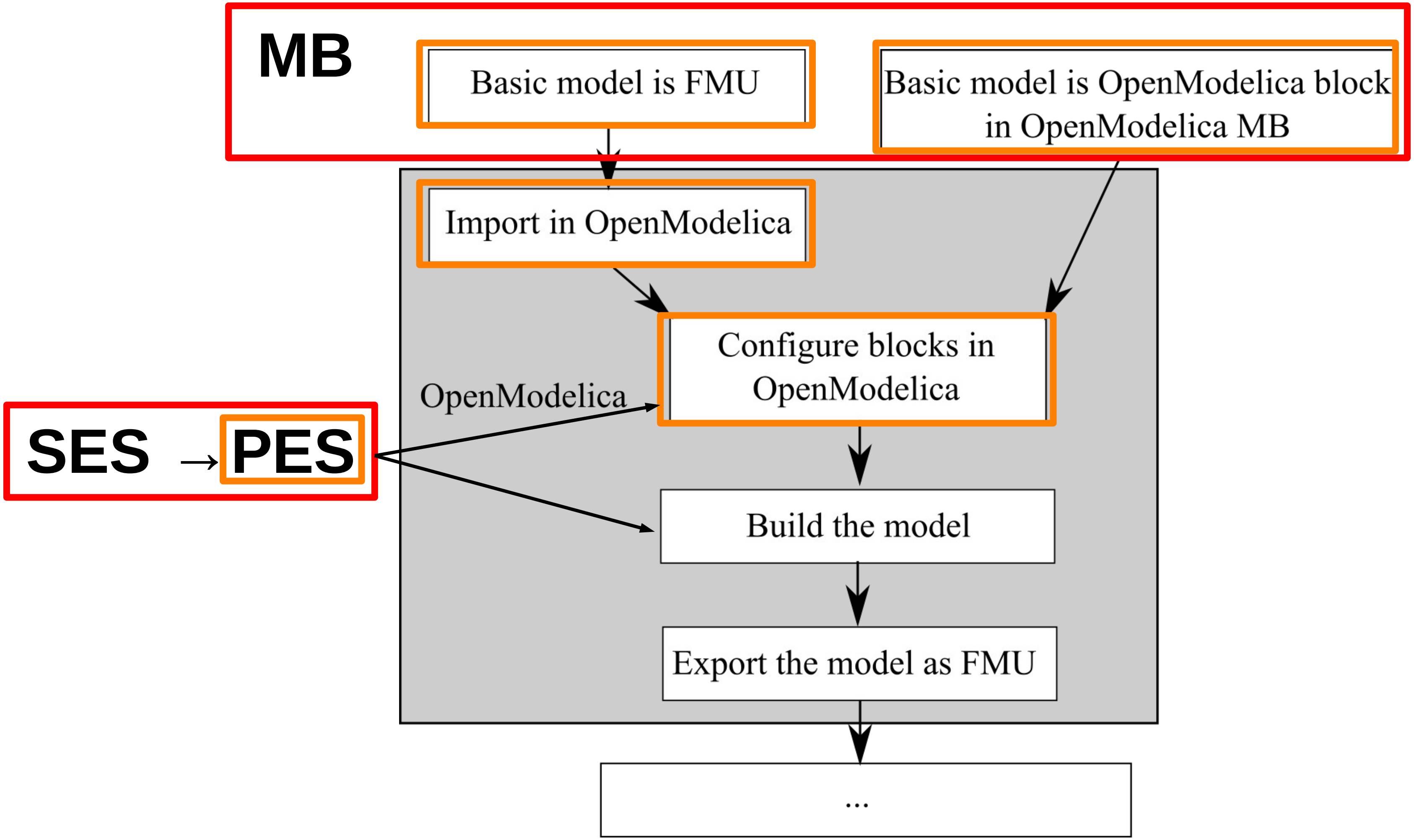
Model Building Using FMI



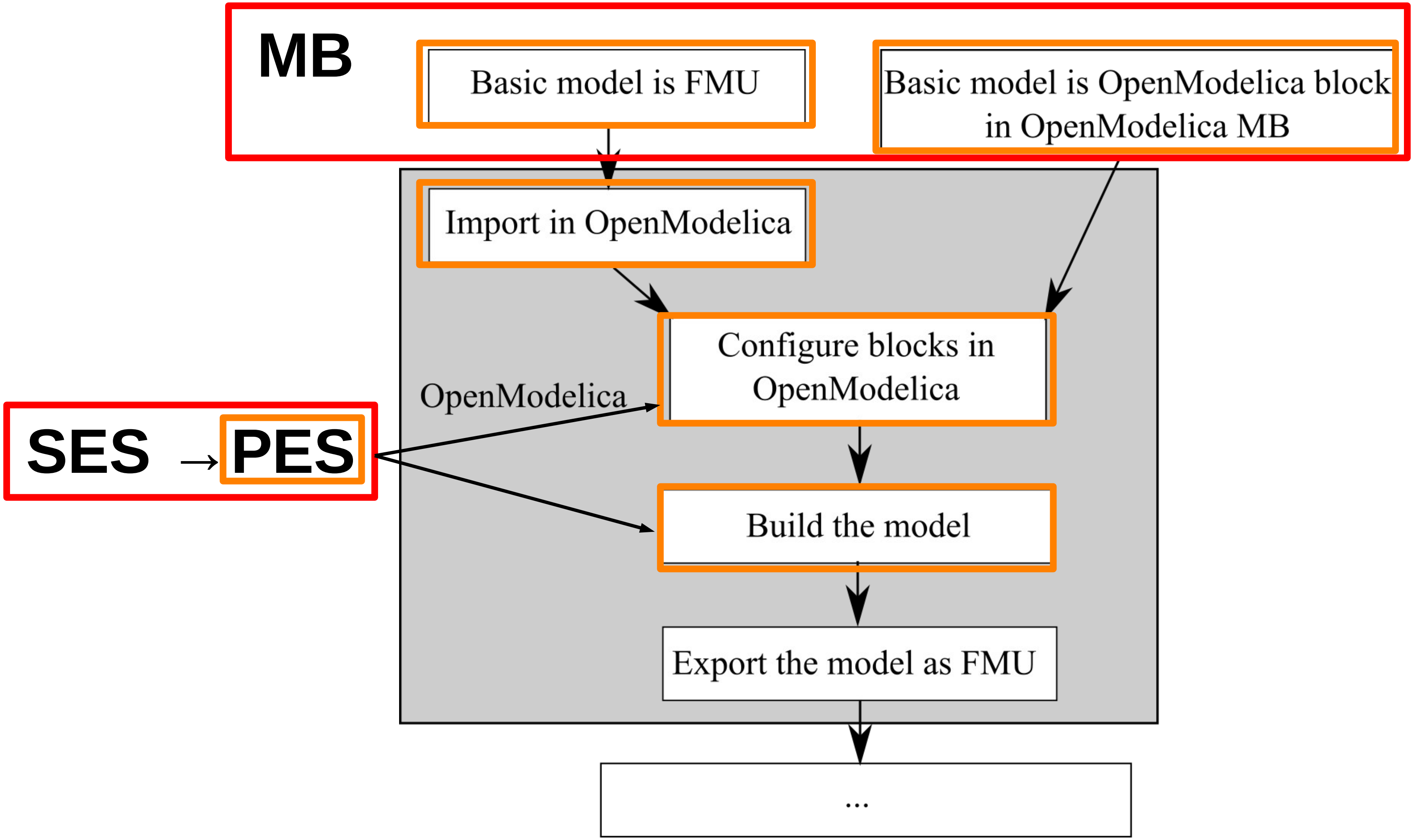
Model Building Using FMI



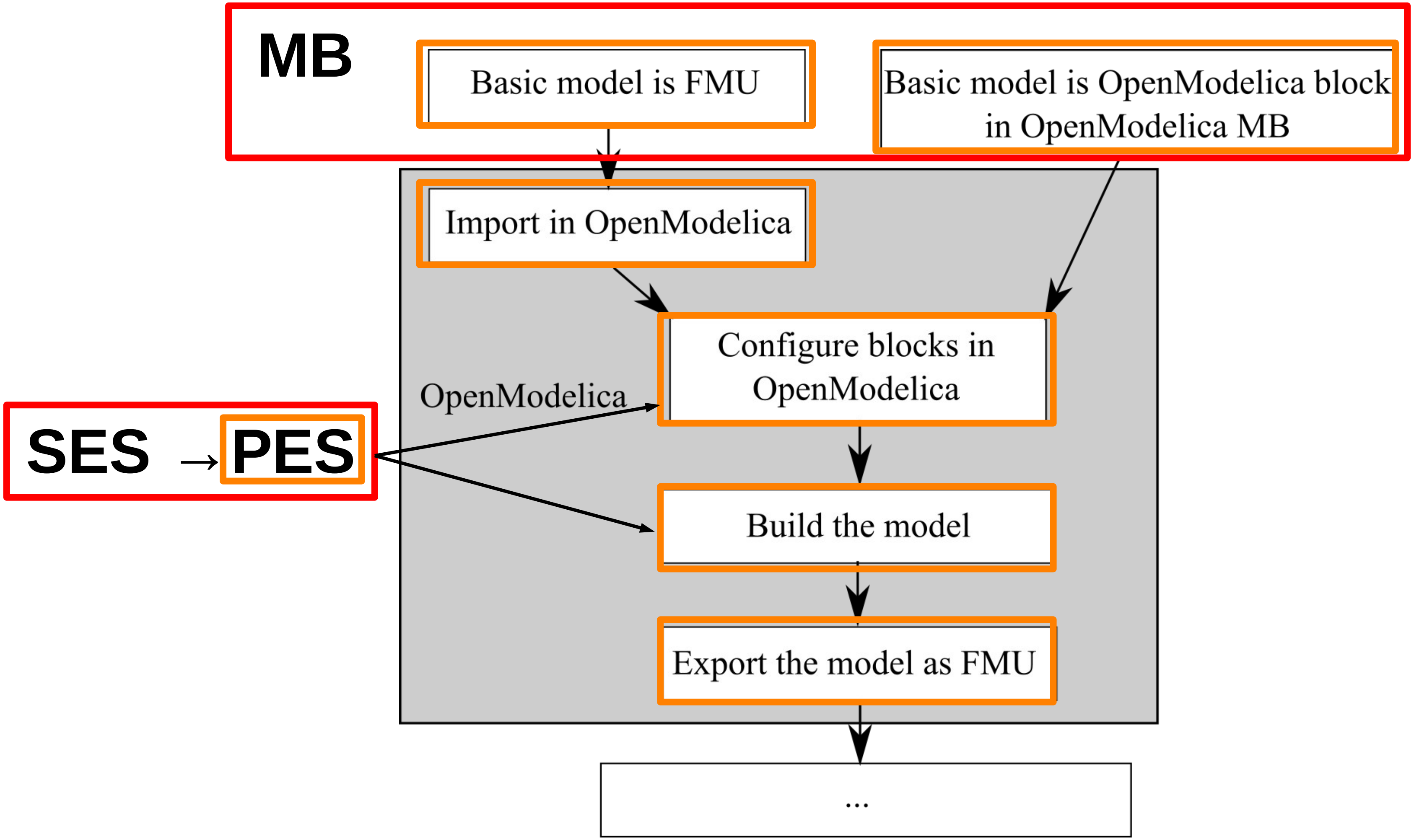
Model Building Using FMI



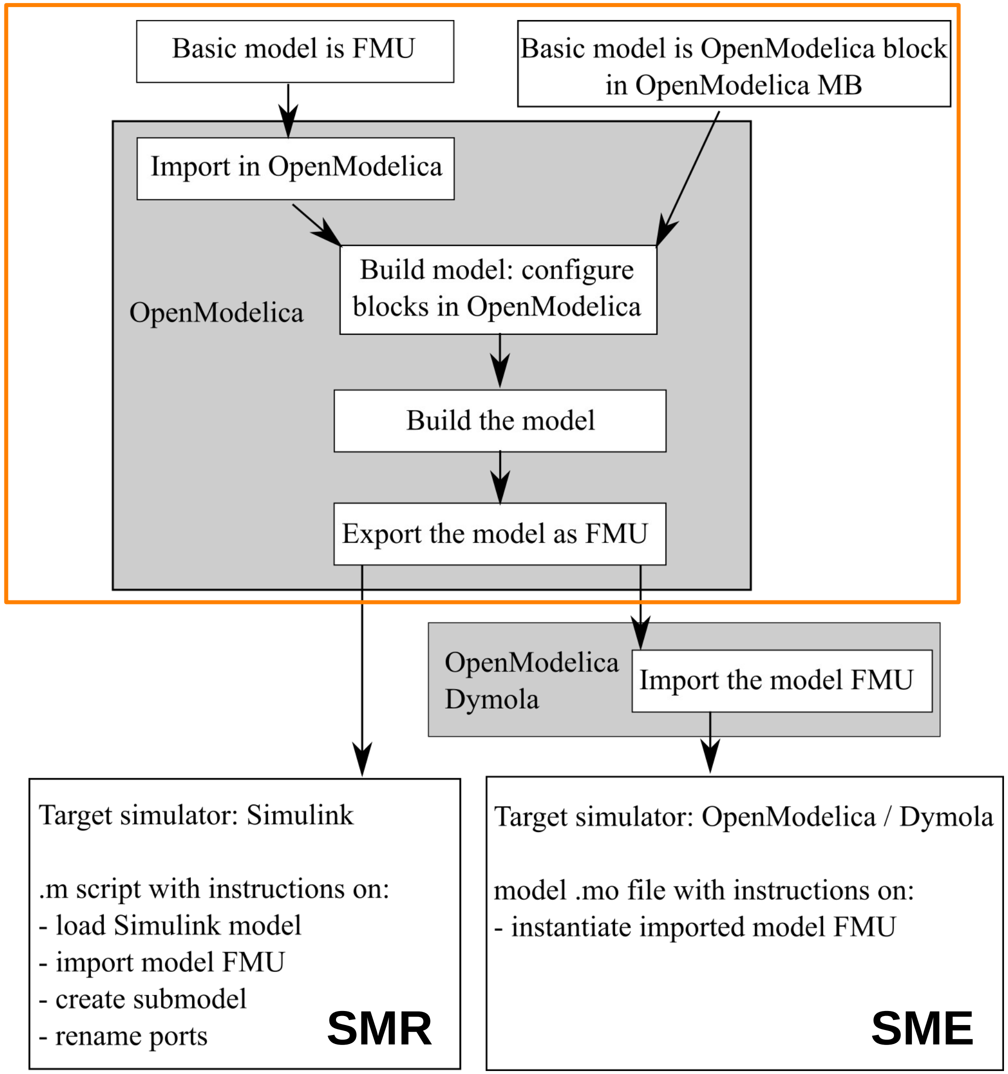
Model Building Using FMI



Model Building Using FMI

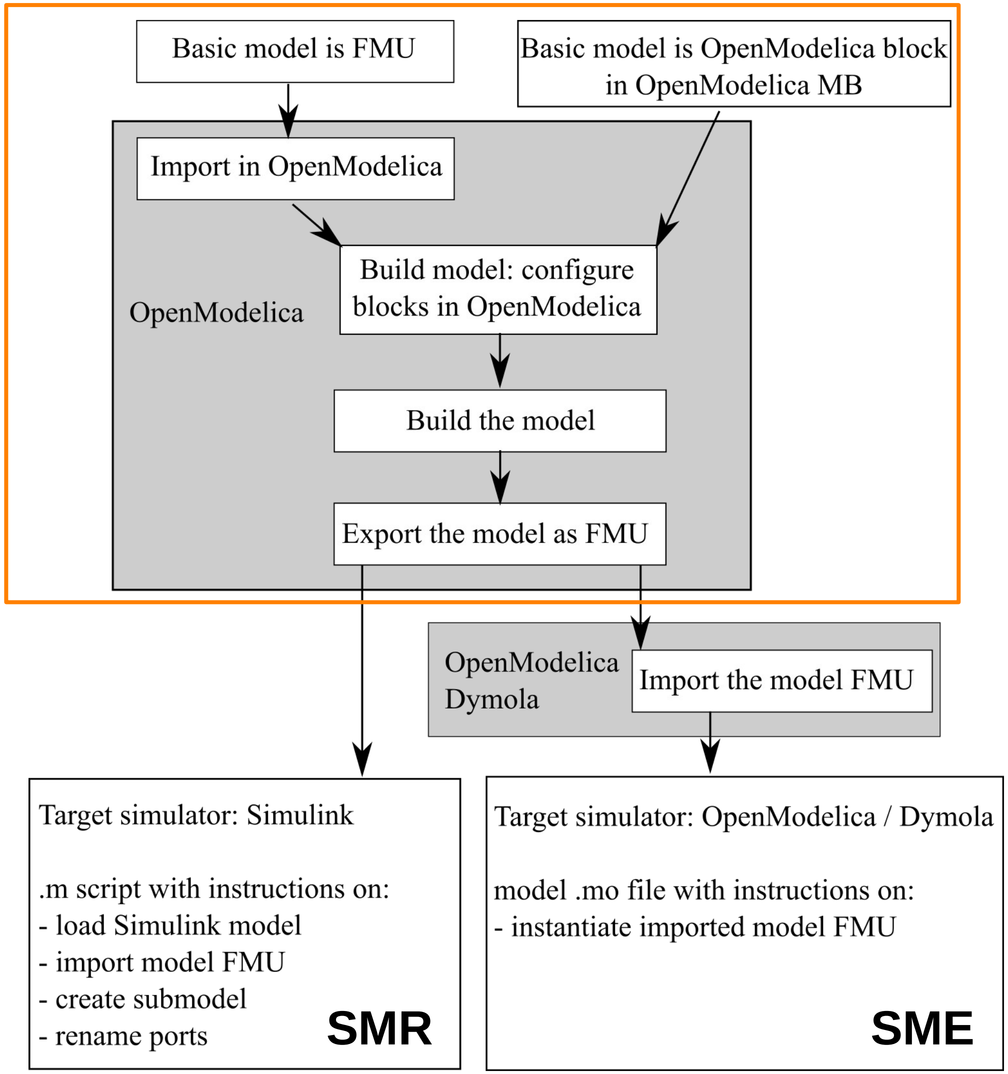


Model Building Using FMI (2)



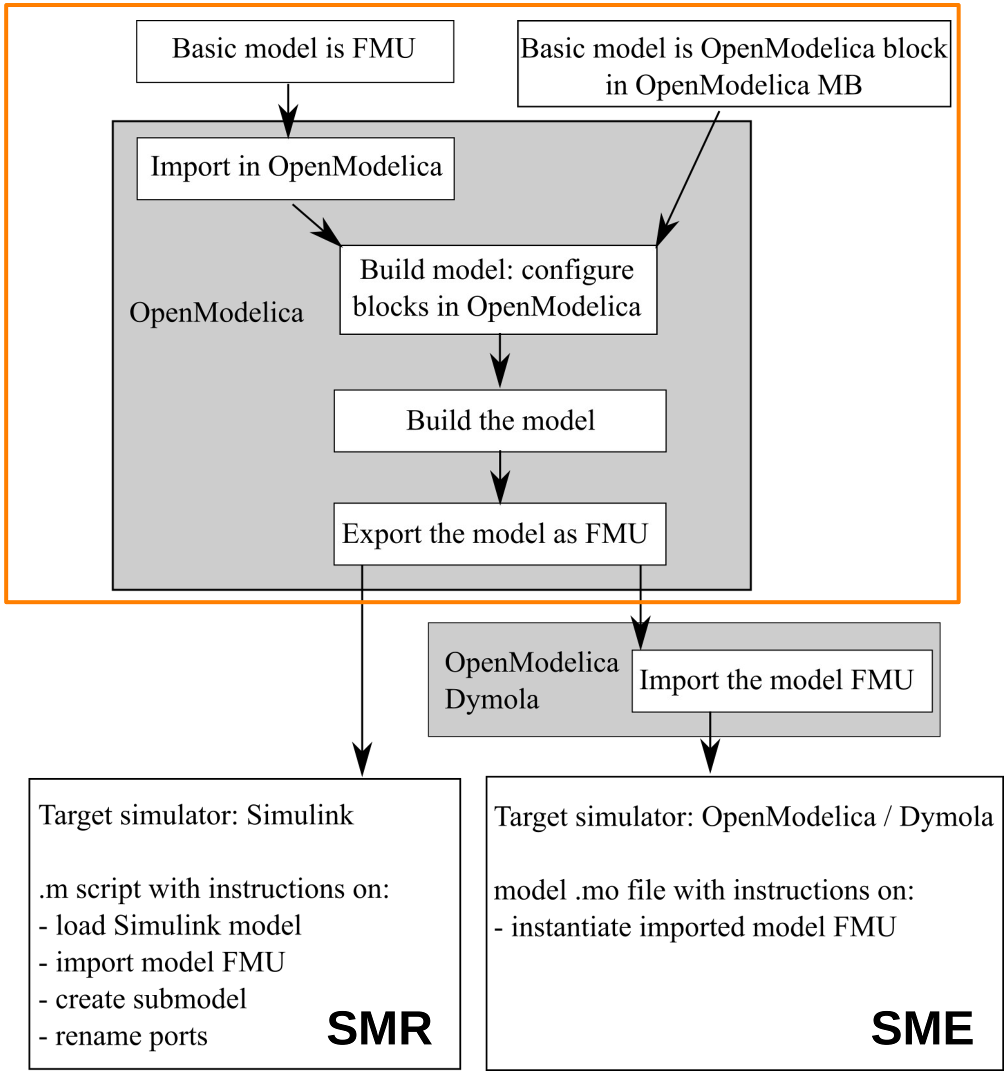
Model Building Using FMI (2)

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



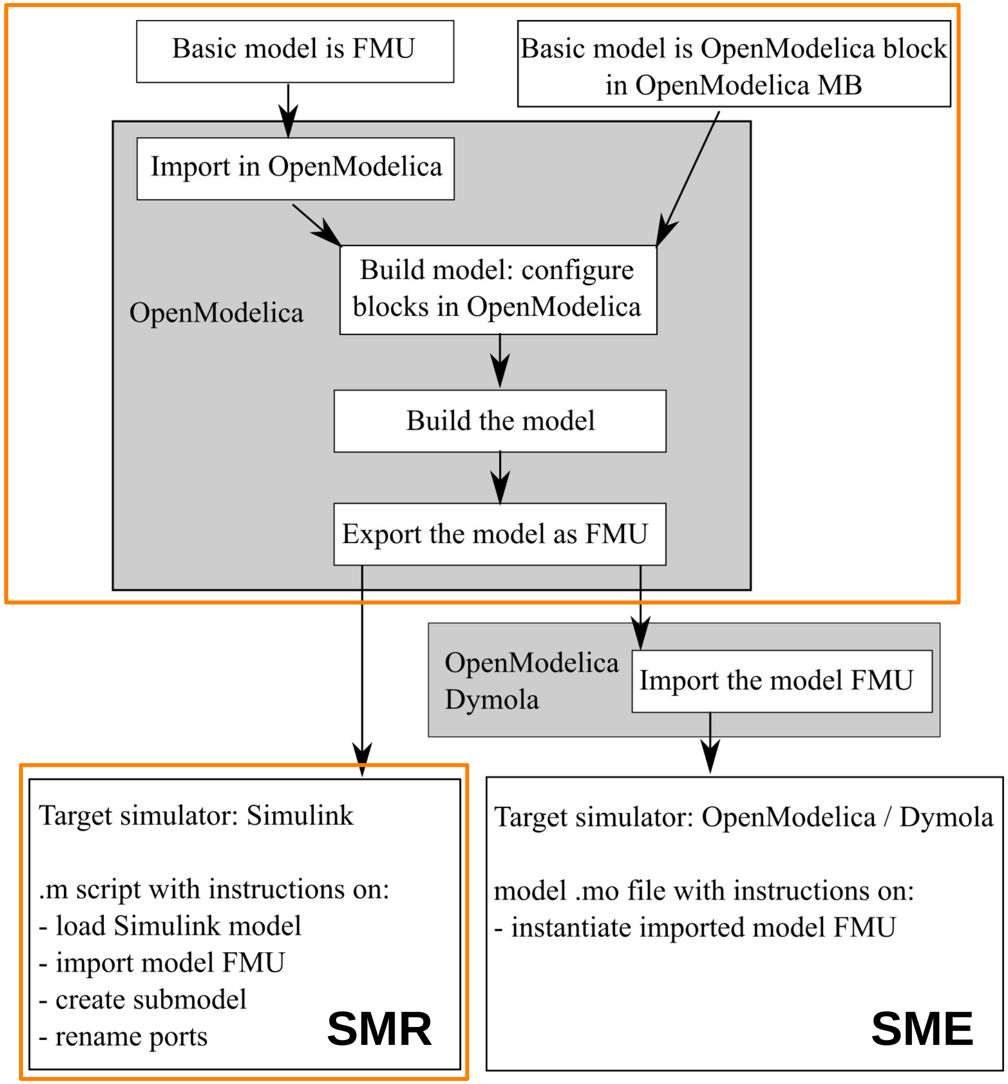
Model Building Using FMI (2)

- **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)**



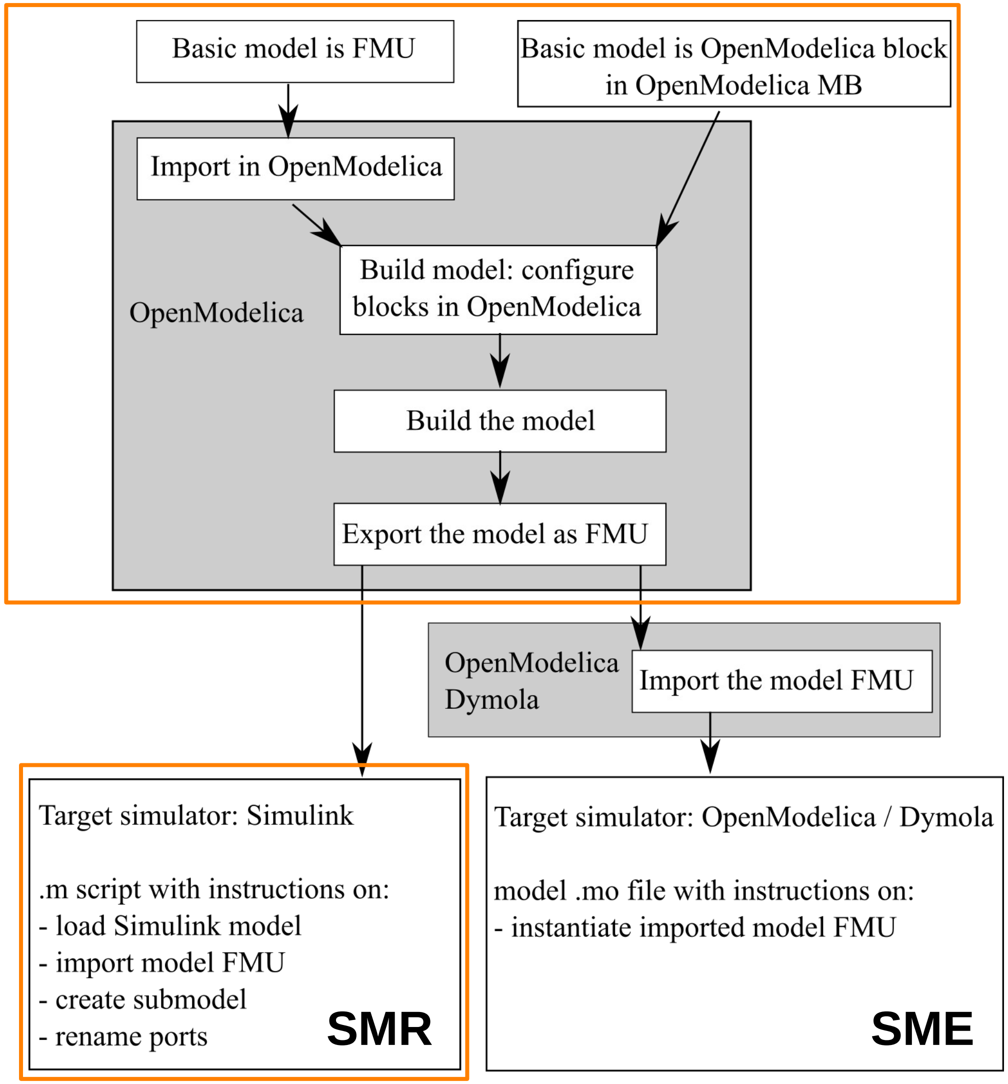
Model Building Using FMI (2)

- **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)**



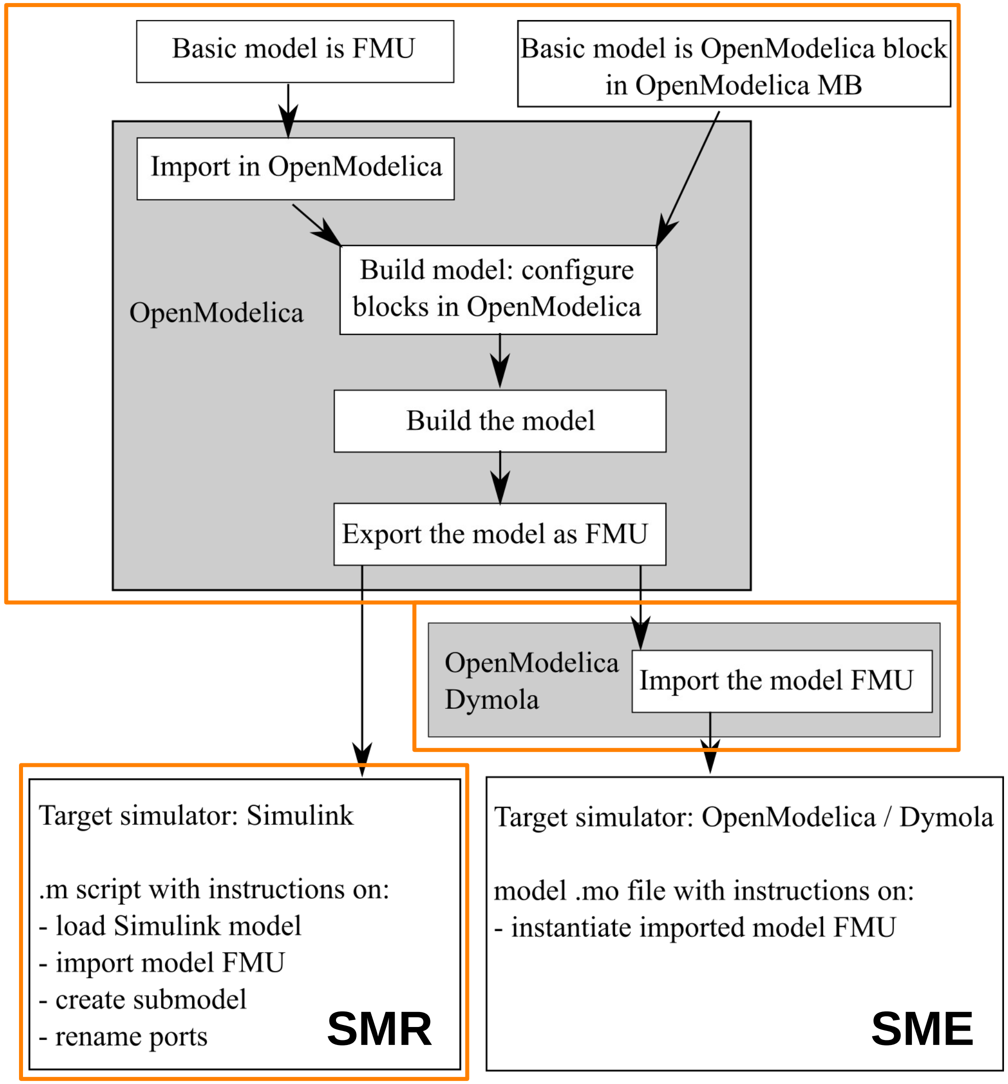
Model Building Using FMI (2)

- **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)**



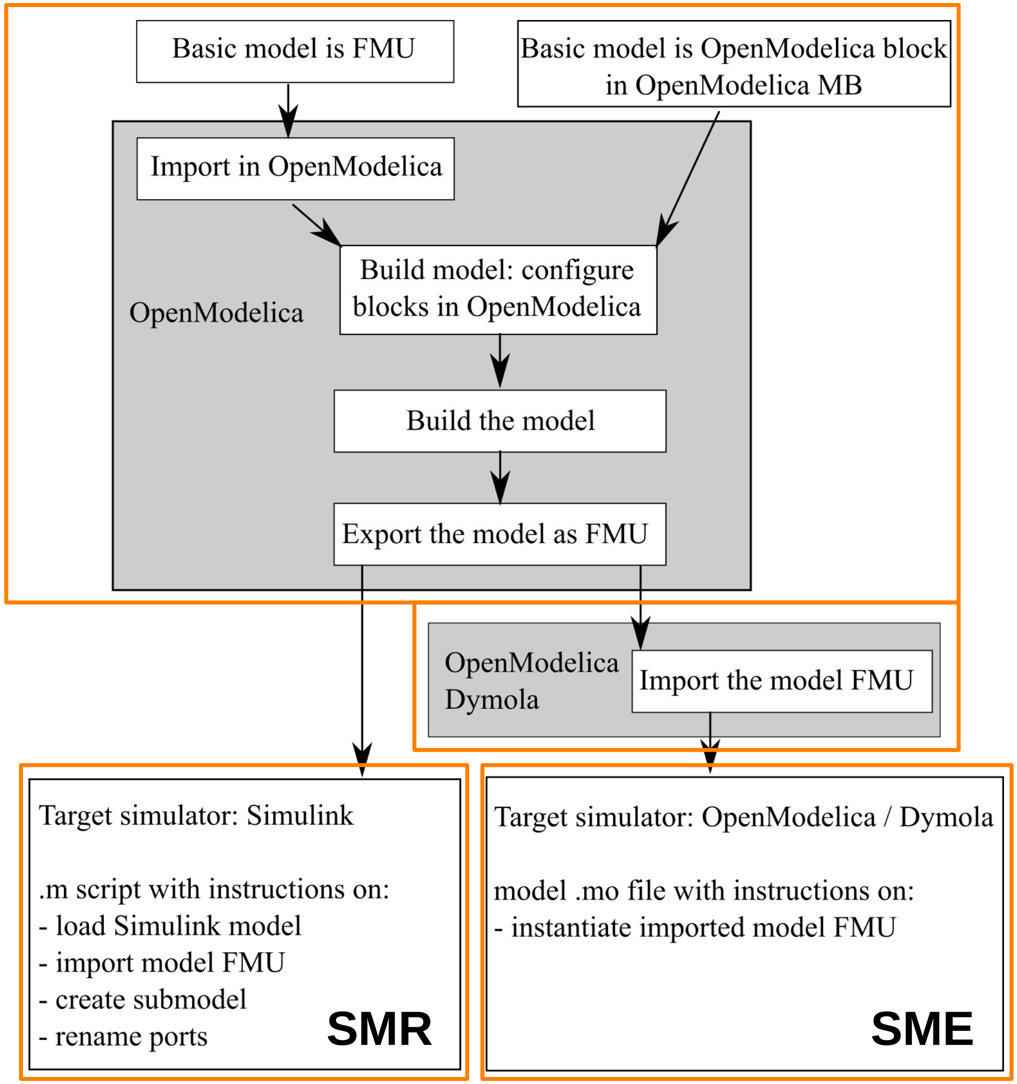
Model Building Using FMI (2)

- **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)**

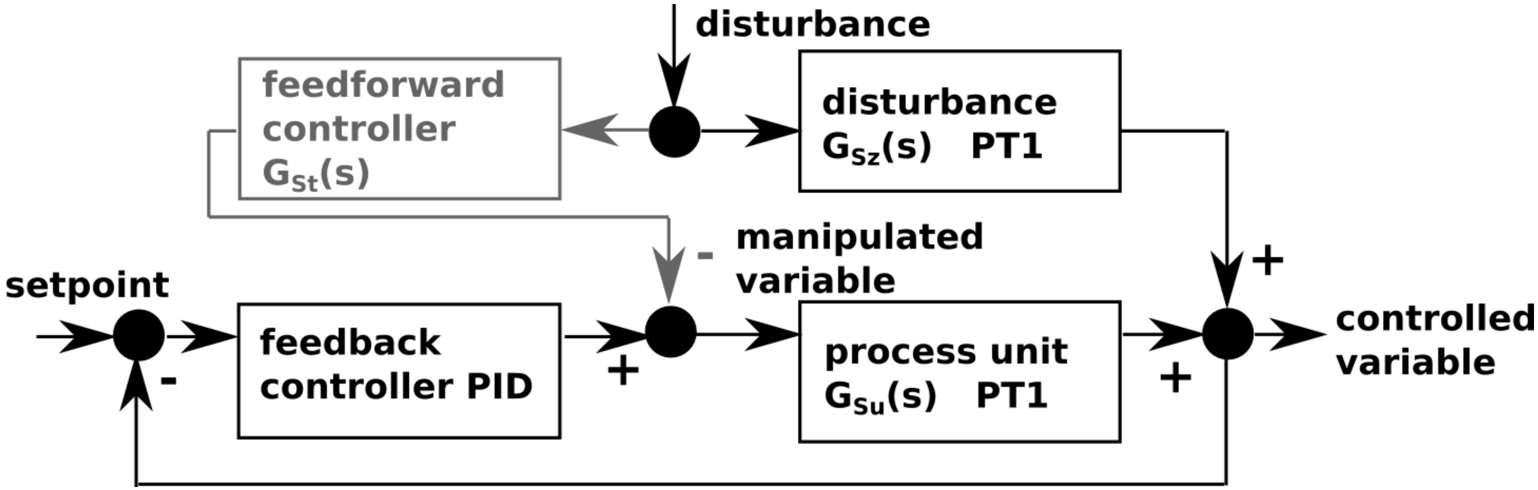


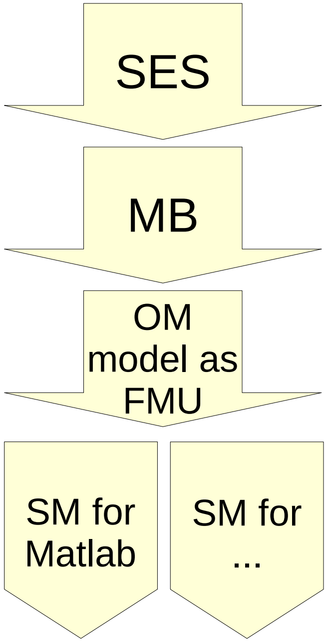
Model Building Using FMI (2)

- **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)**



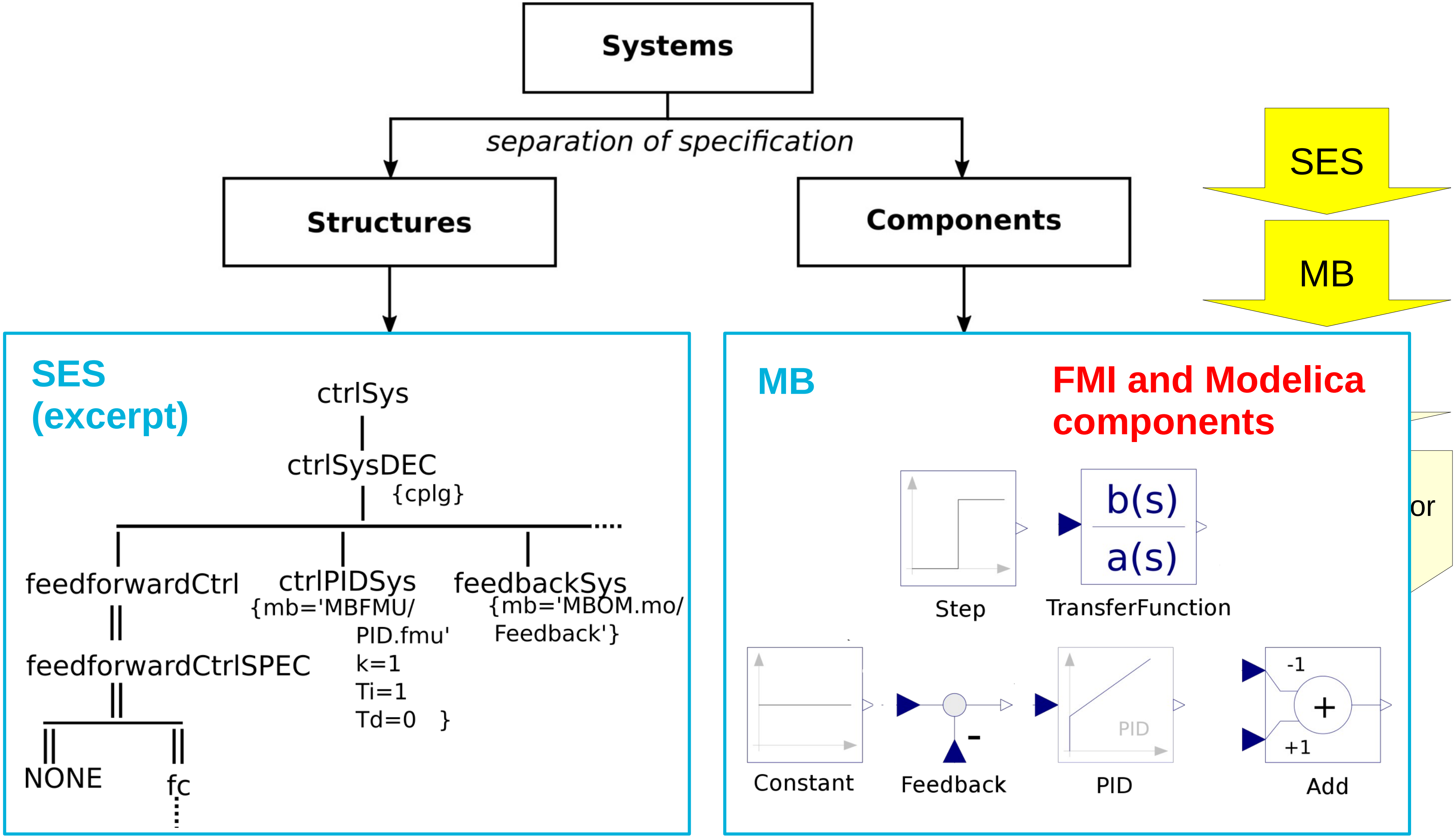
Case Study with FMI



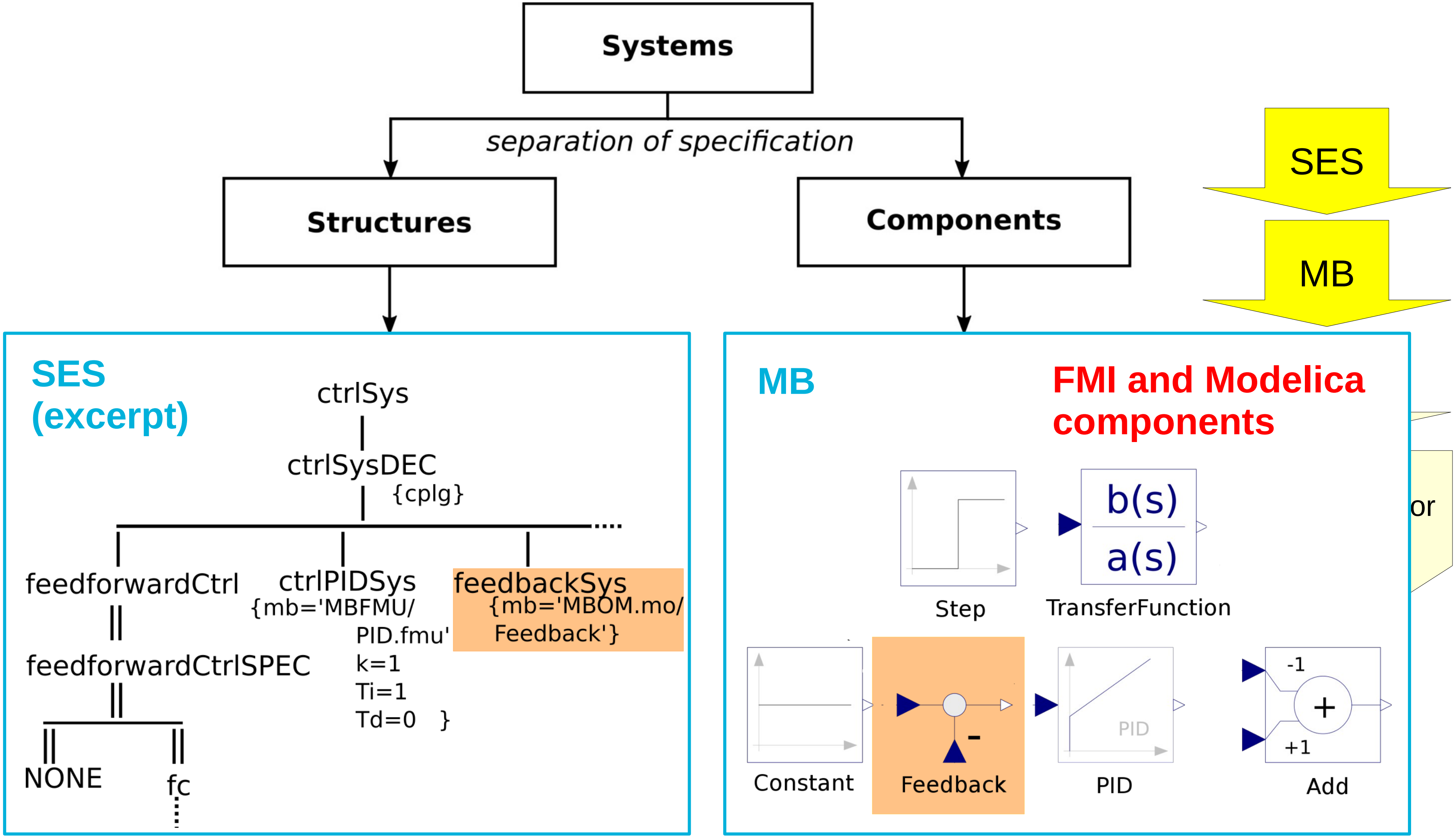


- FMI-based approach implemented in SESMoPy
- Flattened PES for the FMI-based case study in the `examples/Example03_FeedbackControl_FMI` directory of SESMoPy: `Feedback_FPES.jsonsestree`
- Usage of SESMoPy as presented before

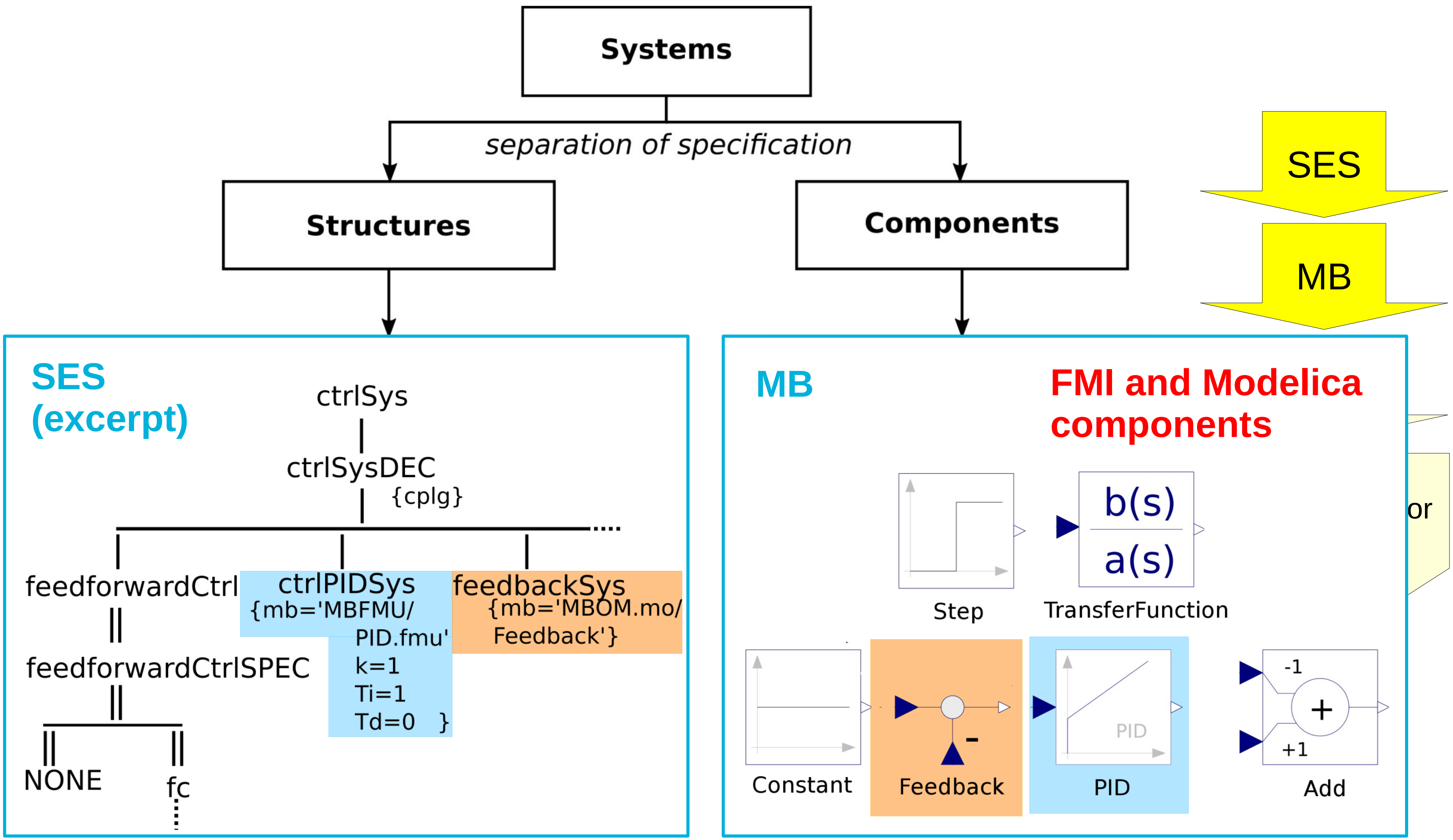
SES/MB Approach with FMI Components



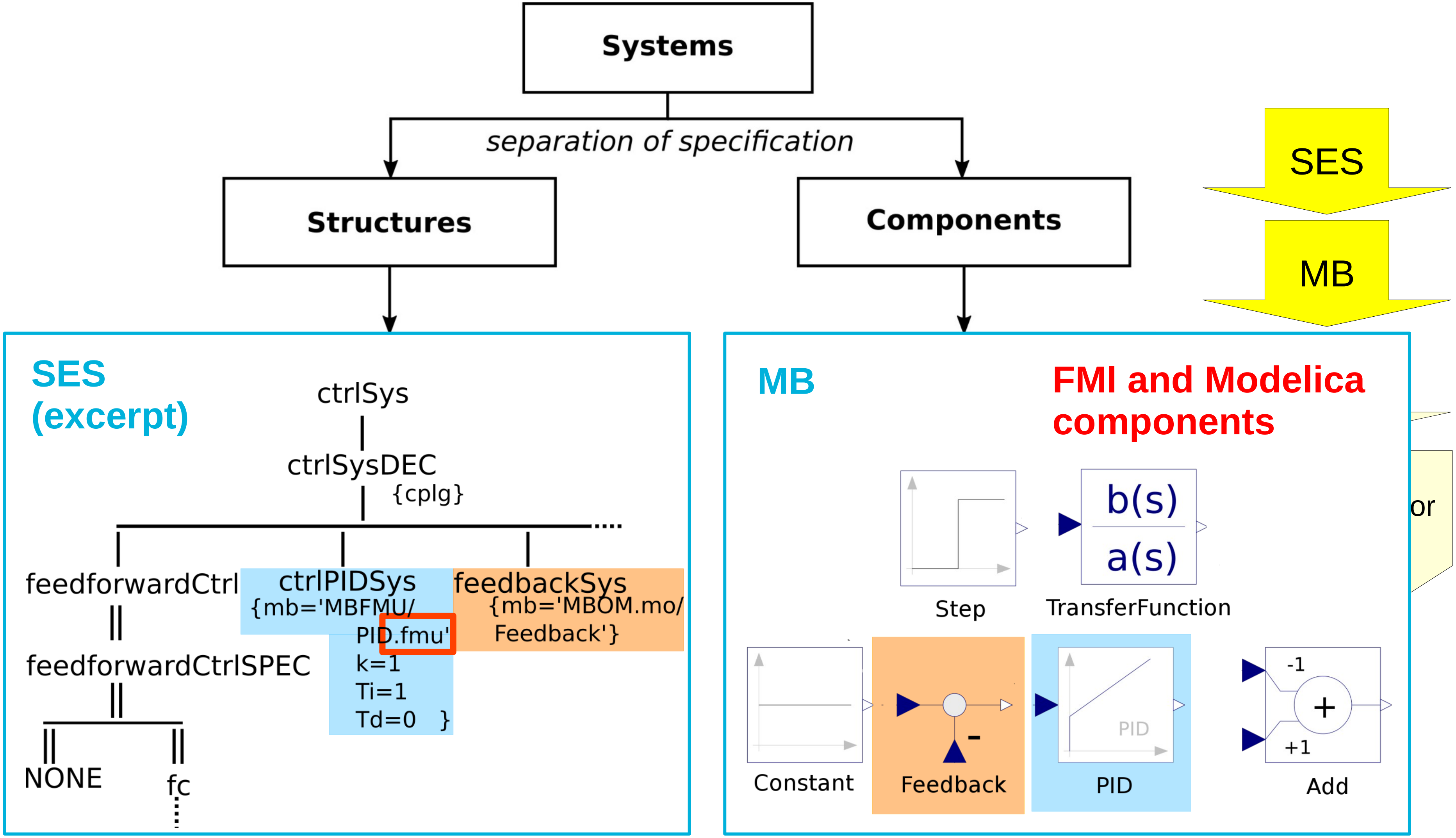
SES/MB Approach with FMI Components



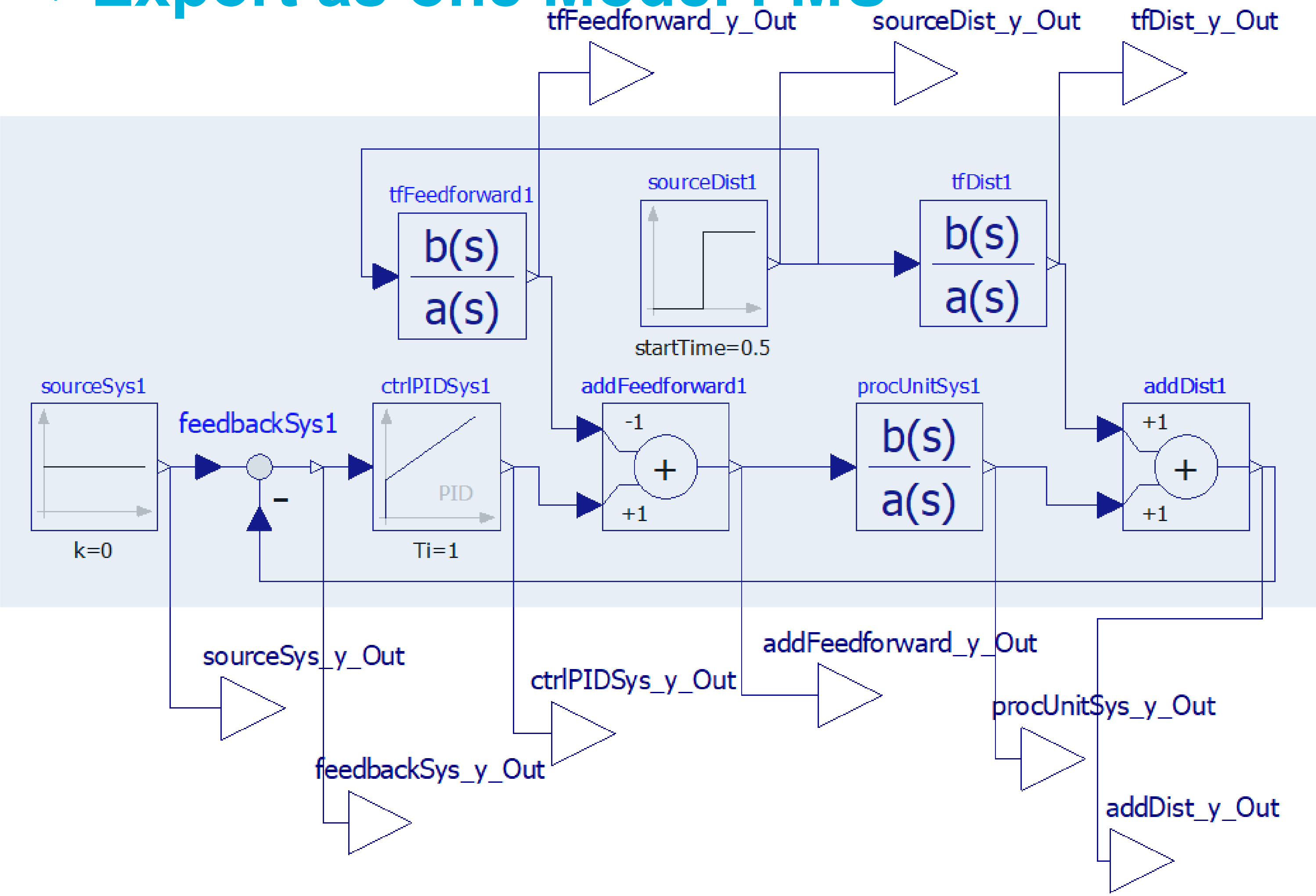
SES/MB Approach with FMI Components

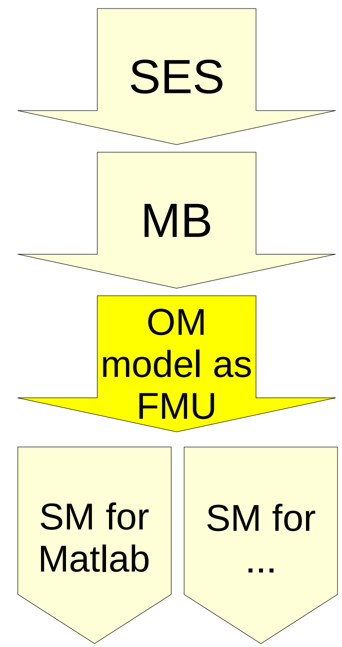


SES/MB Approach with FMI Components

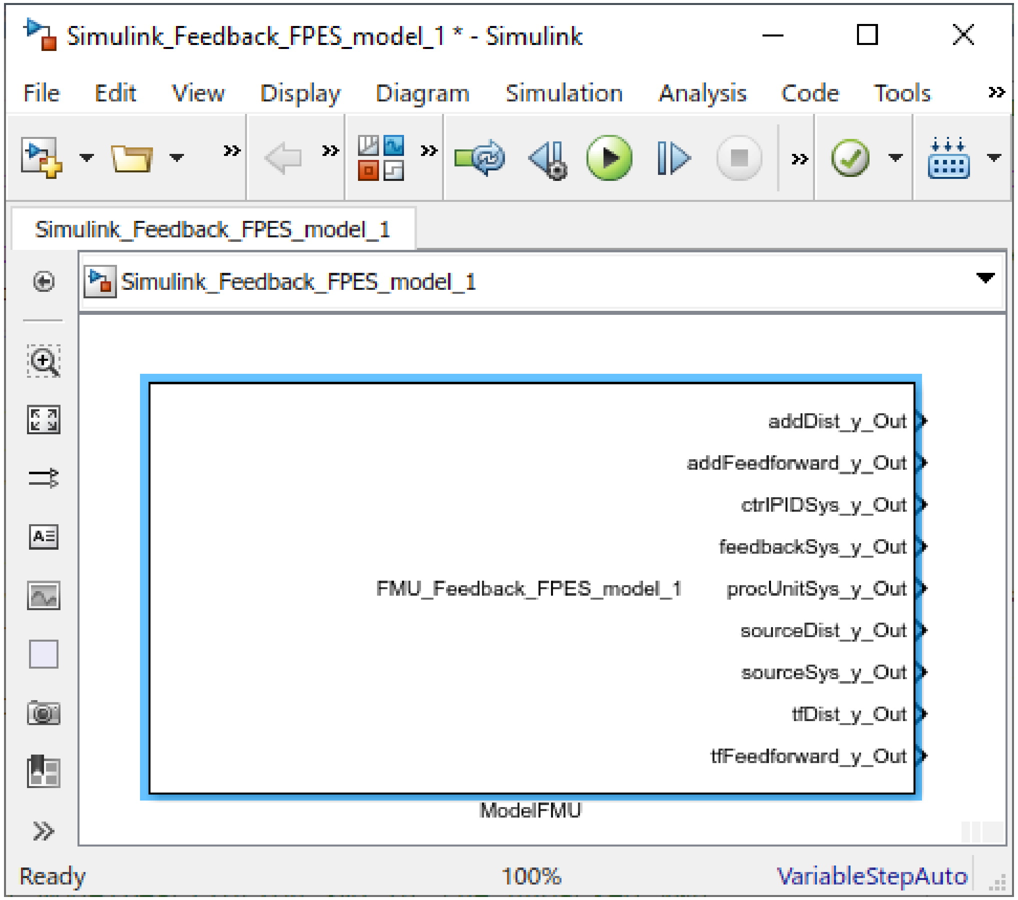


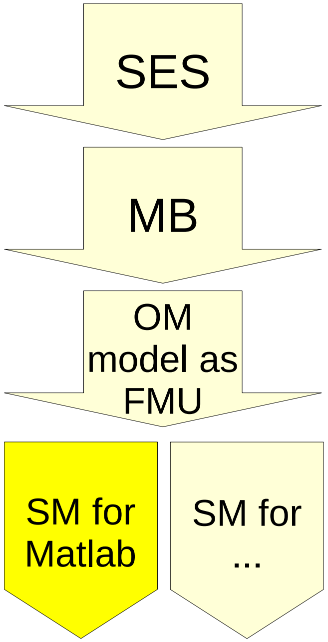
Generated Model in OpenModelica → Export as one Model FMU





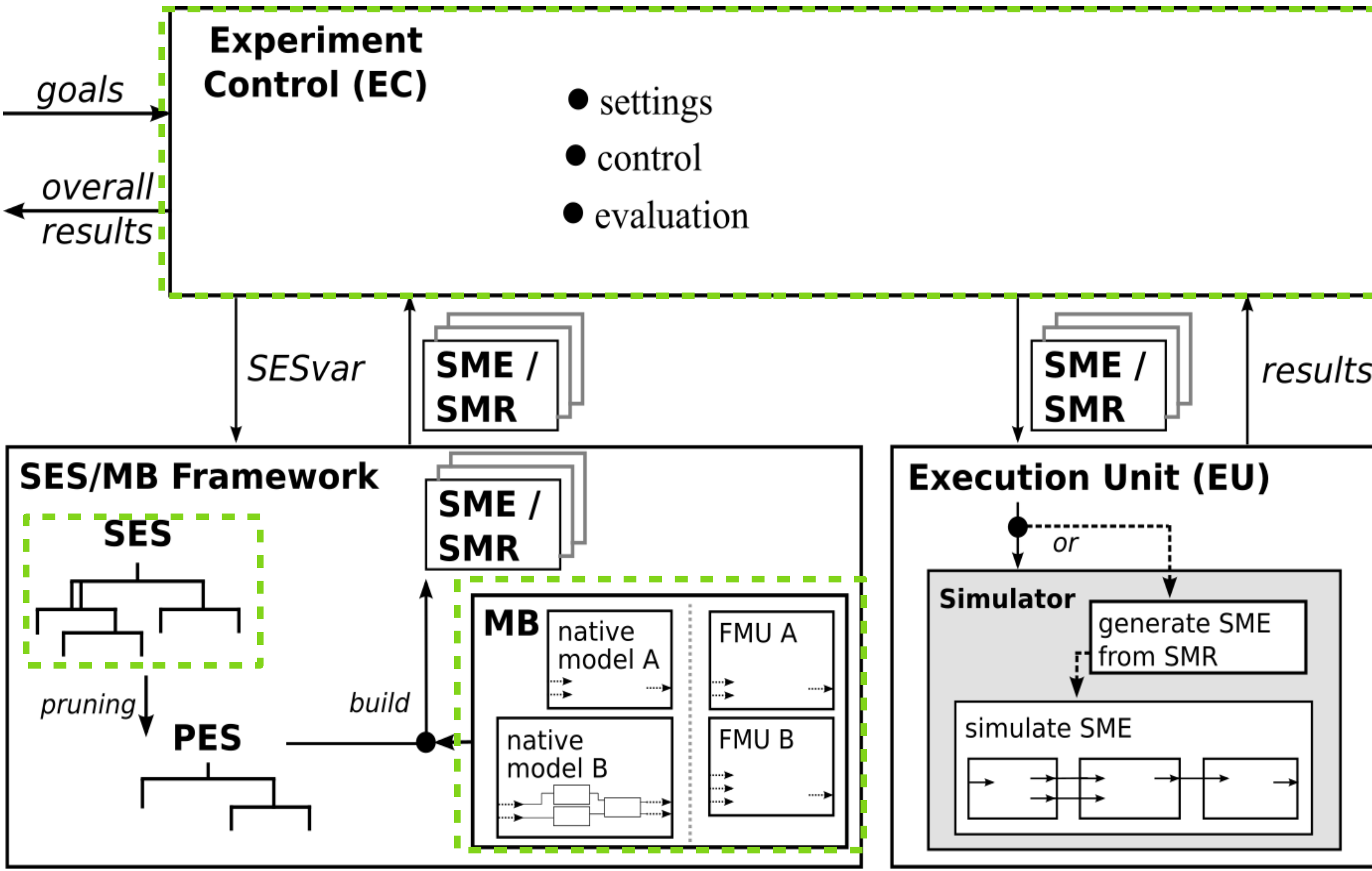
Case Study: Model FMU Imported in MATLAB/Simulink





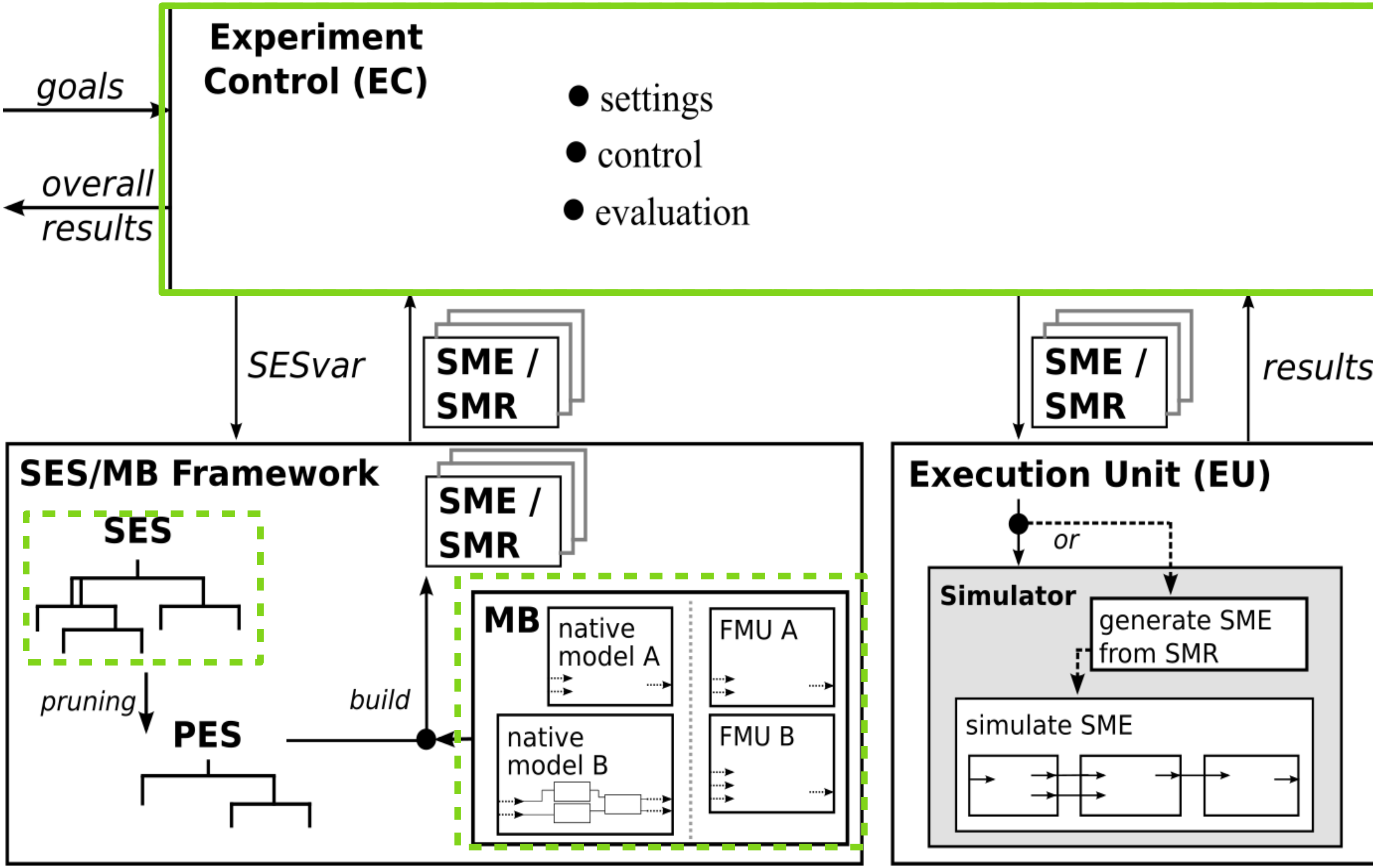
Outline

Extended SES/MB Architecture (Native and FMI)



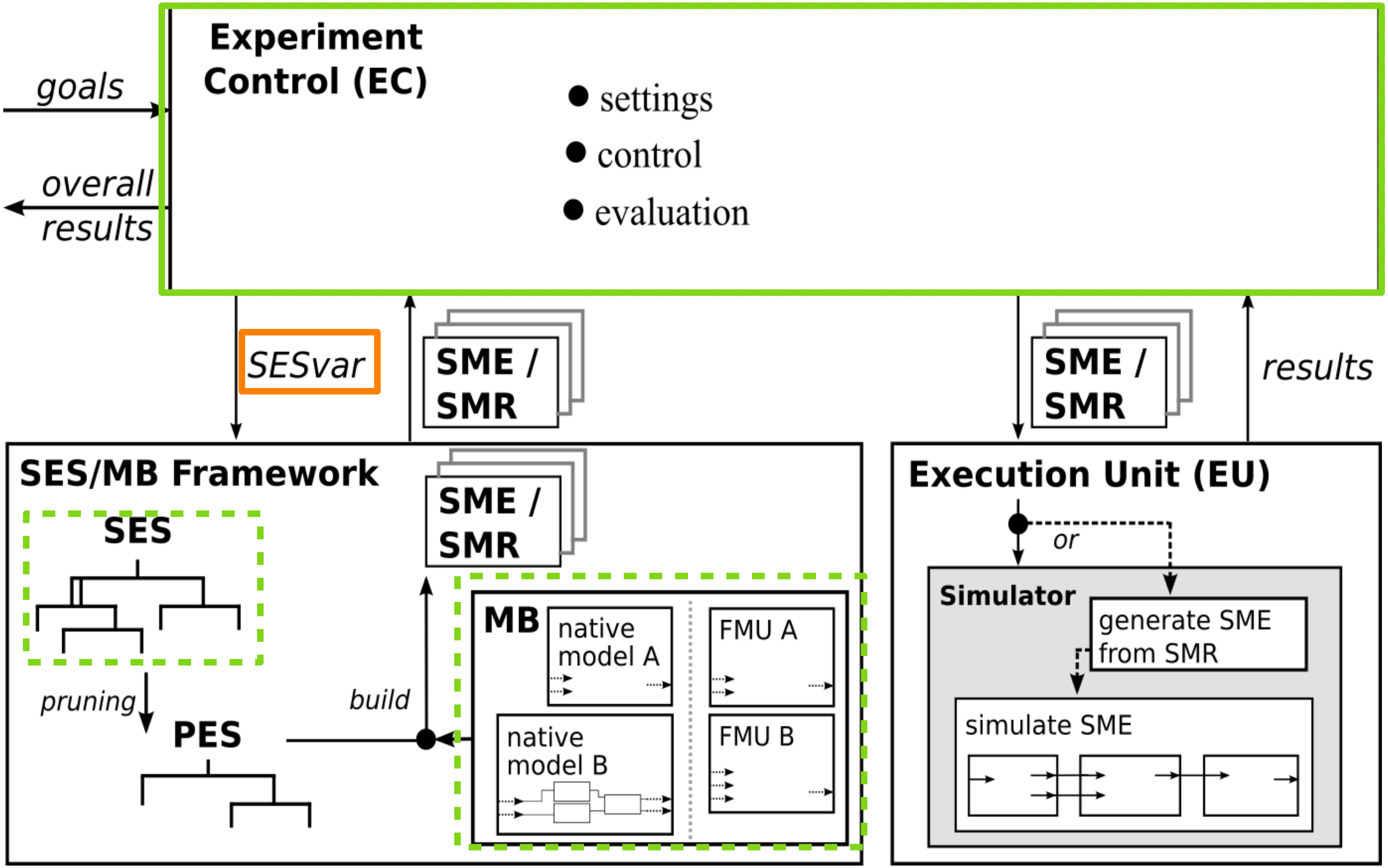
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



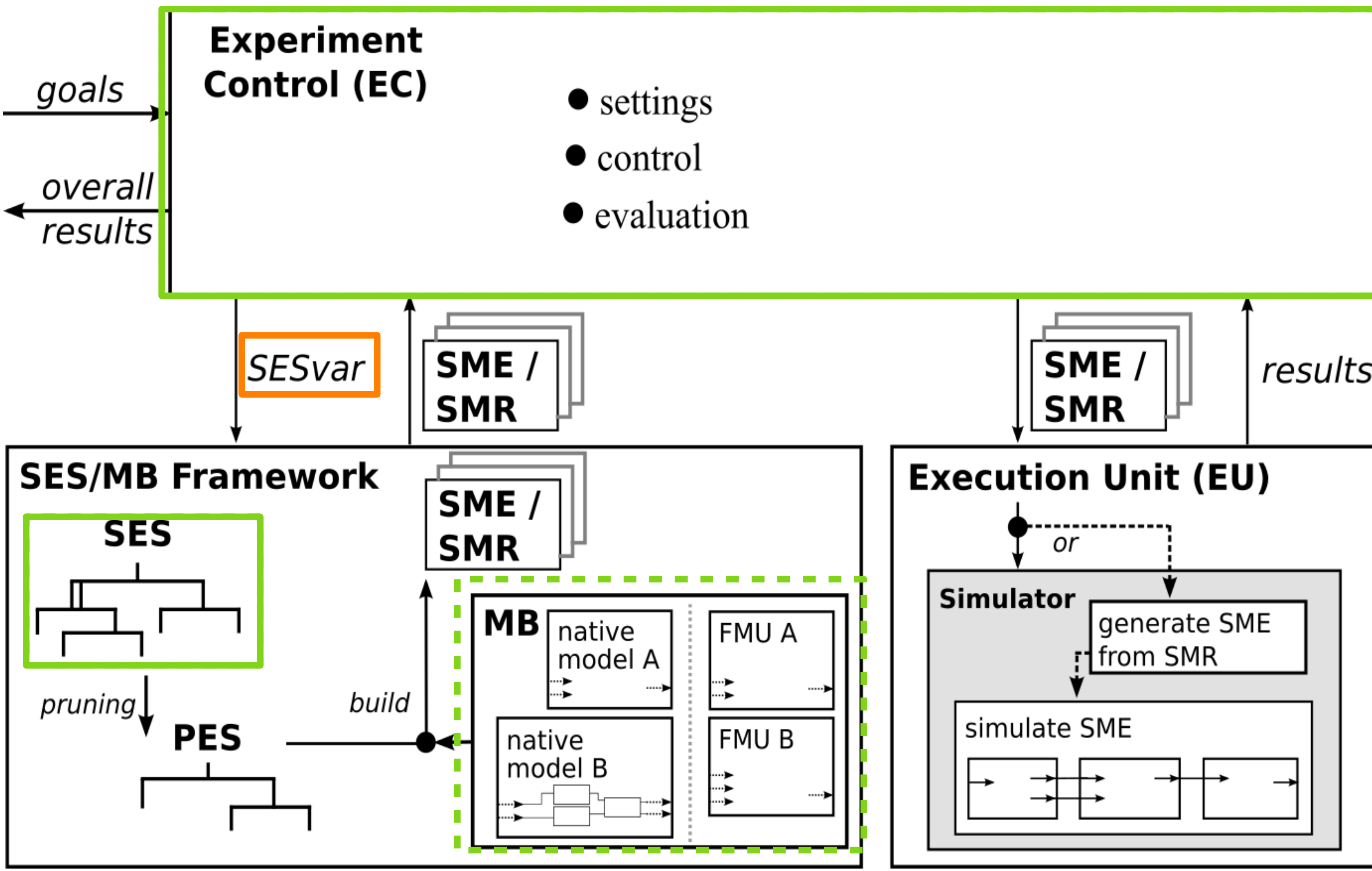
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



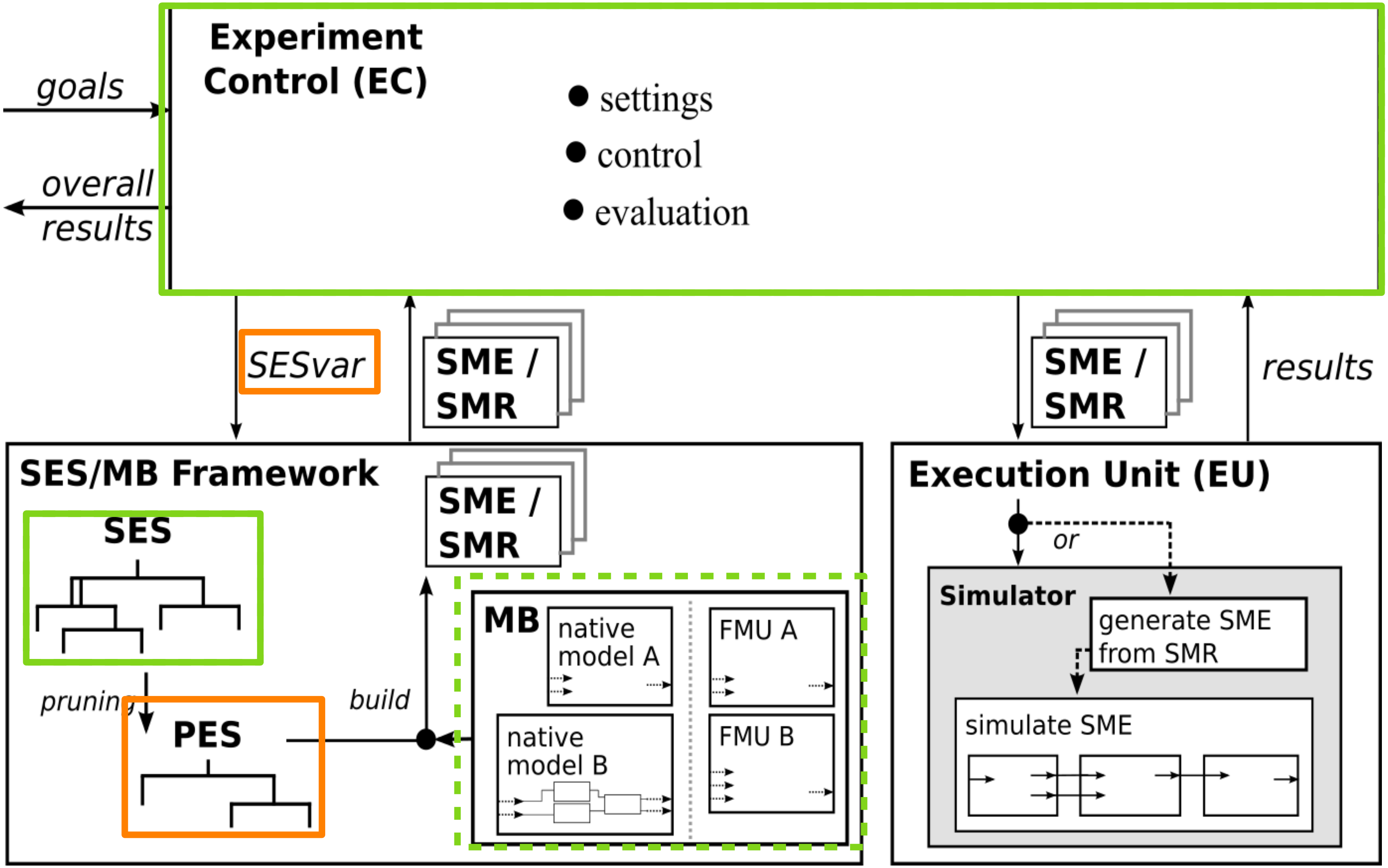
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



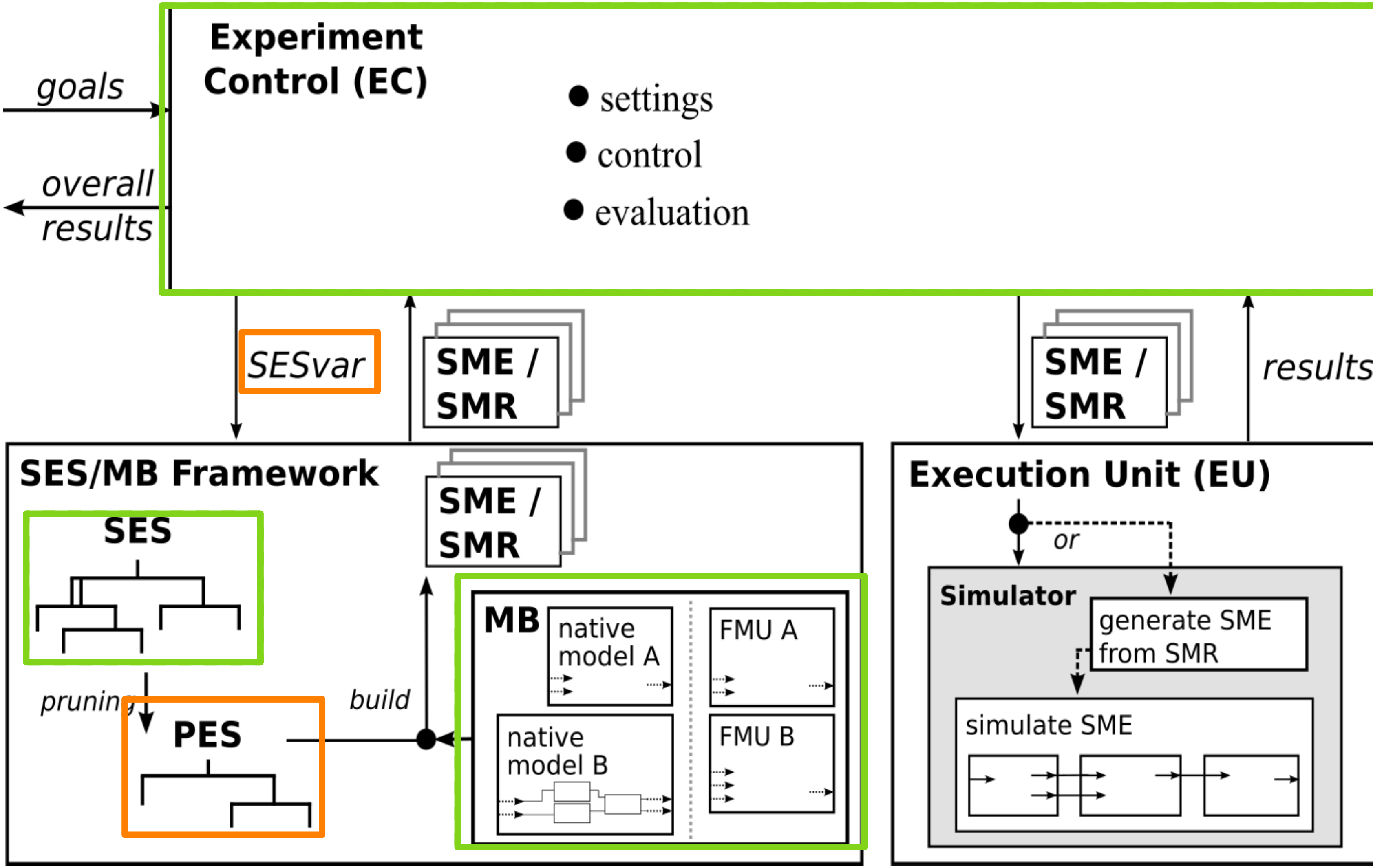
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



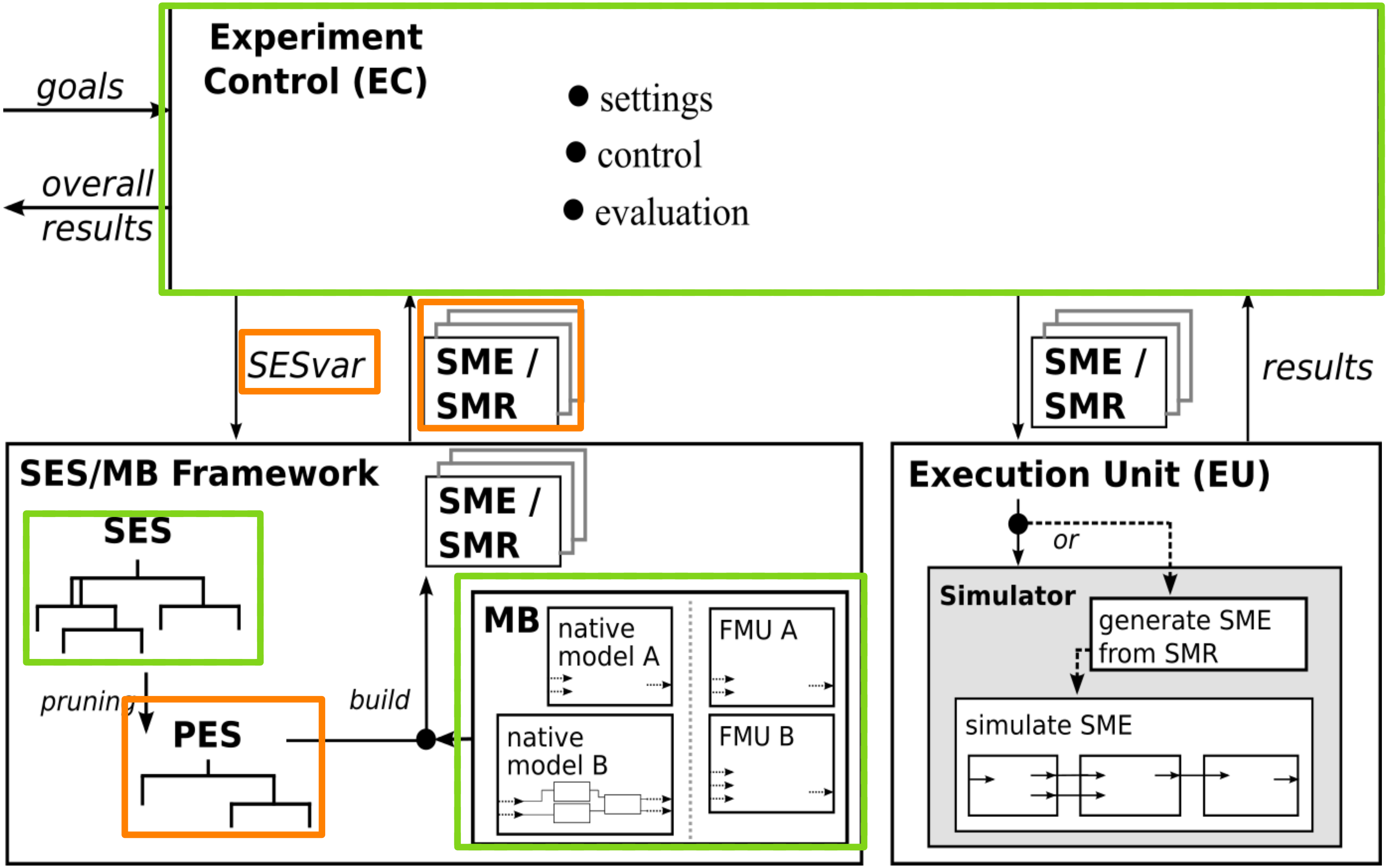
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



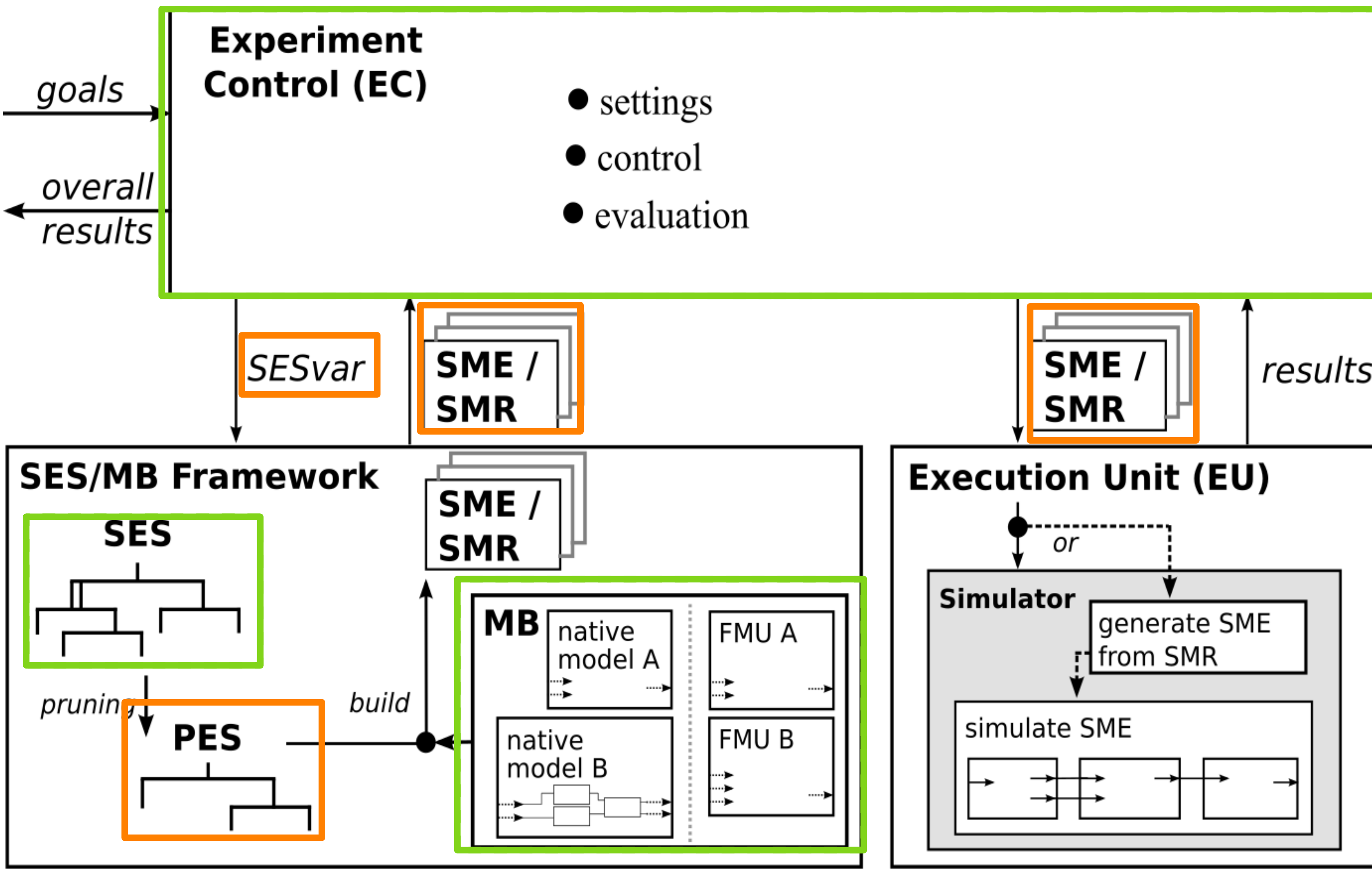
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



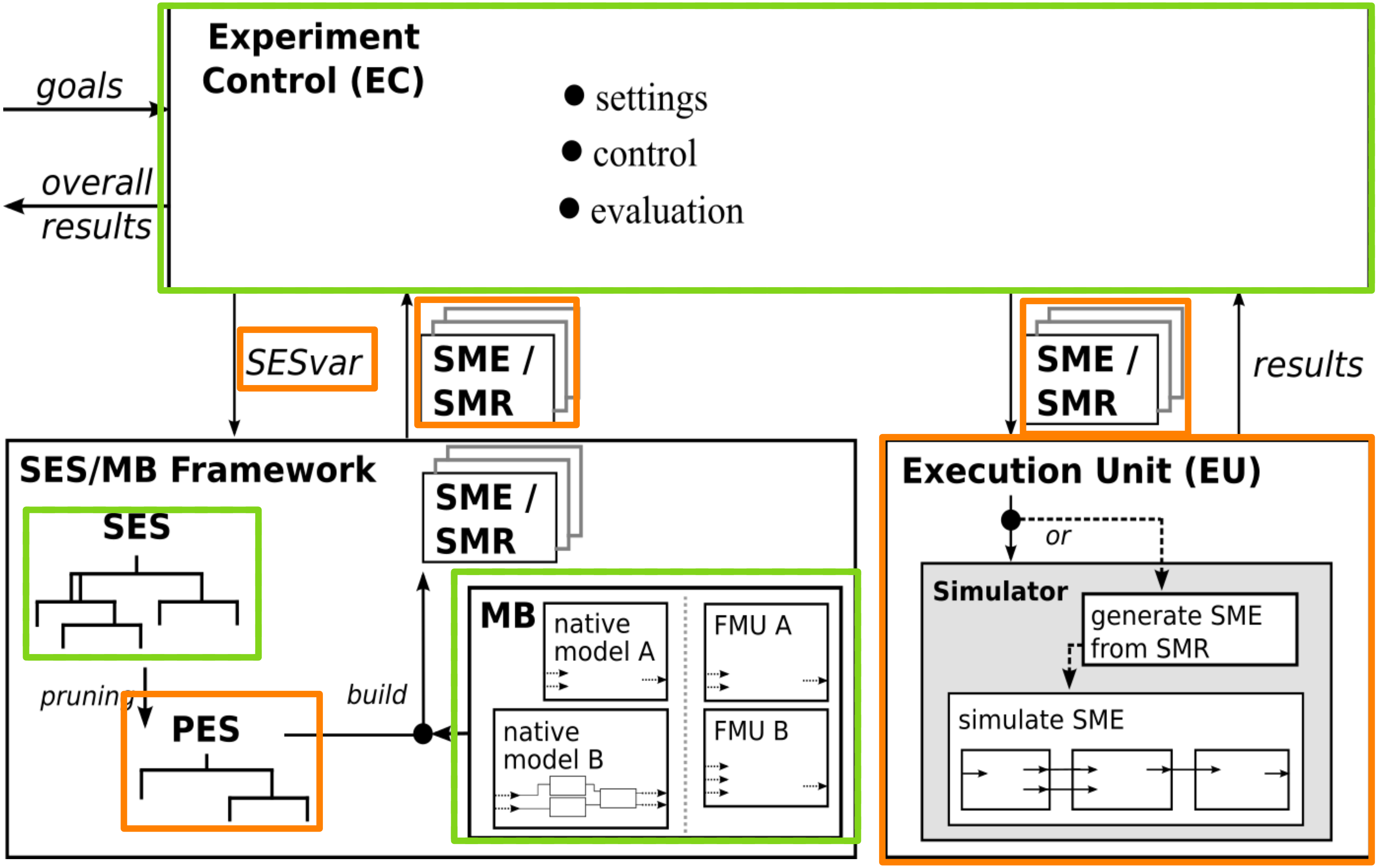
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



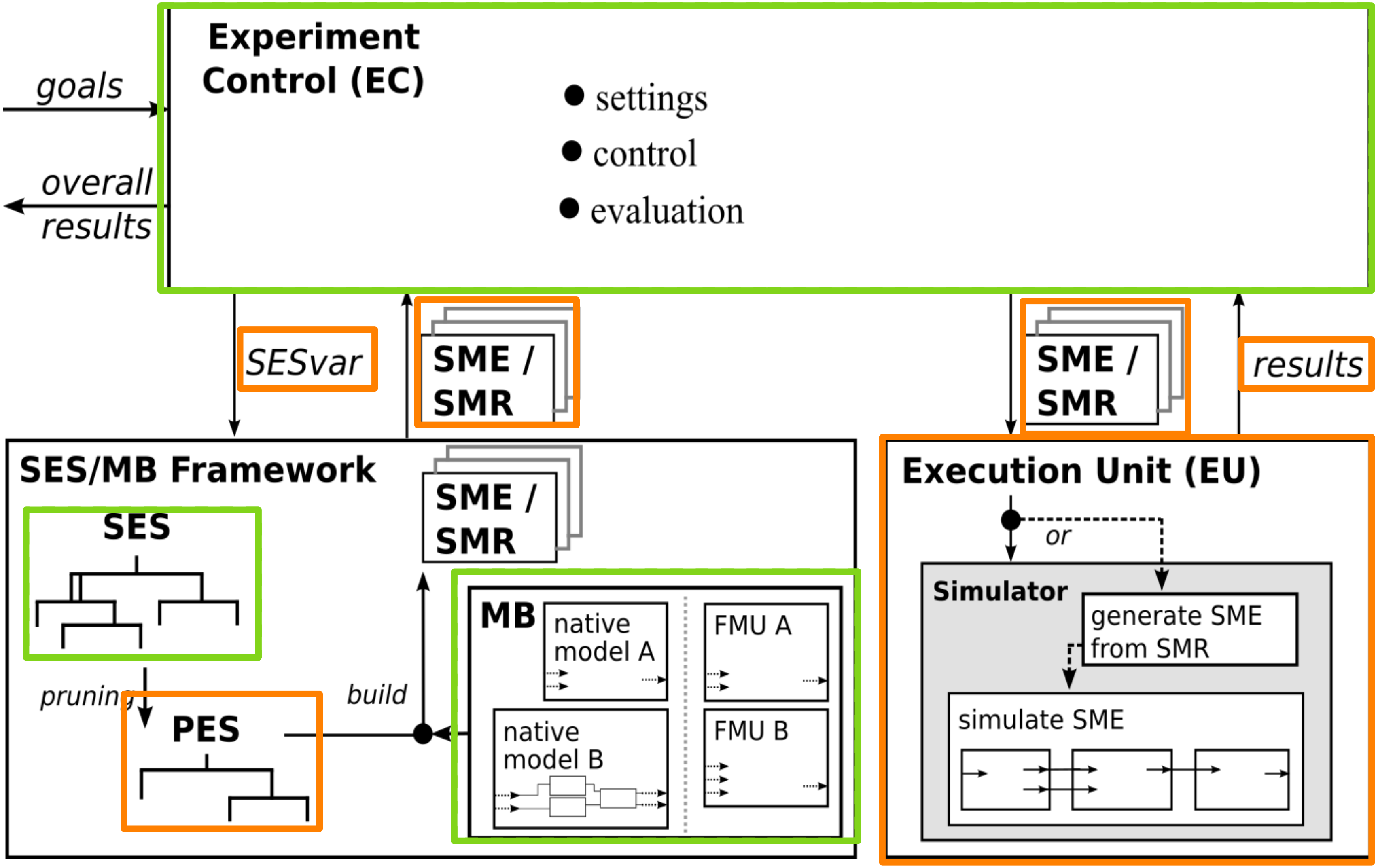
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



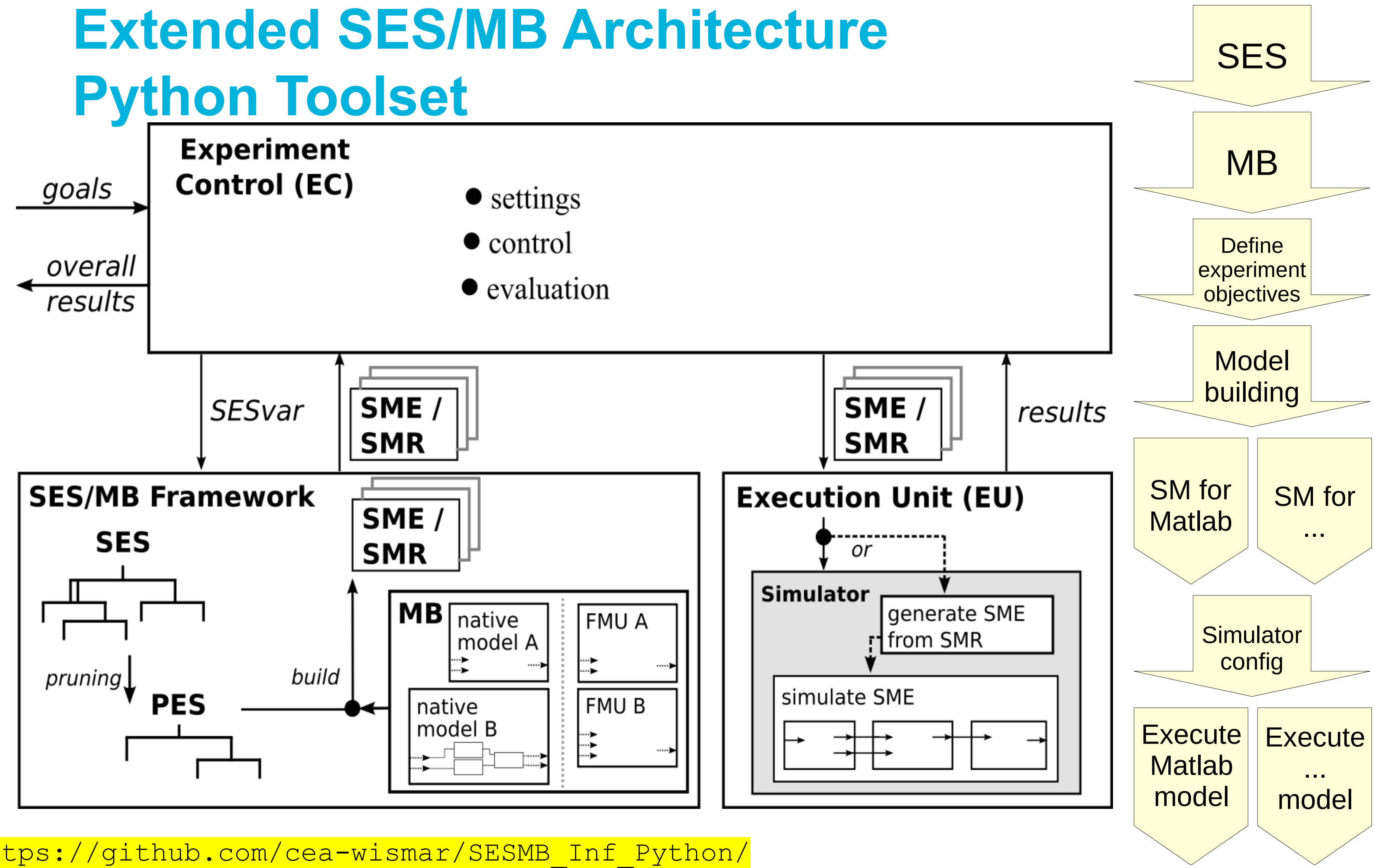
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture (Native and FMI)



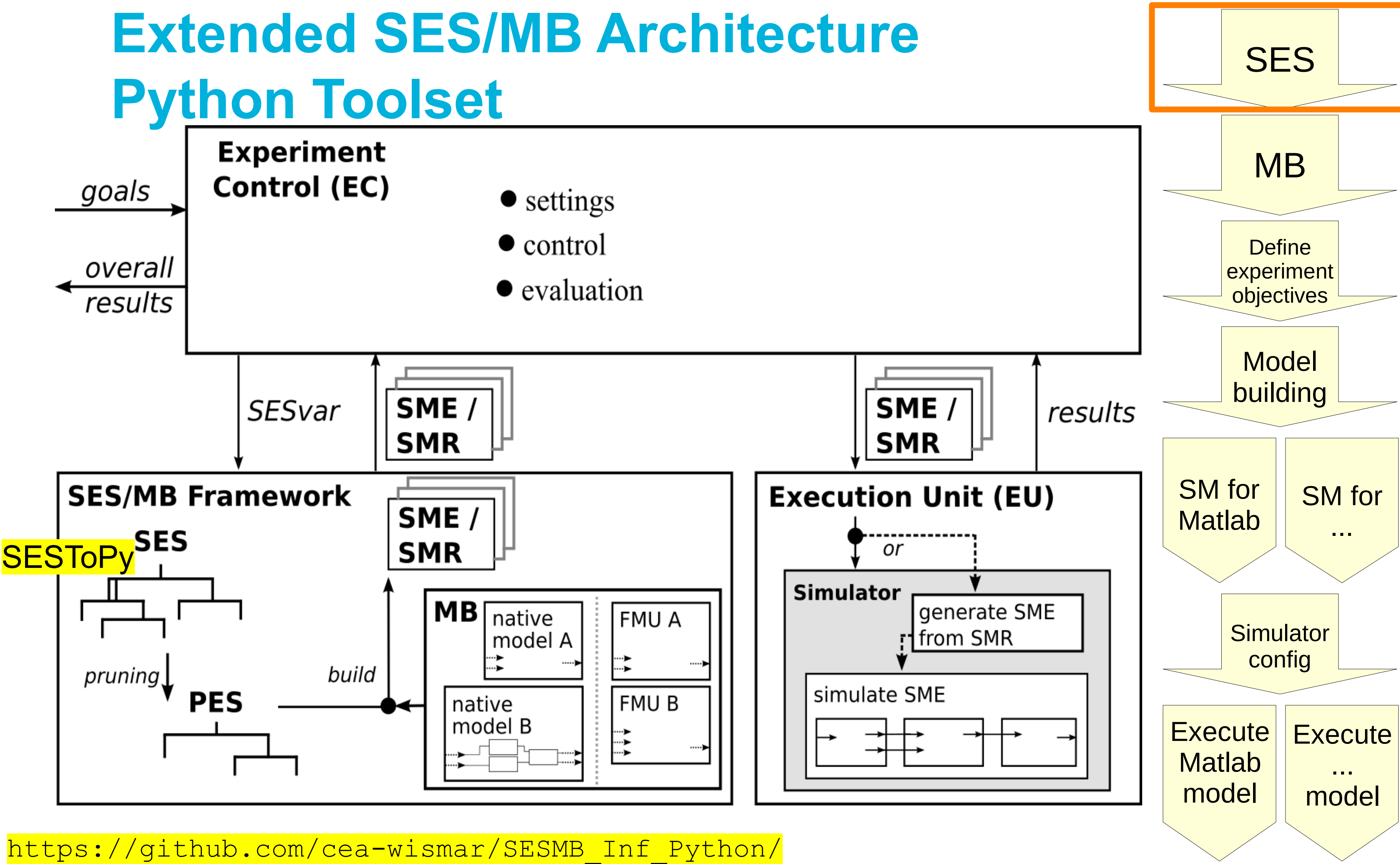
Source: Pawletta, T. and Folkerts, H. (2023). Model Behavior Generation for Multiple Simulators (Section 18.6). InBook: SCS MSBoK Guide - Body of Knowledge of Modeling and Simulation Guide. T. Ören, B.P. Zeigler, A. Tolk (Eds.).

Extended SES/MB Architecture Python Toolset



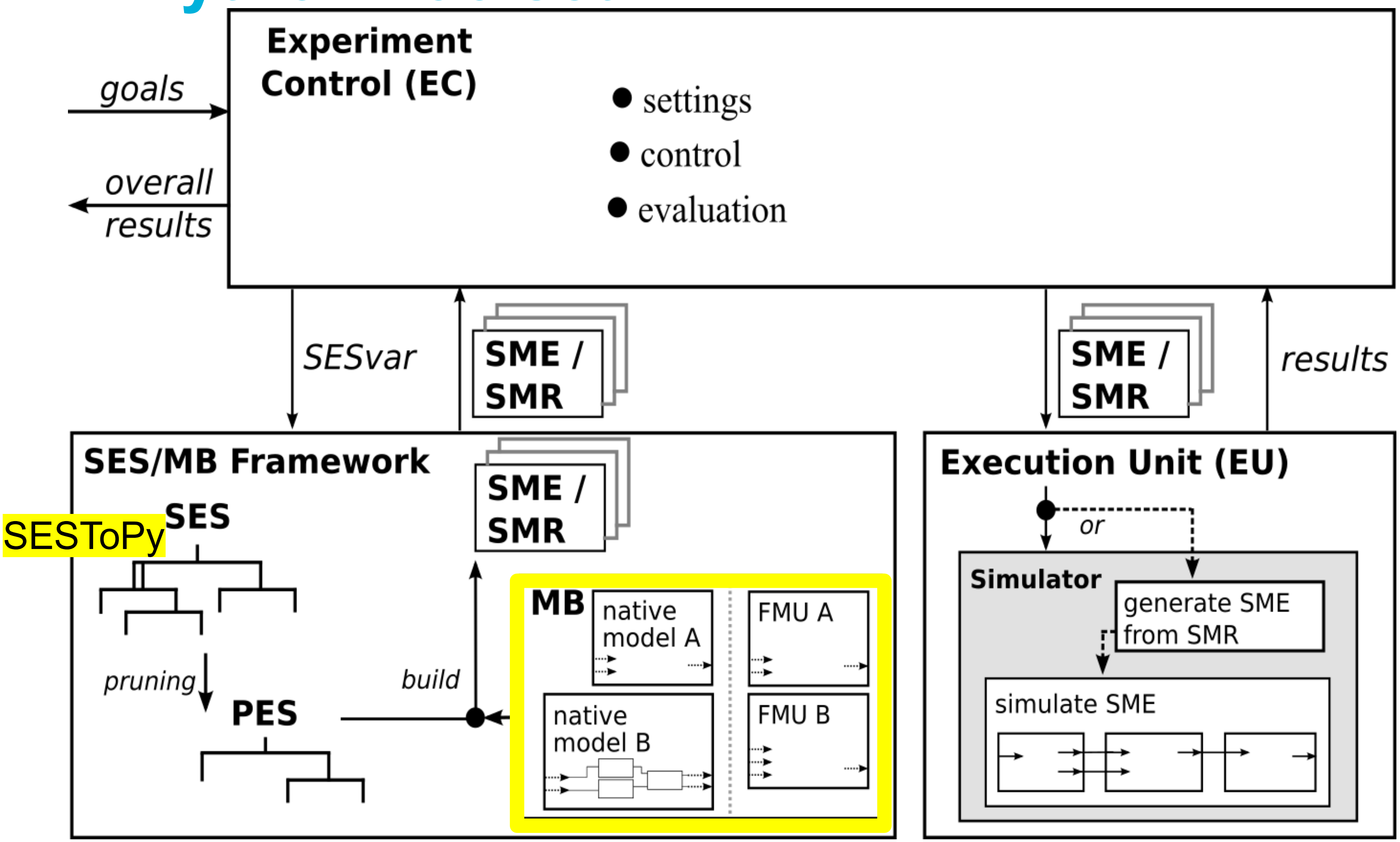
https://github.com/cea-wismar/SESMB_Inf_Python/

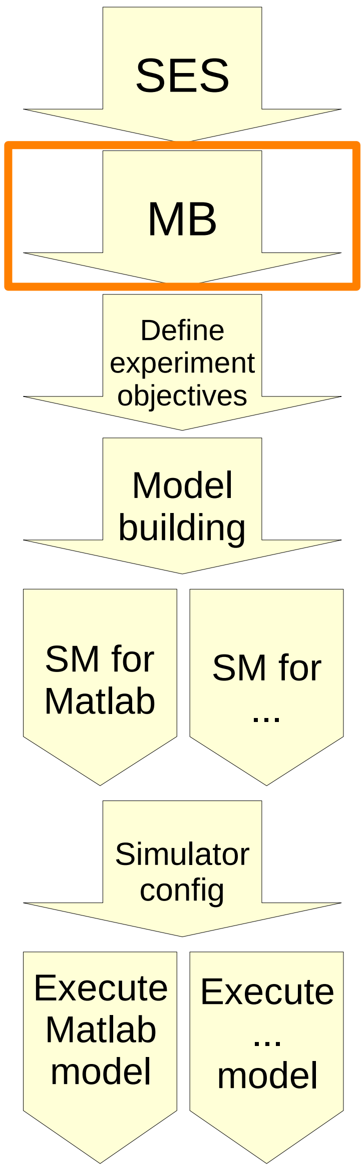
Extended SES/MB Architecture Python Toolset



https://github.com/cea-wismar/SESMB_Inf_Python/

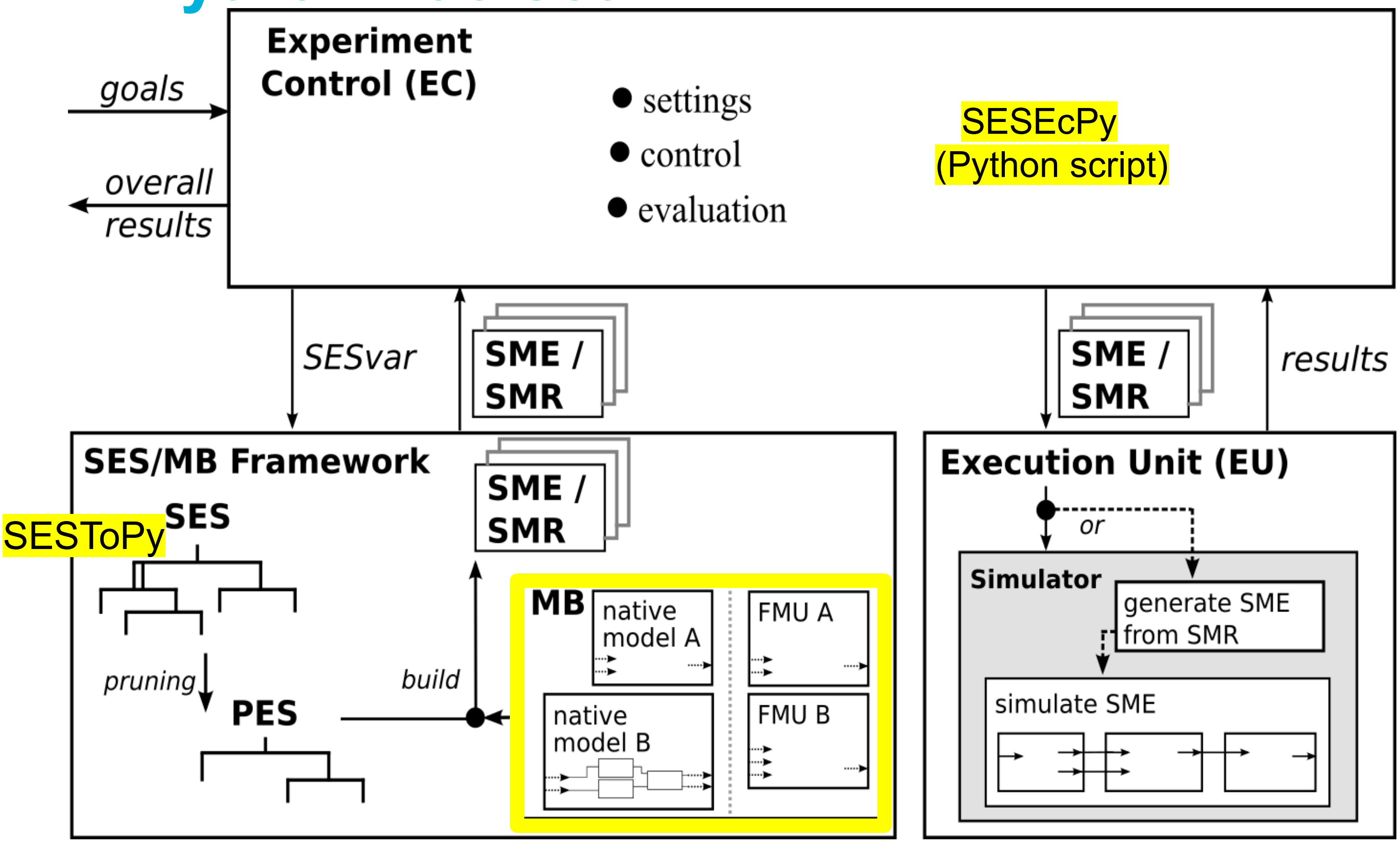
Extended SES/MB Architecture Python Toolset

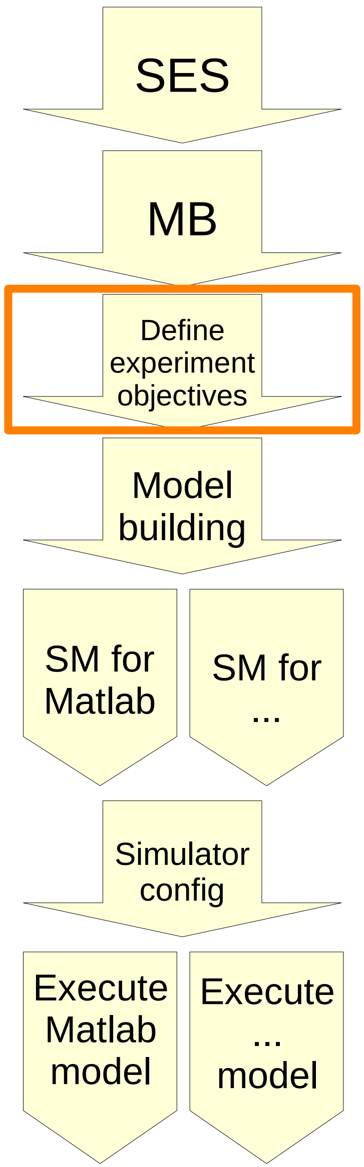




https://github.com/cea-wismar/SESMB_Inf_Python/

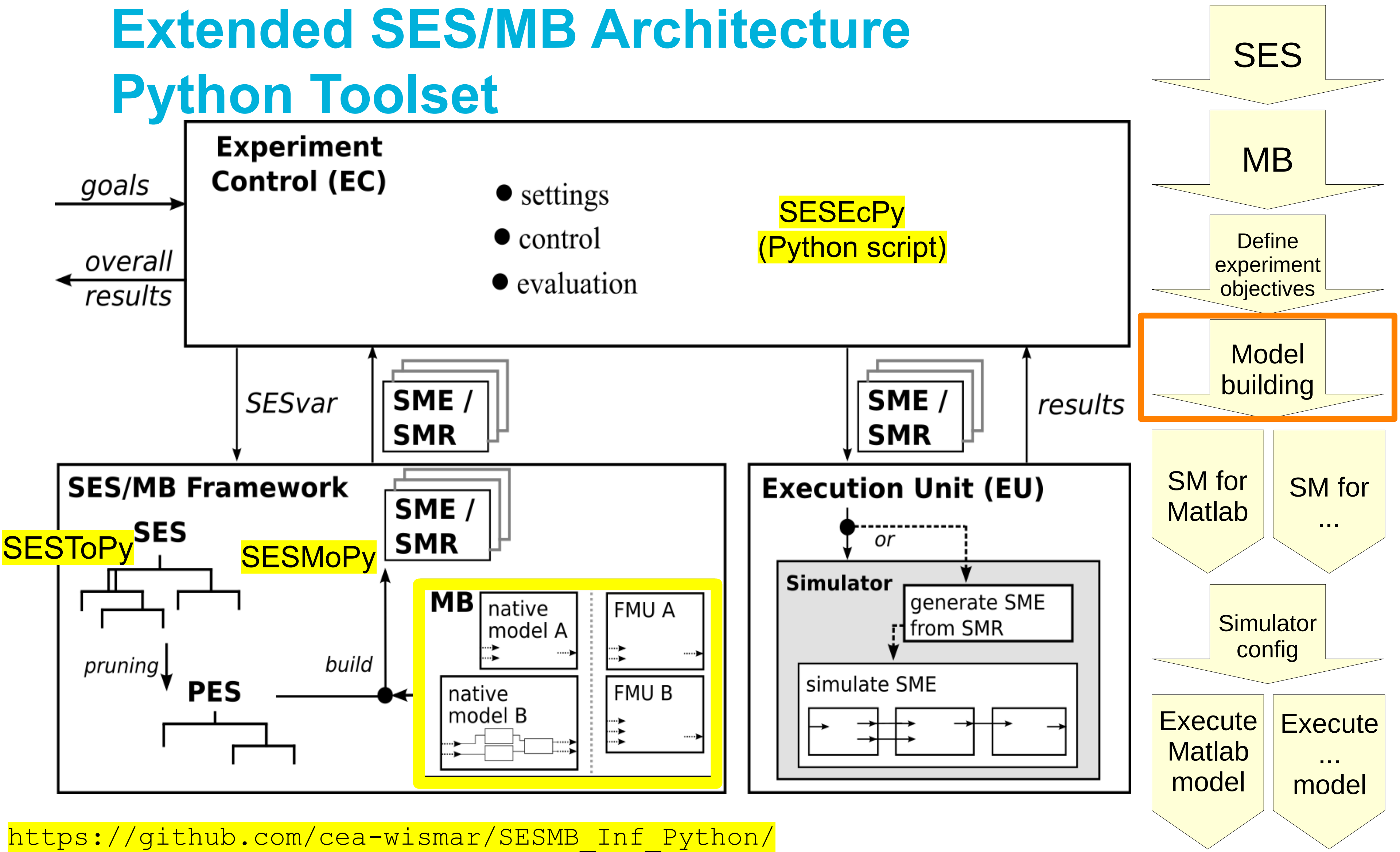
Extended SES/MB Architecture Python Toolset





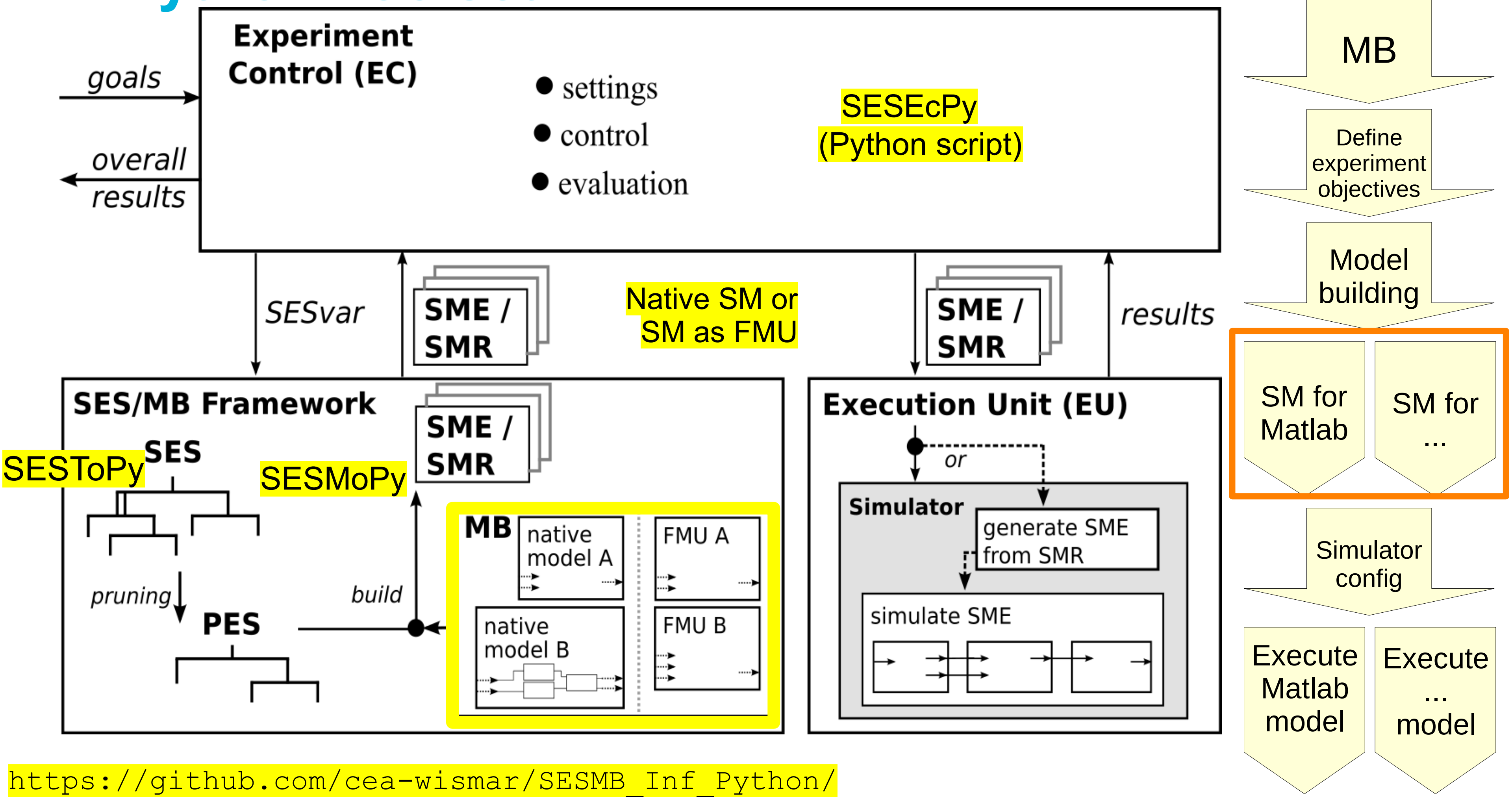
https://github.com/cea-wismar/SESMB_Inf_Python/

Extended SES/MB Architecture Python Toolset



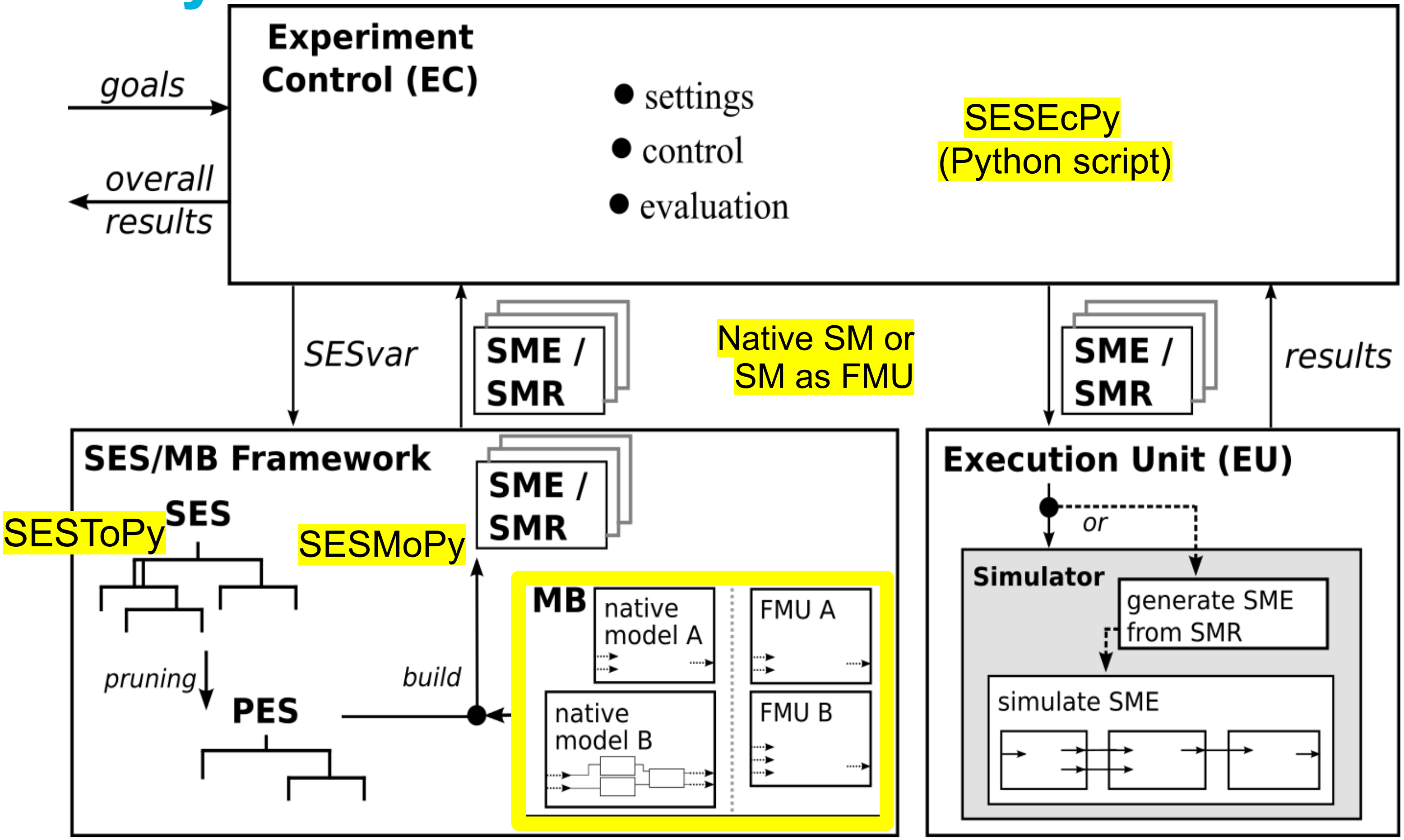
https://github.com/cea-wismar/SESMB_Inf_Python/

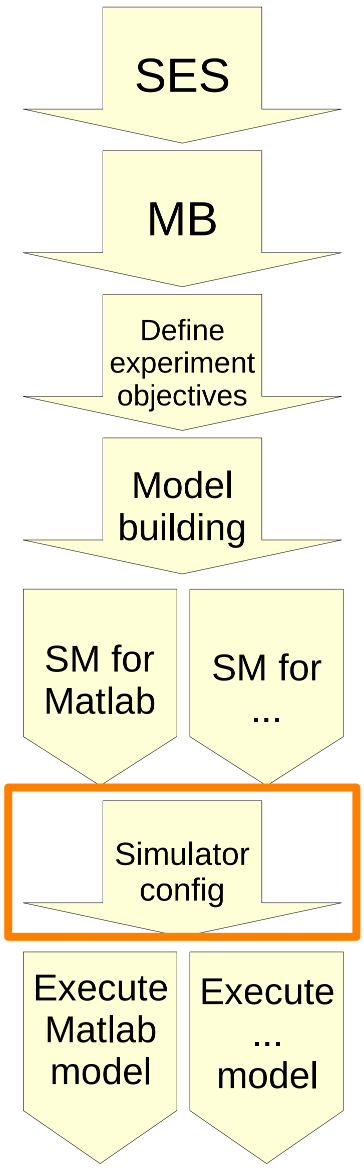
Extended SES/MB Architecture Python Toolset



https://github.com/cea-wismar/SESMB_Inf_Python/

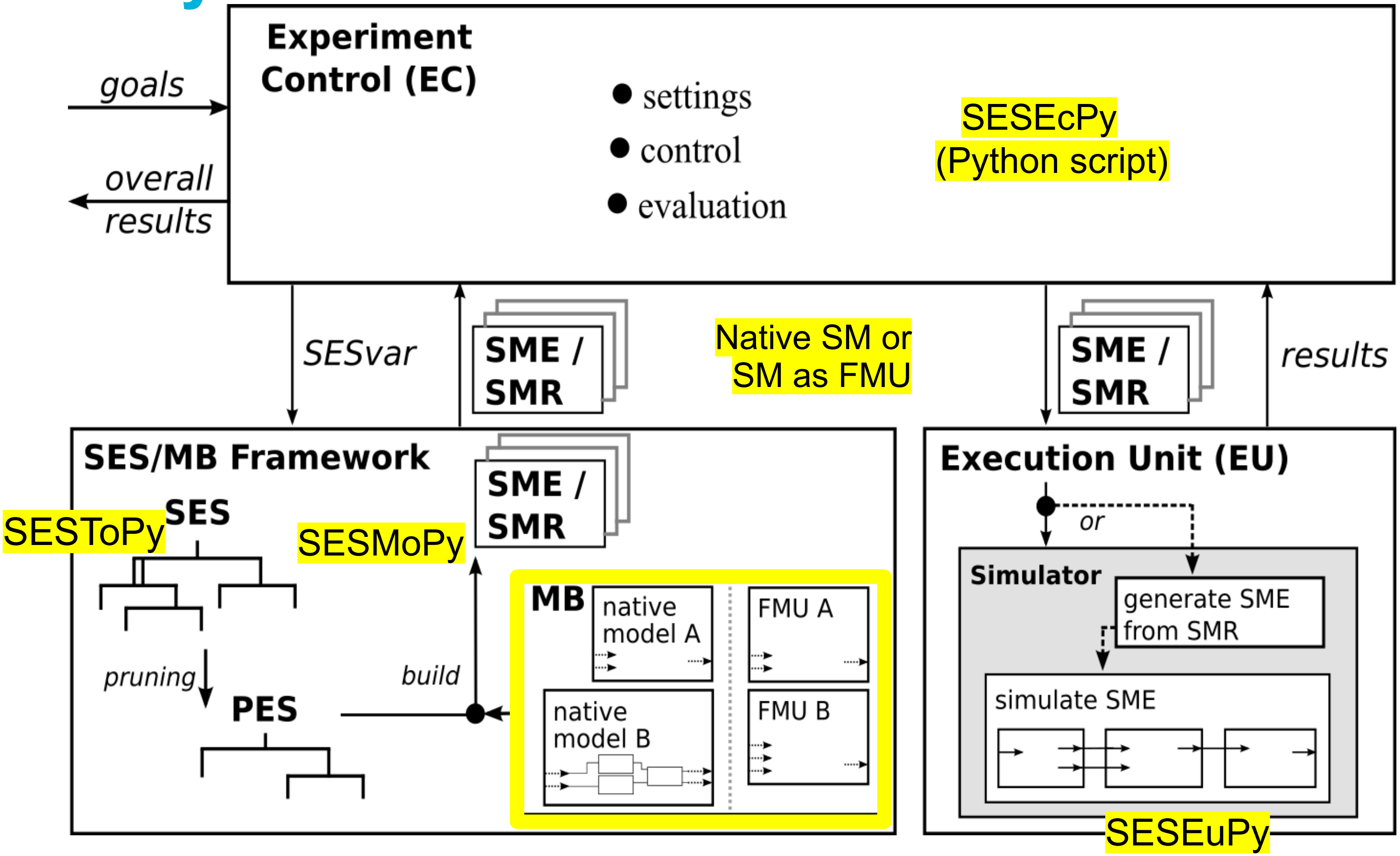
Extended SES/MB Architecture Python Toolset

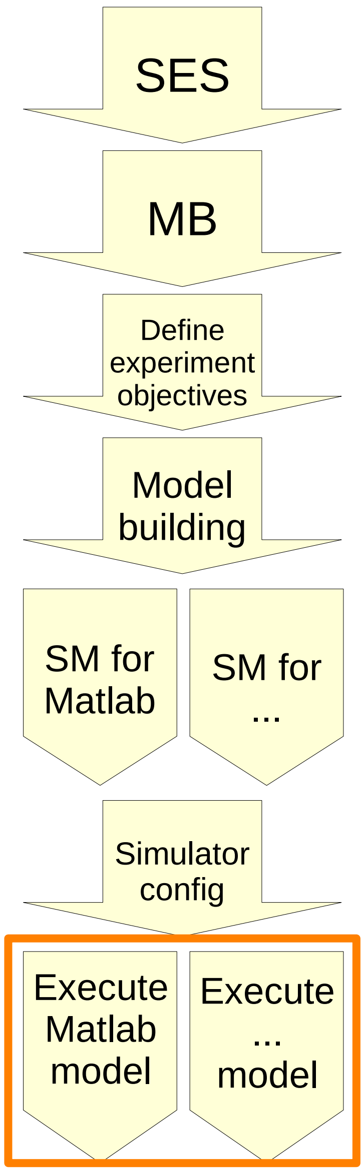




https://github.com/cea-wismar/SESMB_Inf_Python/

Extended SES/MB Architecture Python Toolset



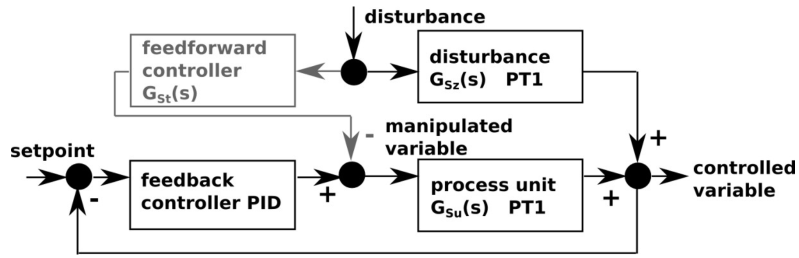


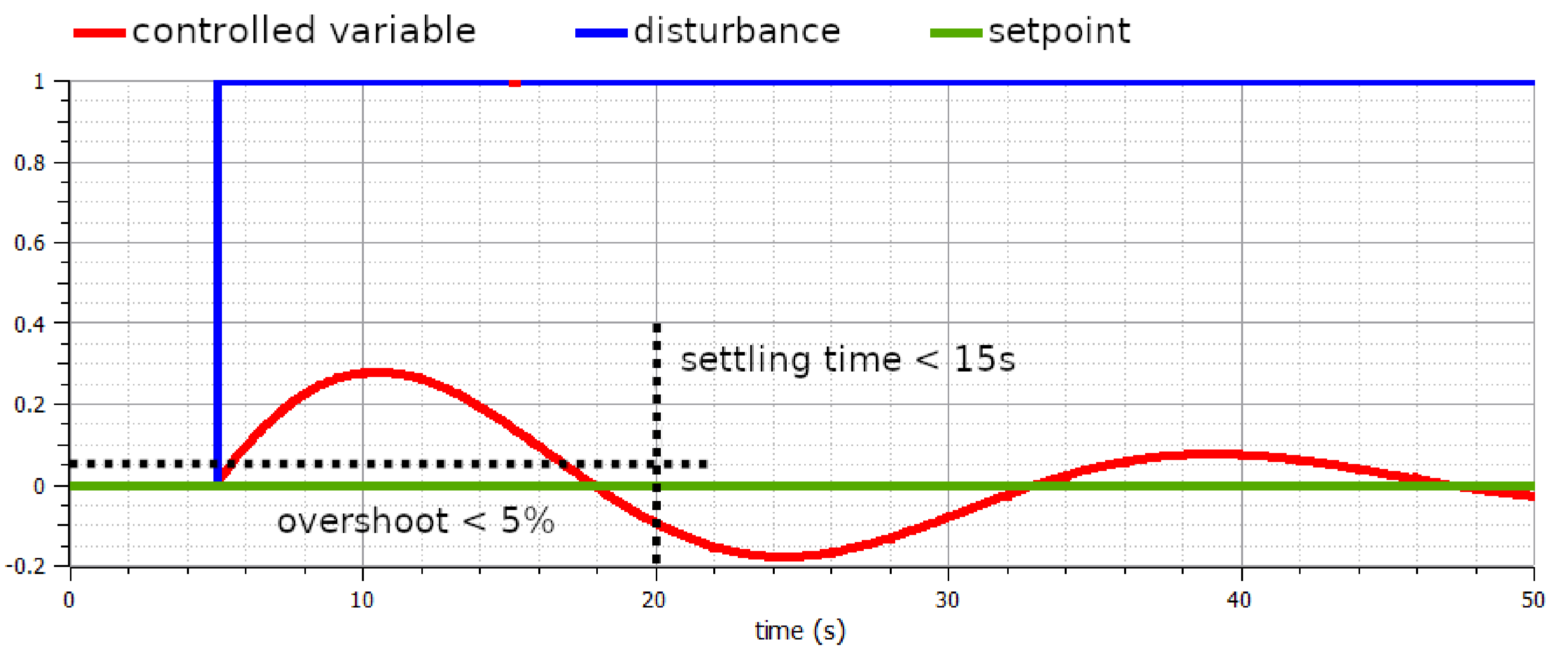
https://github.com/cea-wismar/SESMB_Inf_Python/

(wrapper for a simulator)

Automation of Case Study: Control Goals

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





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**

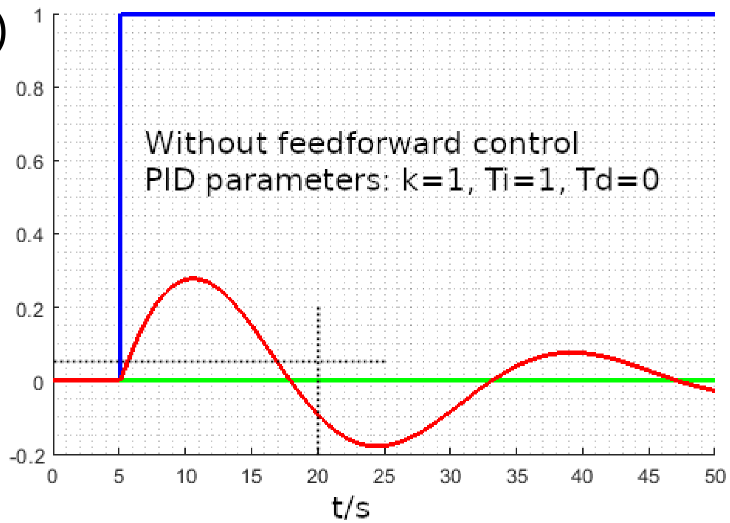
  - **Try without a feedforward control:**
    - `feedforward=0` simulate with PID: `k=1, Ti=1, Td=0`
    - `feedforward=0` simulate with PID: `k=5, Ti=0.5, Td=0`
  - **If the goals are reached with one of these configurations:**
    - Return PID configuration as overall result
  - **Else try with a feedforward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

--

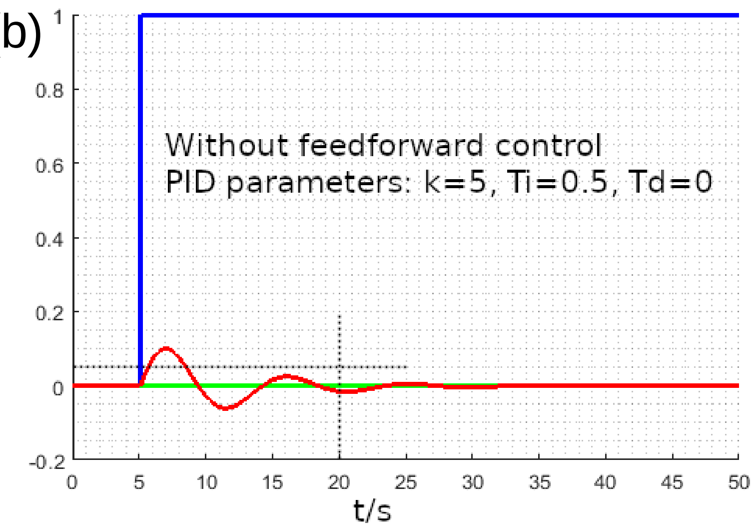
Starting over with another simulator possible (model by model validation)

Case Study: Simulation Results

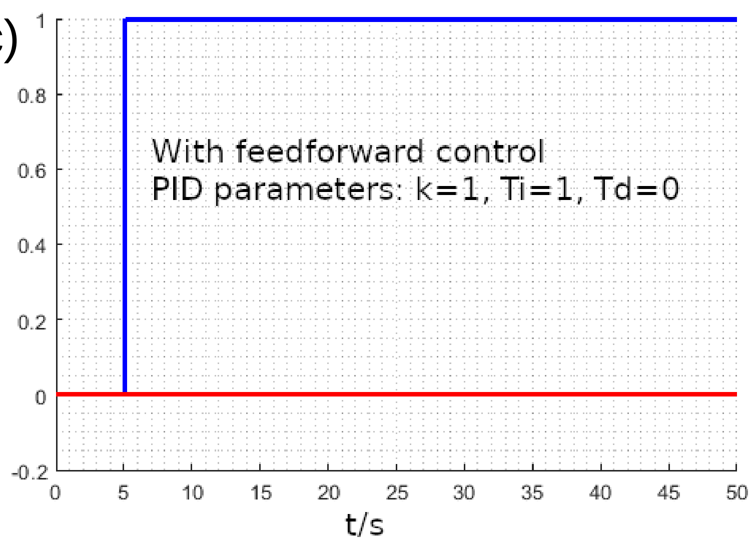
(a)



(b)



(c)

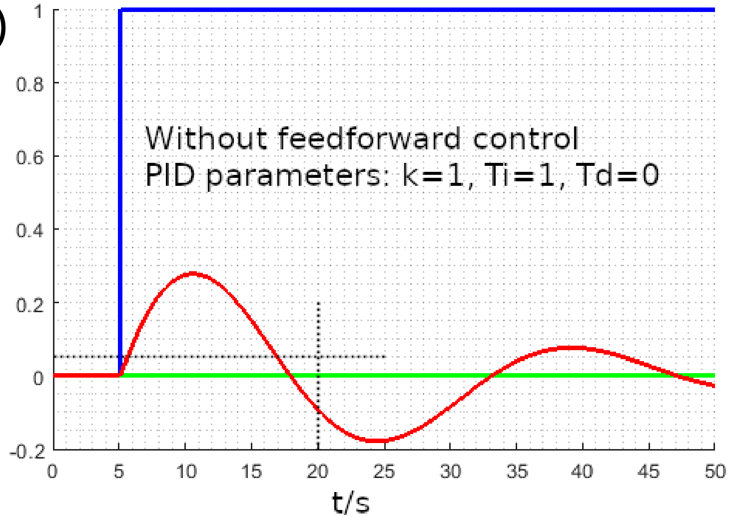


Key:
- setpoint
- disturbance
- controlled variable

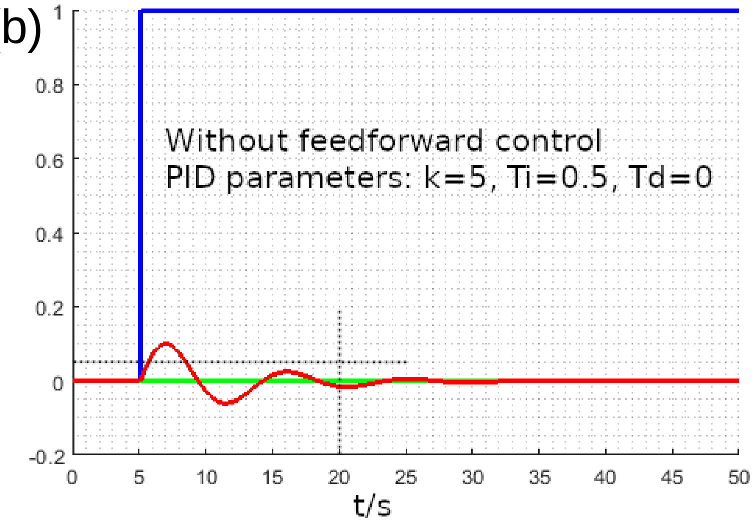
Control goals:
overshoot < 5%
settling time < 15s

Case Study: Simulation Results

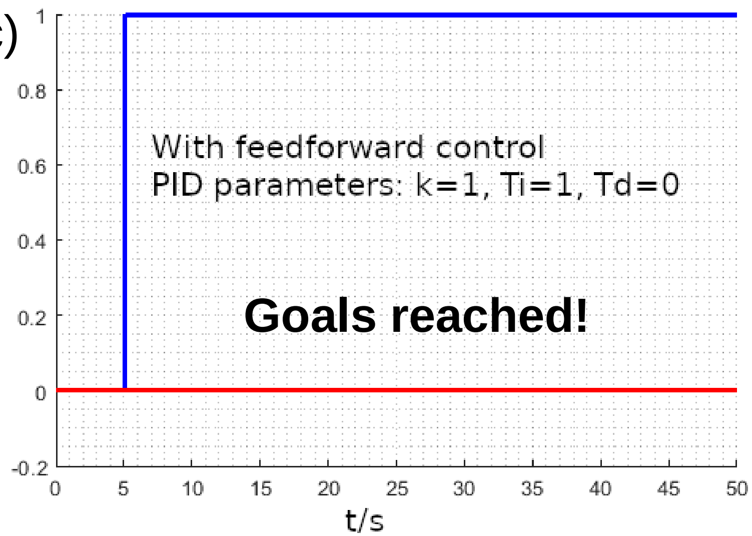
(a)



(b)



(c)



Key:
- setpoint
- disturbance
- controlled variable

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

Outline

1. Case study
2. Implementation of the SES and an MB
3. Model selection and model generation
4. Organization of a simulator-independent MB
5. Full automation of simulation experiments
6. **Summary**

Summary

Summary

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

Summary

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

Summary

- SES supports simulator-independent modeling of model configurations regarding model structures and parameter settings
- MBs are usually simulator-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
  - But still problems for discrete event models

Summary

- SES supports simulator-independent modeling of model configurations regarding model structures and parameter settings
- MBs are usually simulator-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
  - But still problems for discrete event models
- The Extended SES/MB Architecture supports a full experiment automation regarding defined design objectives (using multiple simulators)

Summary

- SES supports simulator-independent modeling of model configurations regarding model structures and parameter settings
- MBs are usually simulator-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
  - But still problems for discrete event models
- The Extended SES/MB Architecture supports a full experiment automation regarding defined design objectives (using multiple simulators)