

# Model Behaviour Generation for Multiple Simulators

Faculty of Engineering / Research Group CEA

Thorsten Pawletta  
Hendrik Folkerts

E-Mail:  
[thorsten.pawletta@hs-wismar.de](mailto:thorsten.pawletta@hs-wismar.de)  
[hendrik.folkerts@hs-wismar.de](mailto:hendrik.folkerts@hs-wismar.de)

Web:  
[www.hs-wismar.de](http://www.hs-wismar.de) / [www.cea-wismar.de](http://www.cea-wismar.de)





## 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

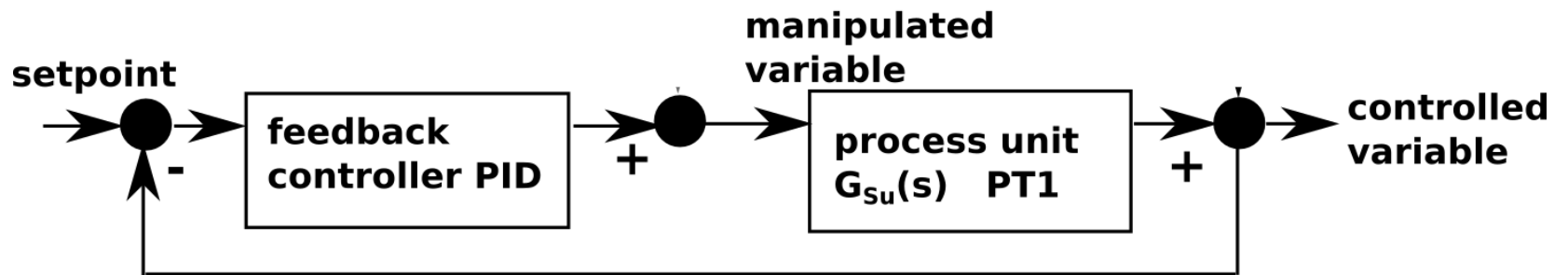


# Case Study



## Case Study

- Feedback control system

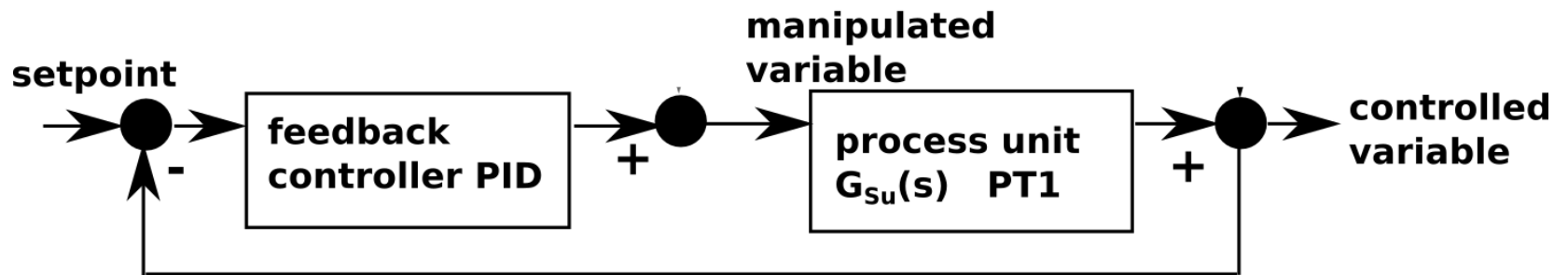




## Case Study

- Feedback control system
- Described by transfer functions

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



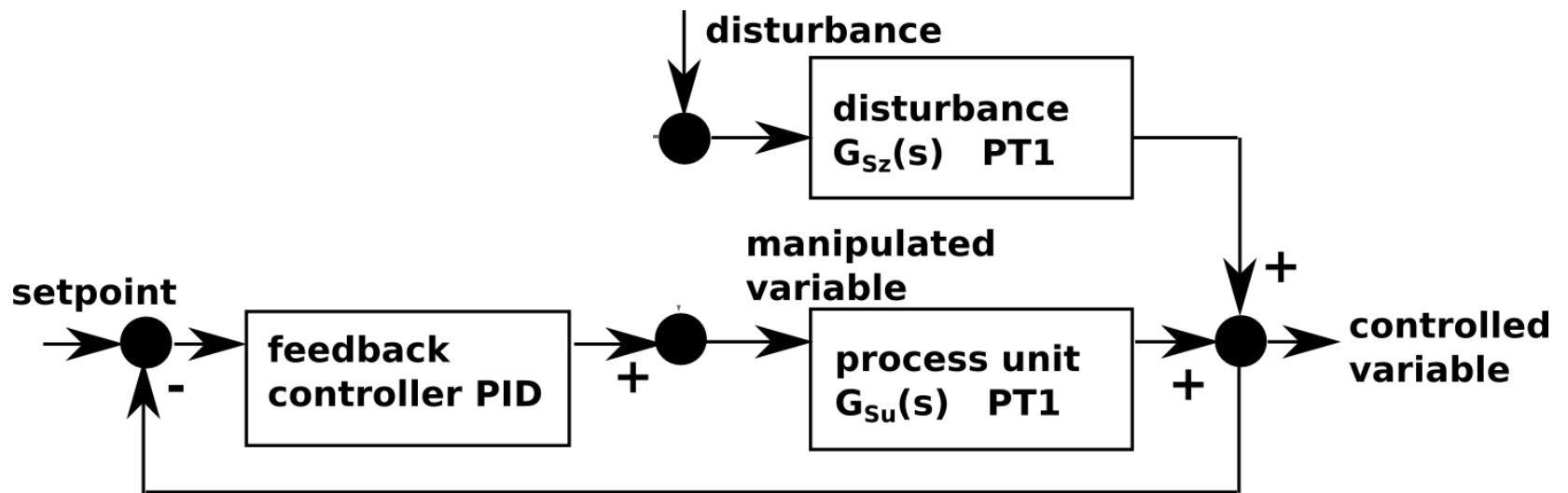


## Case Study

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

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

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







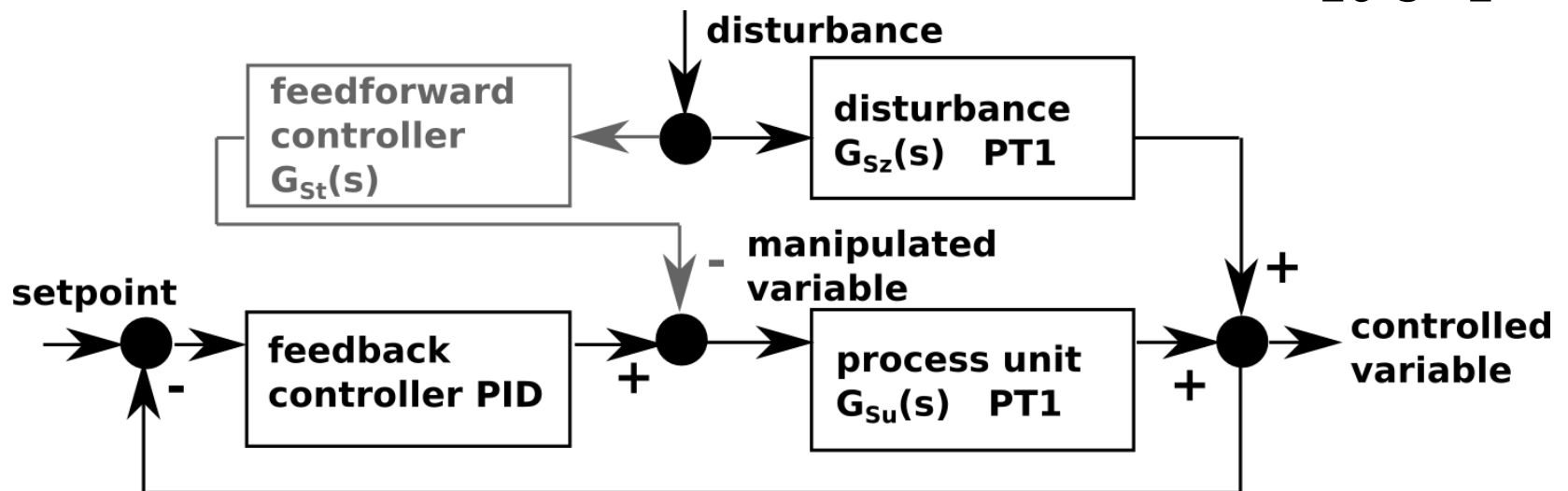
## Case Study

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

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

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

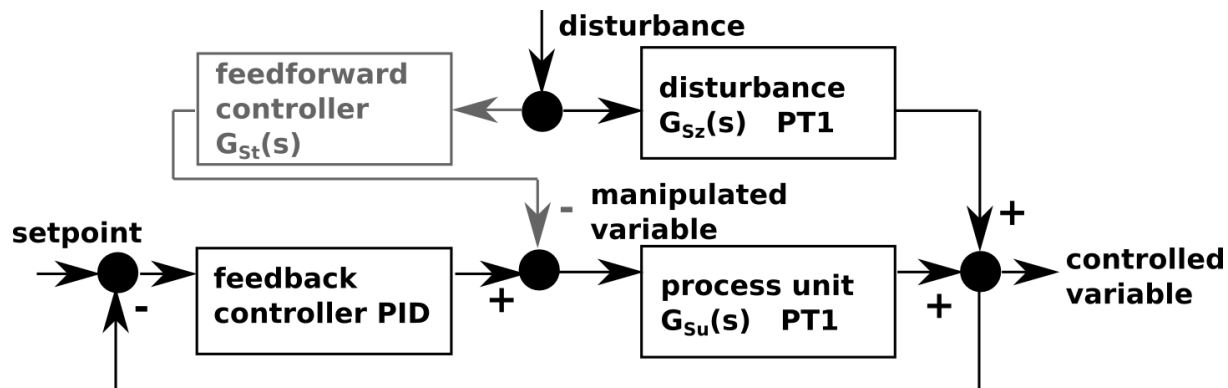
$$G_{St}(s) = \frac{20 \cdot s + 1}{10 \cdot s + 1}$$





## Case Study (2)

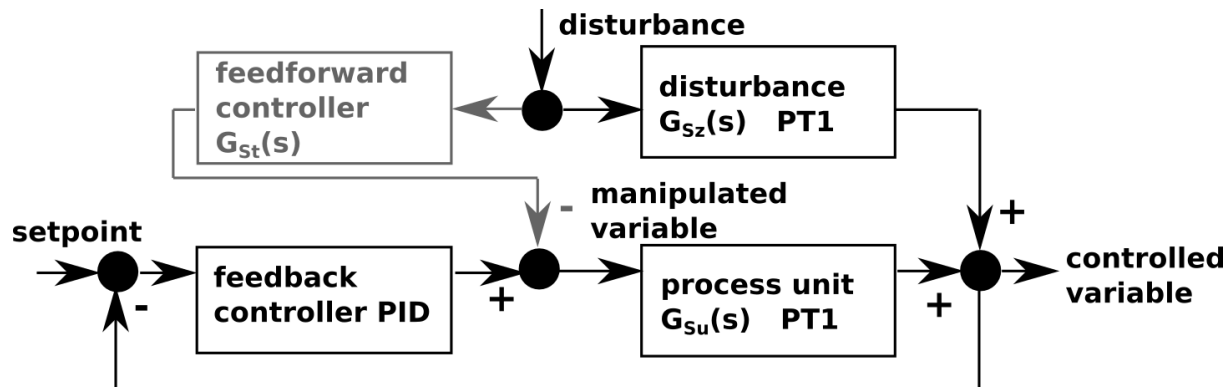
- **Two system structure variants**
  - Without feedforward control: `feedforward=0`
  - With feedforward control: `feedforward=1`





## 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)

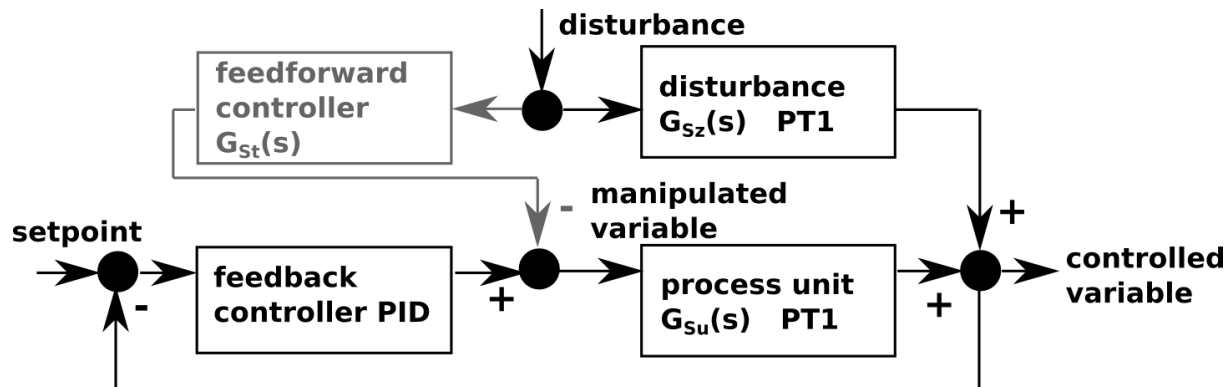




## Case Study (2)

**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)



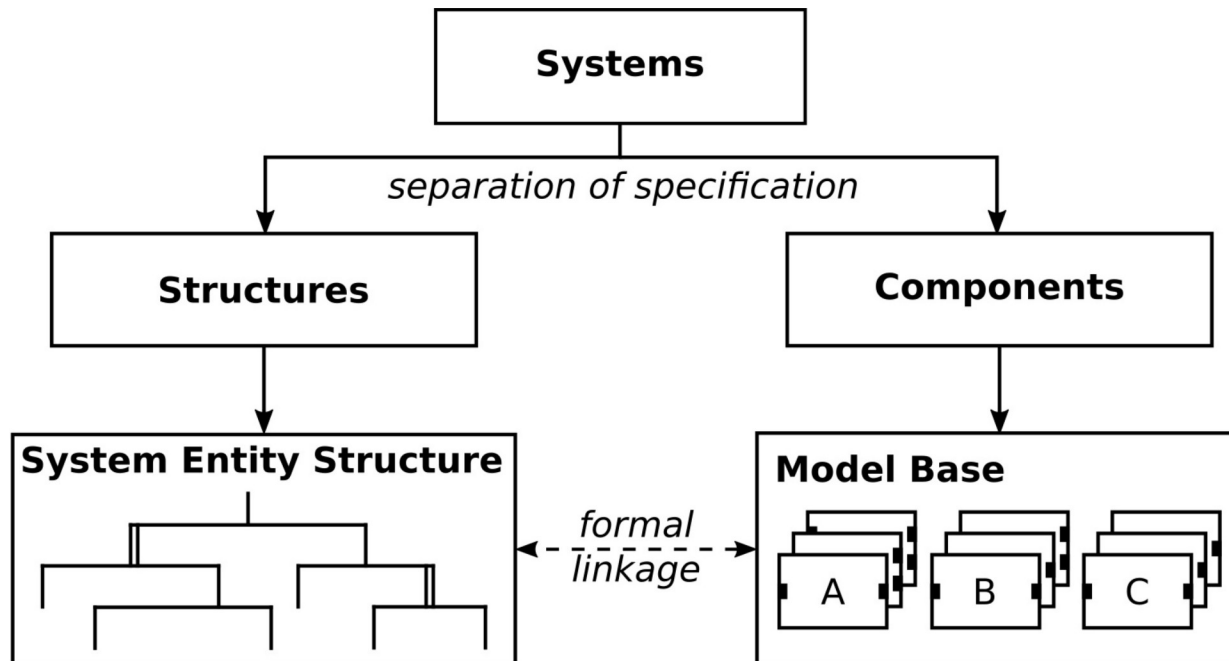


# Outline

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



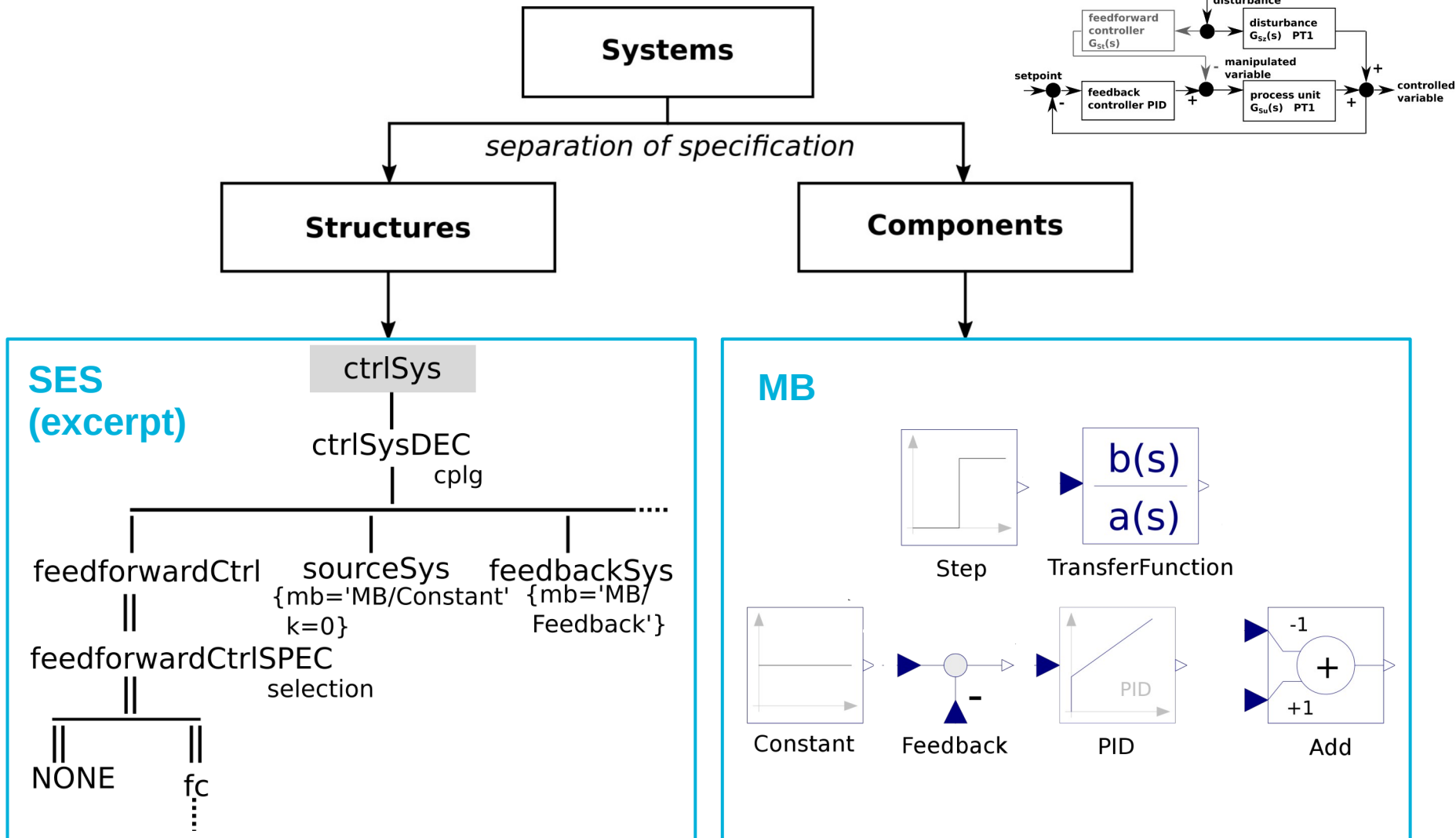
# SES/MB-based Modeling



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

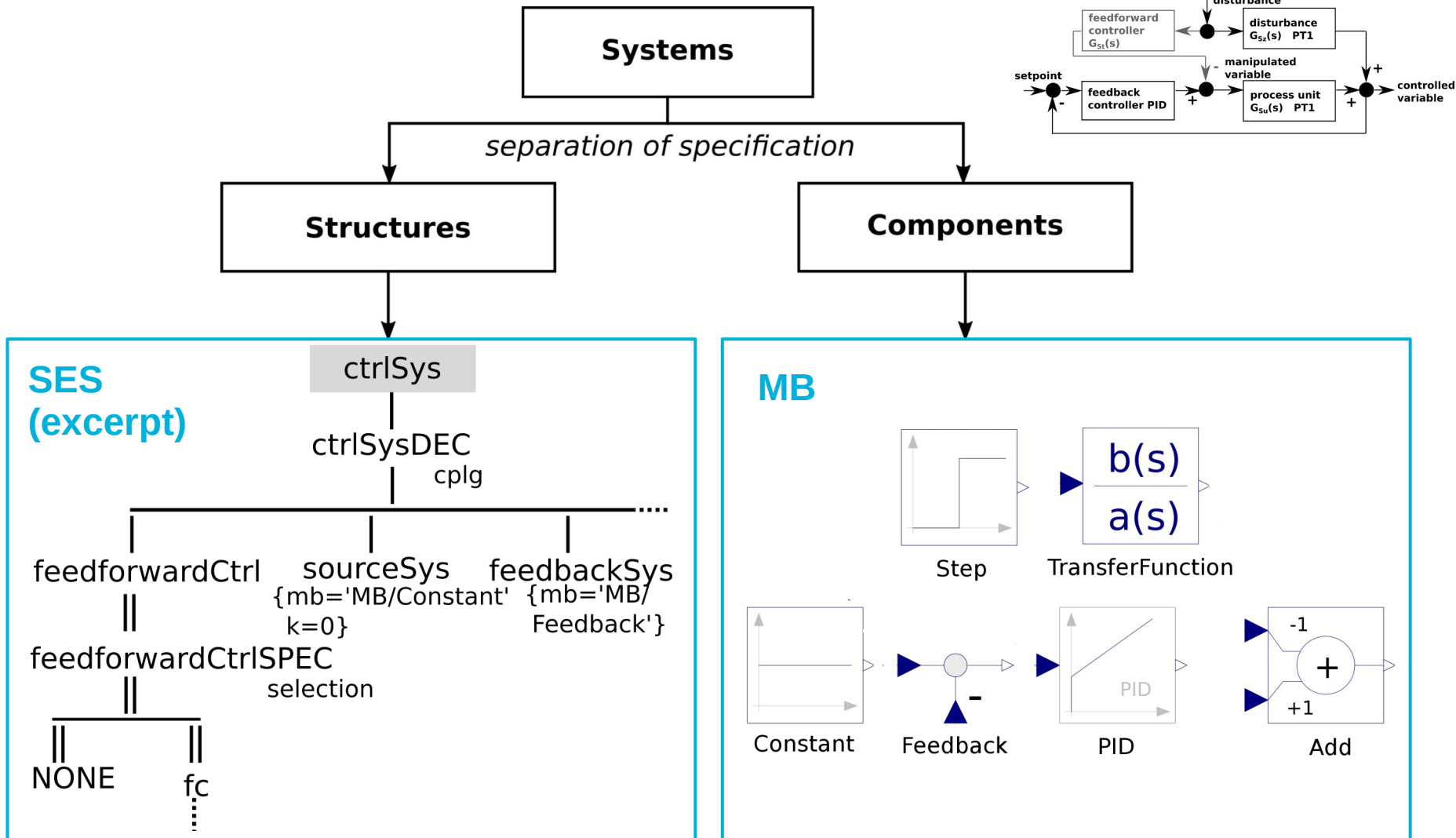


# SES/MB-based Modeling of the Case Study





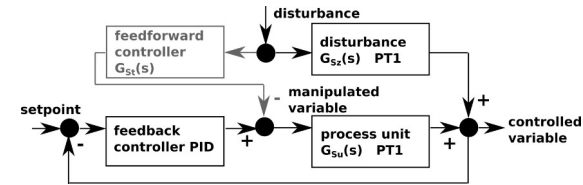
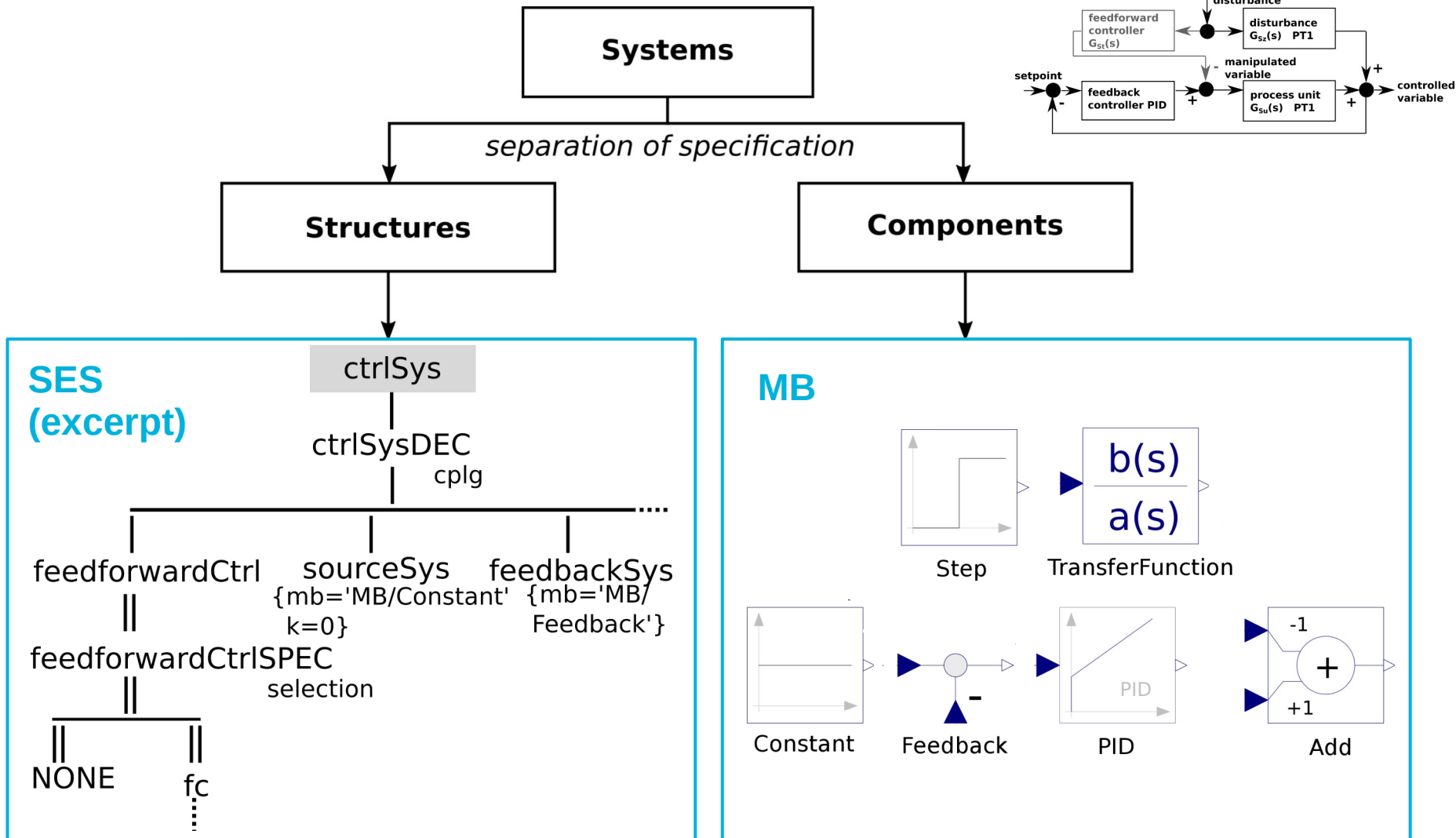
# SES/MB-based Modeling of the Case Study





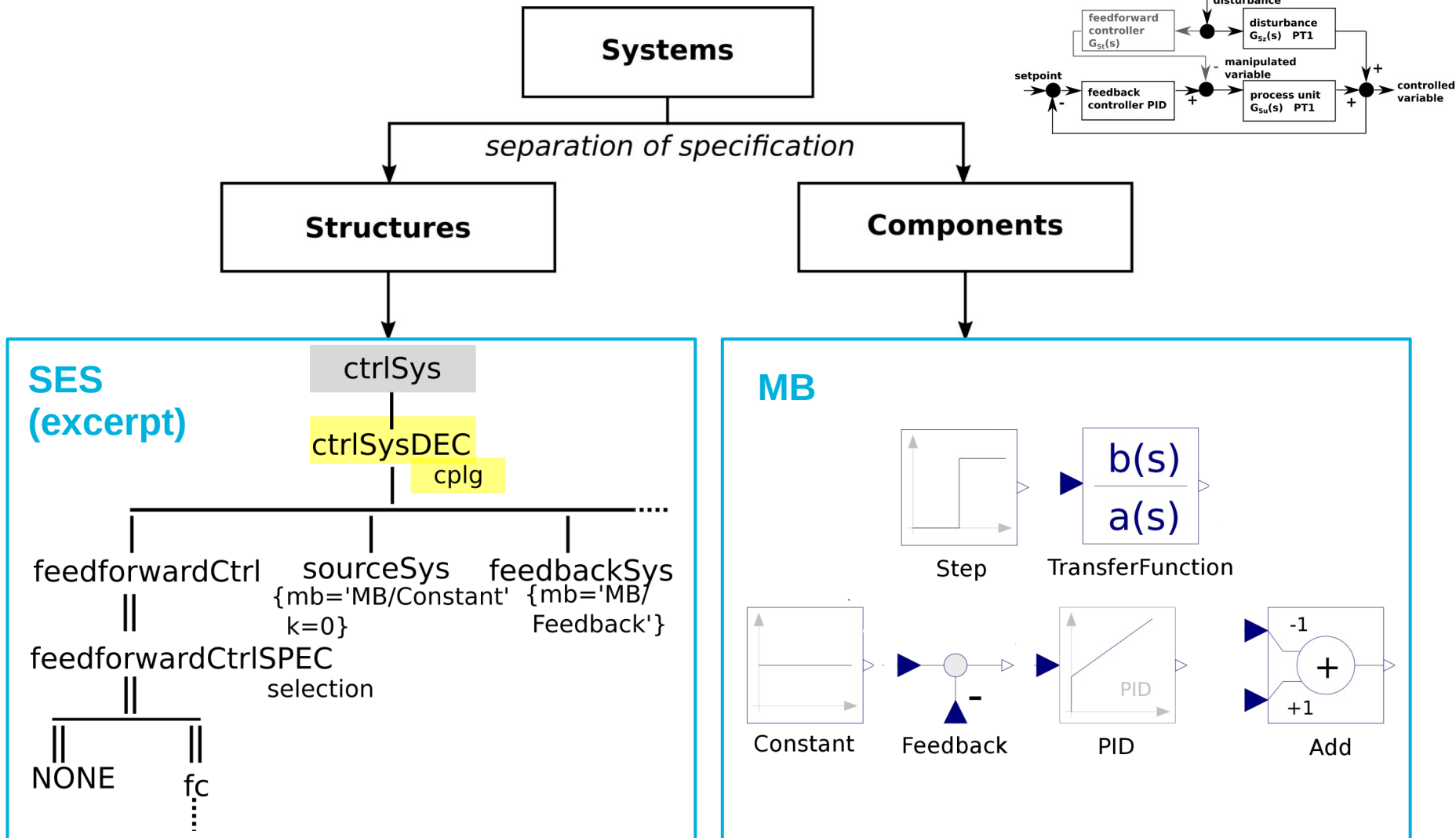


# SES/MB-based Modeling of the Case Study



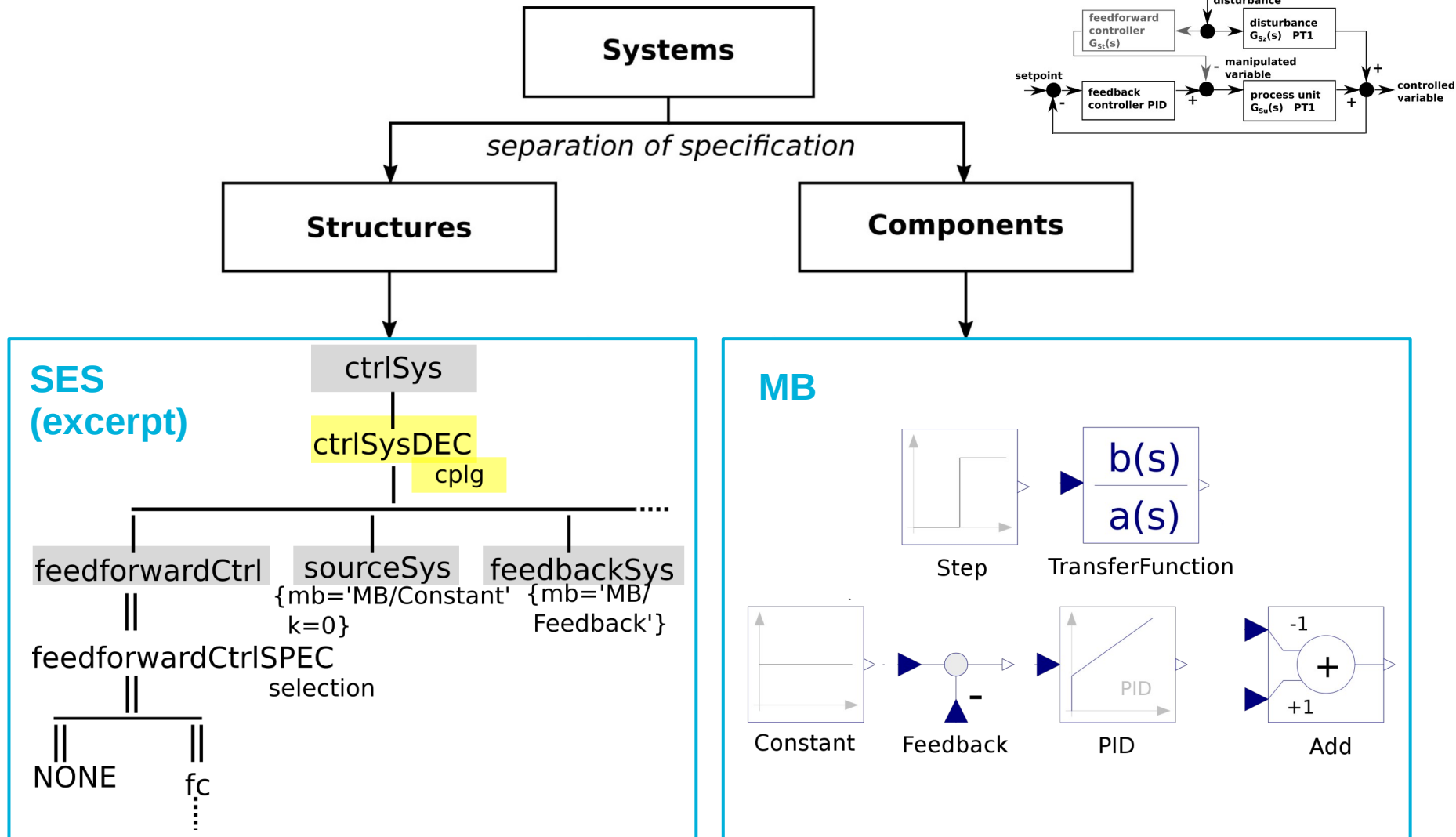


# SES/MB-based Modeling of the Case Study



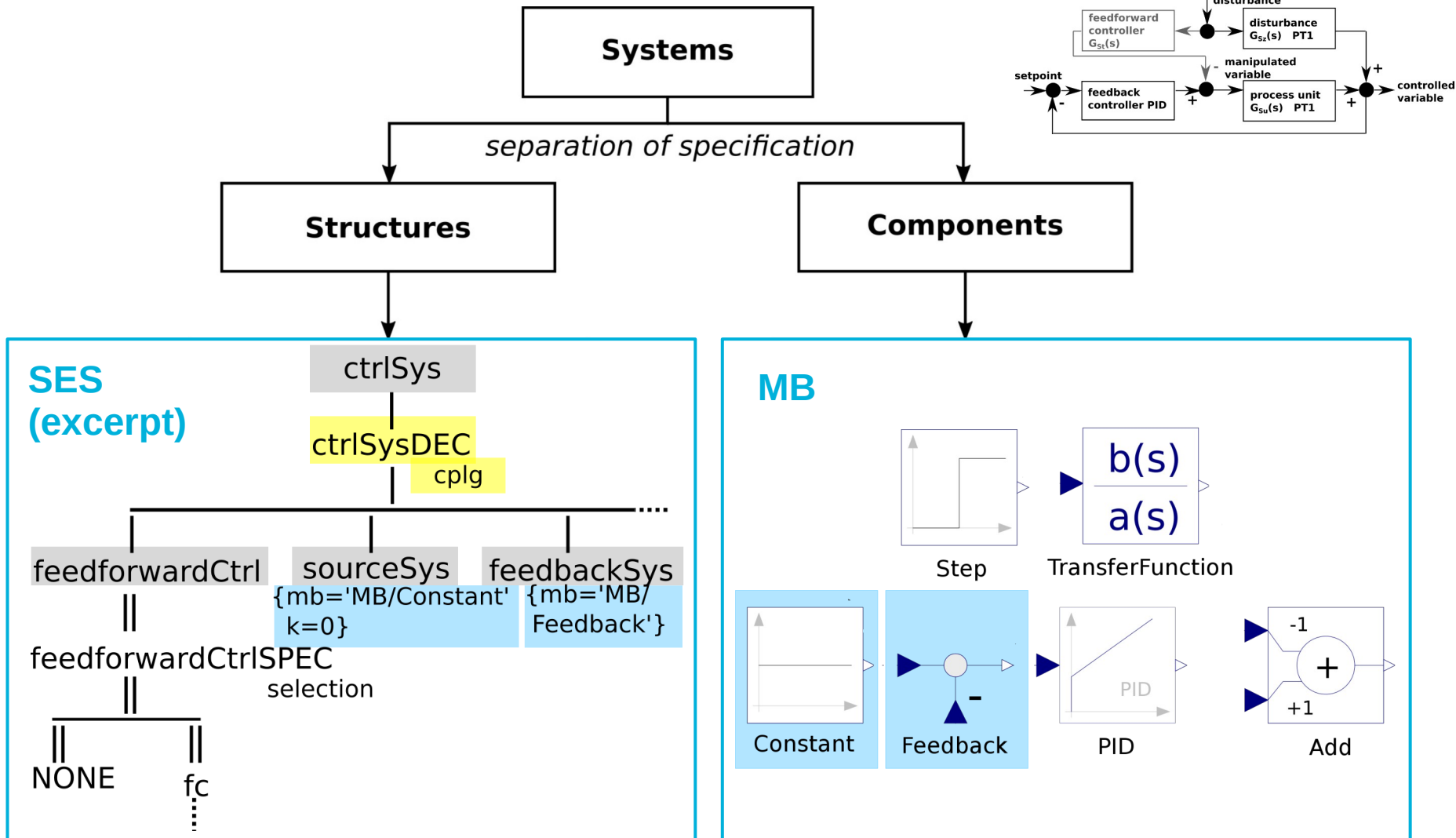


# SES/MB-based Modeling of the Case Study



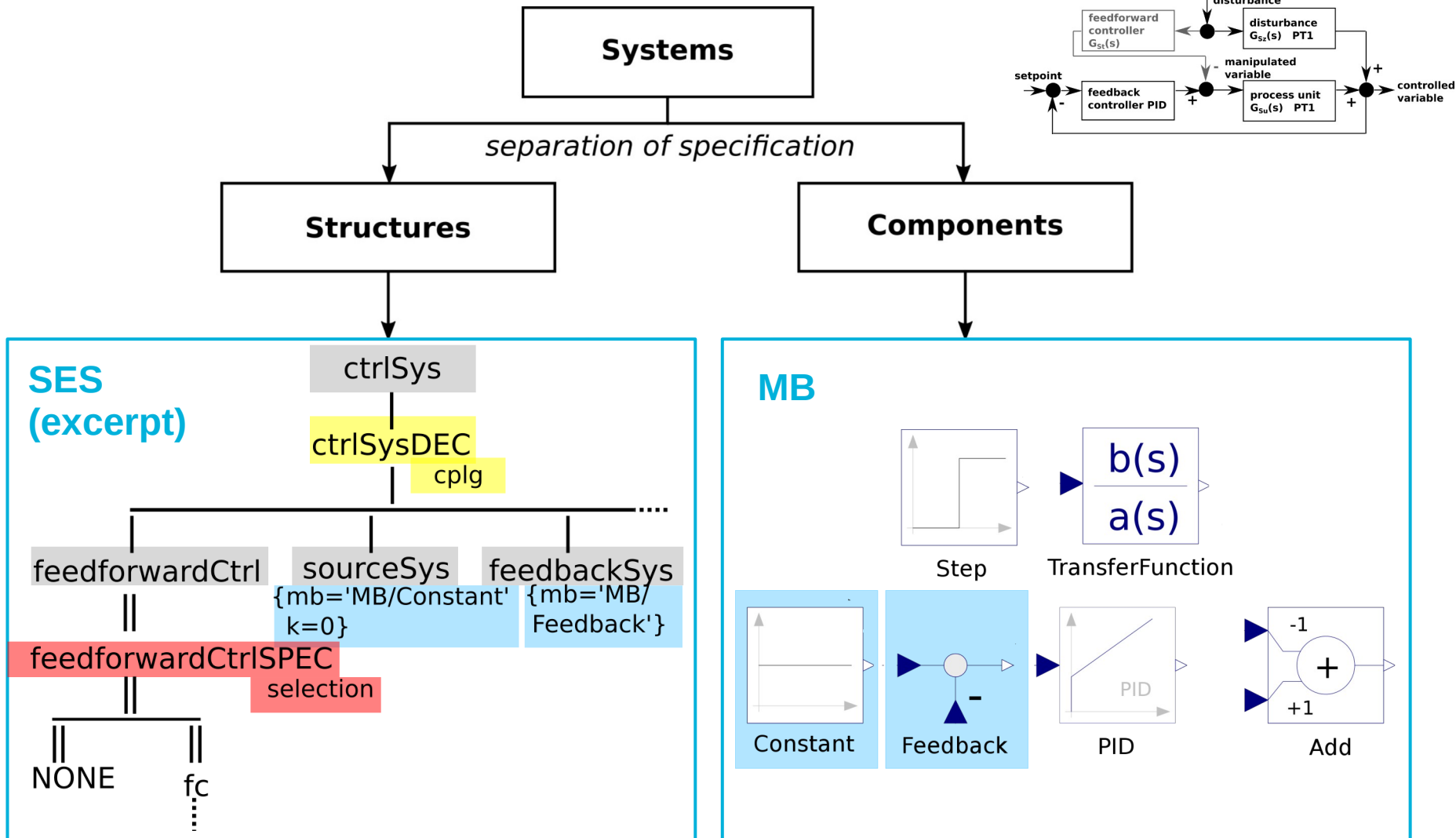


# SES/MB-based Modeling of the Case Study



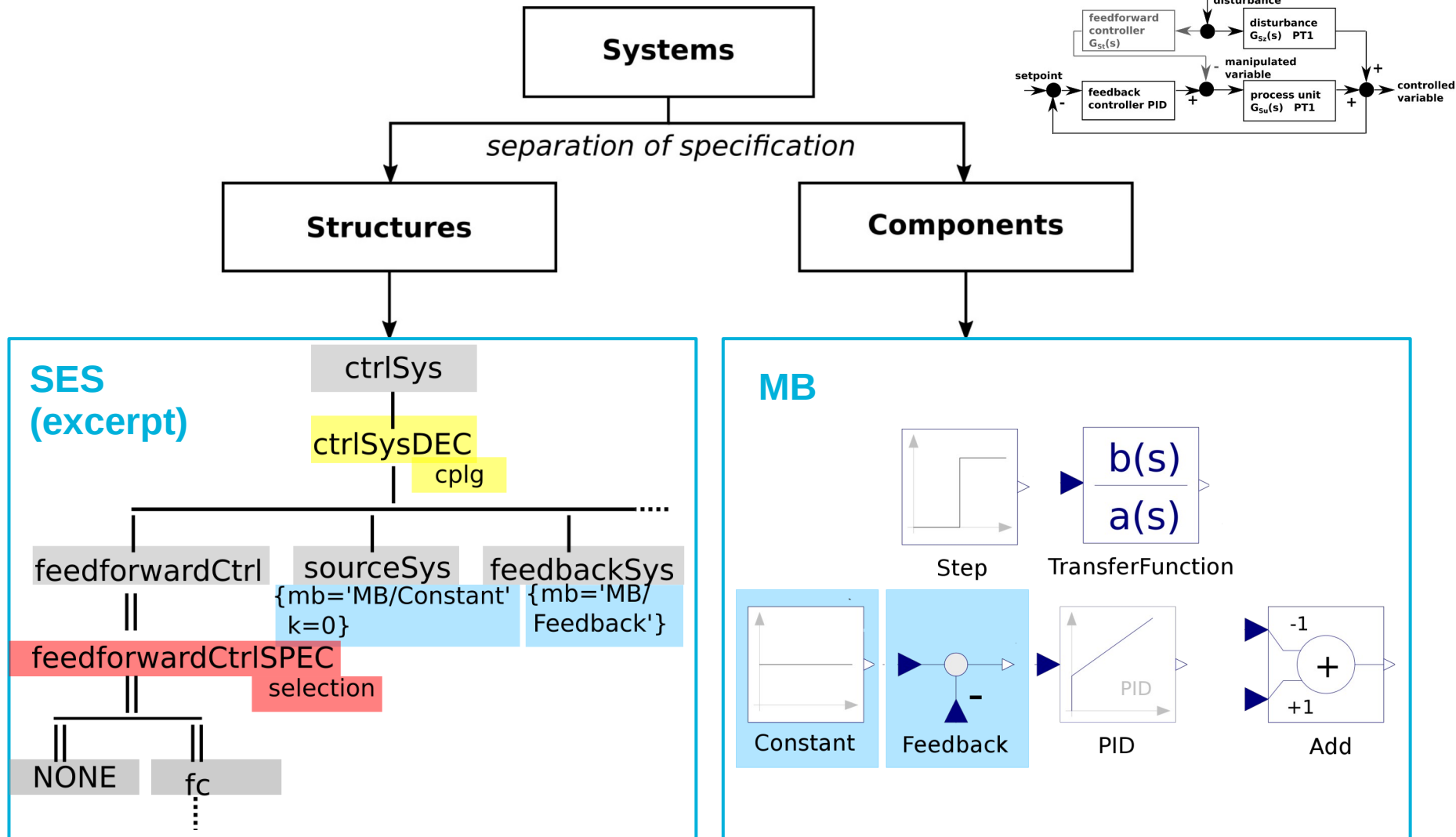


# SES/MB-based Modeling of the Case Study





# SES/MB-based Modeling of the Case Study

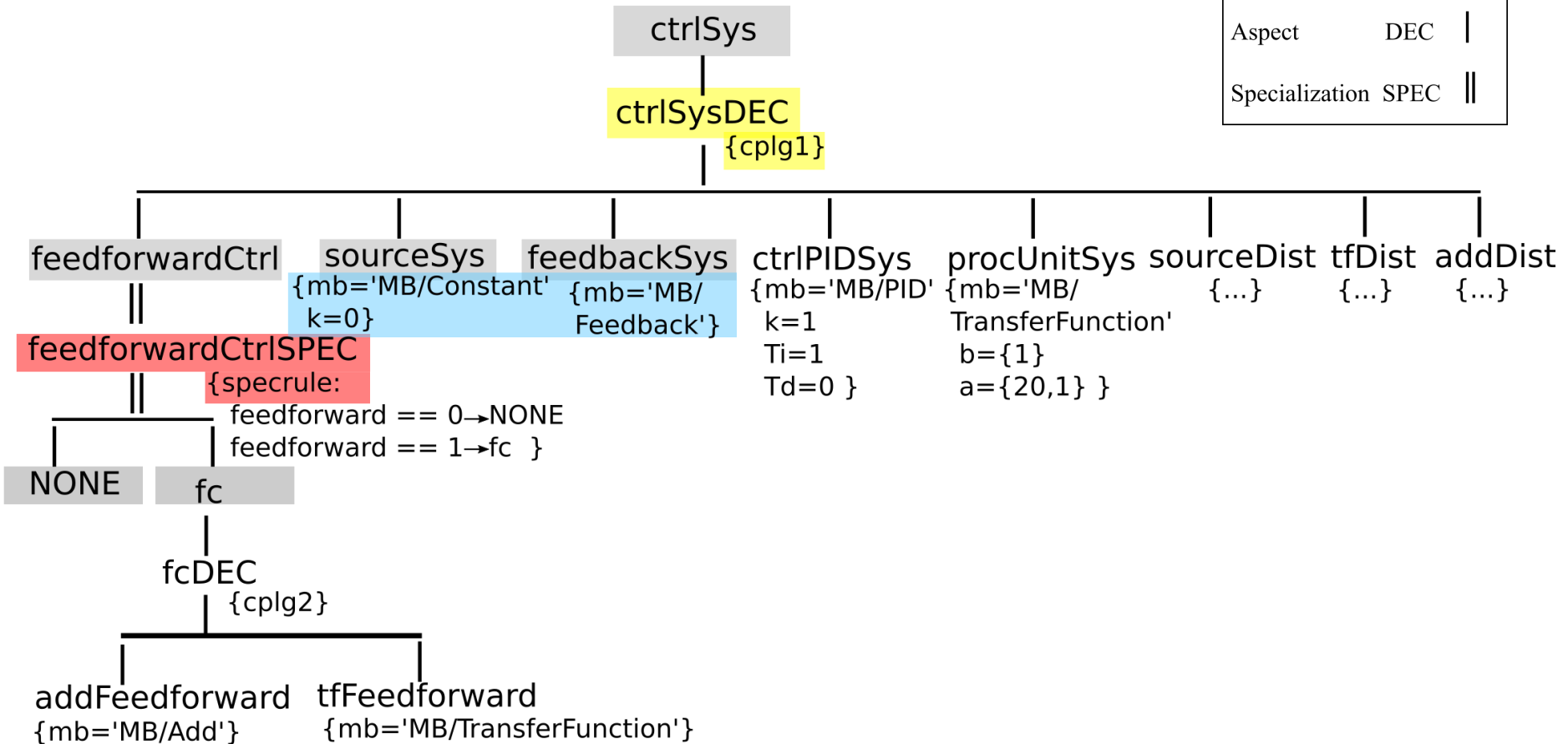




# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	

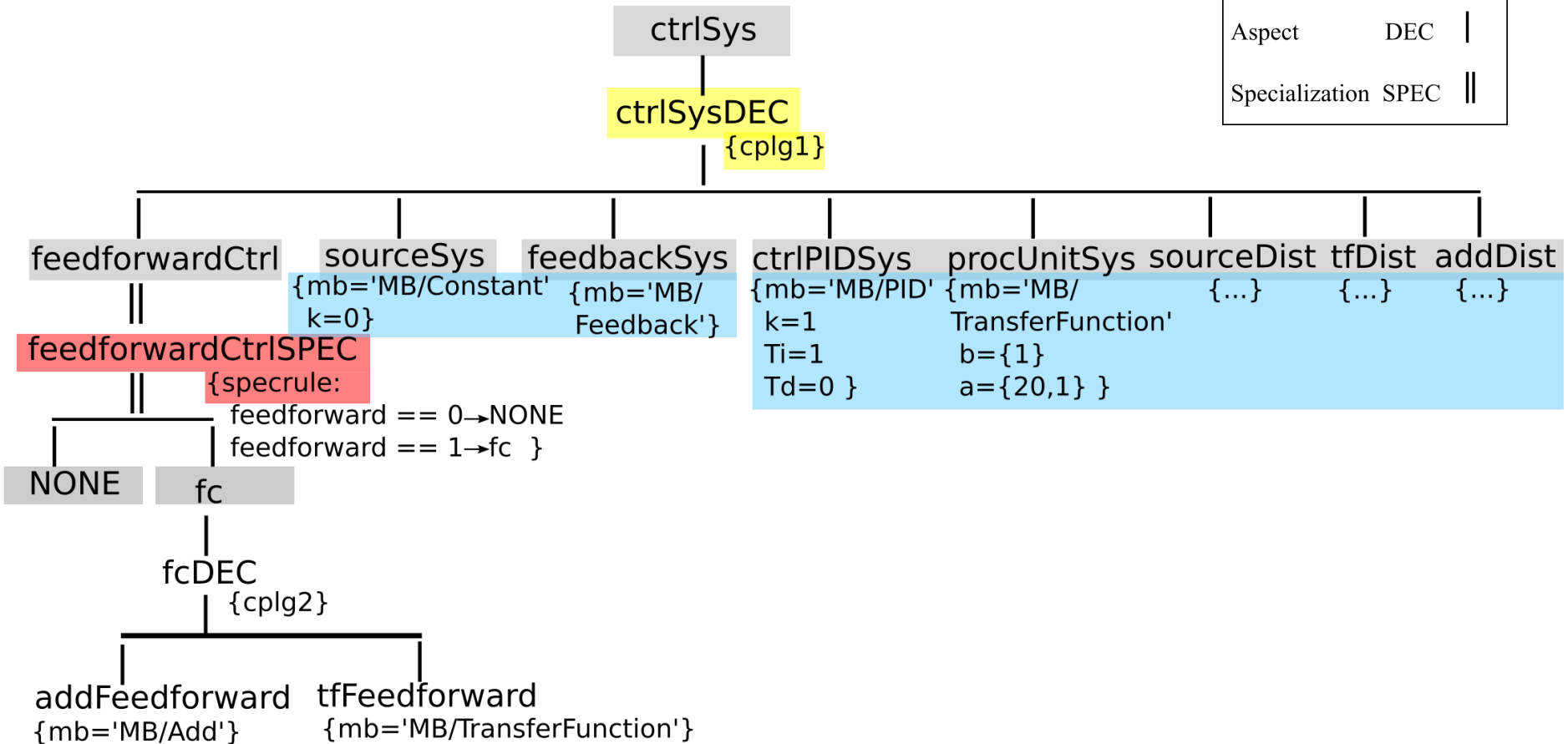




# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	



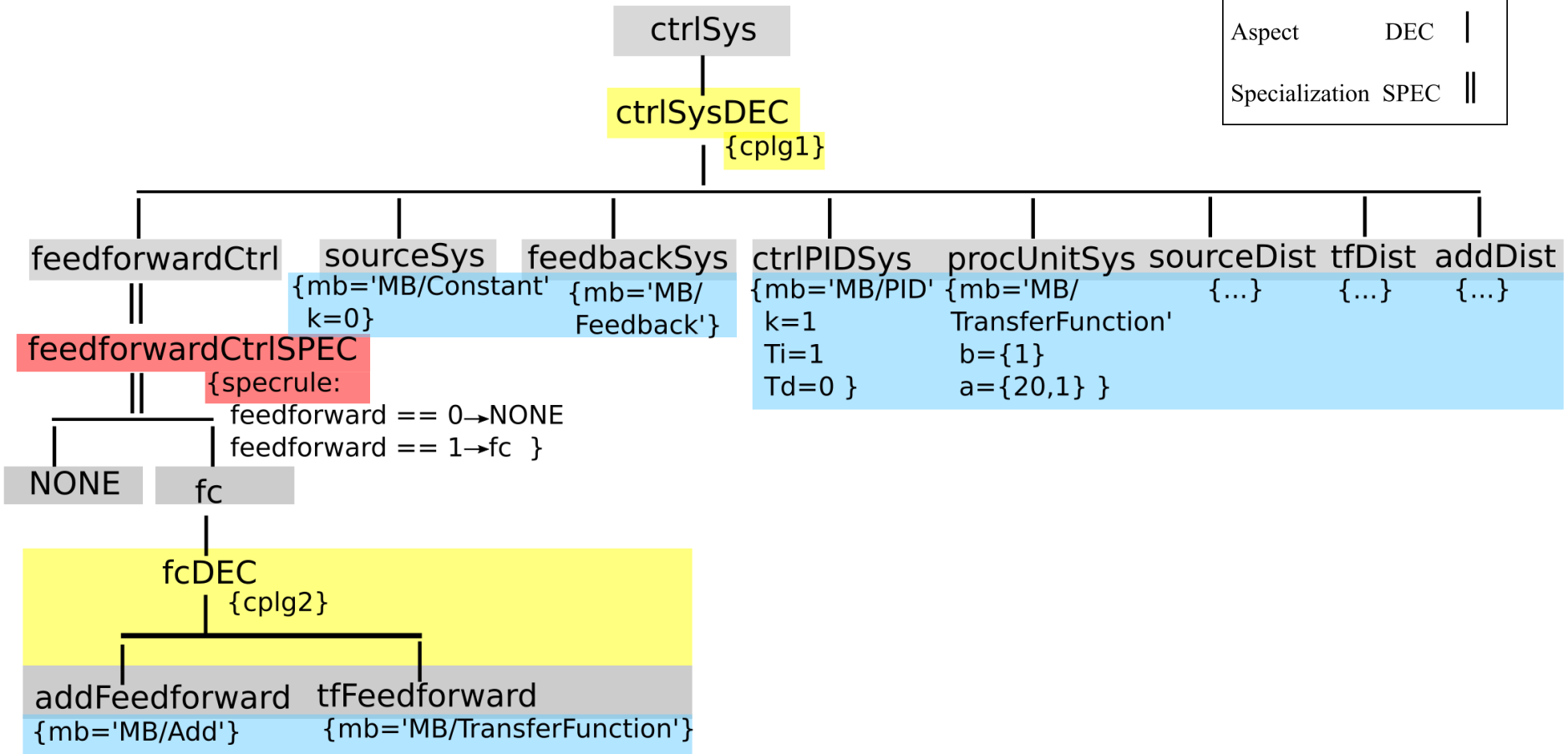




# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	

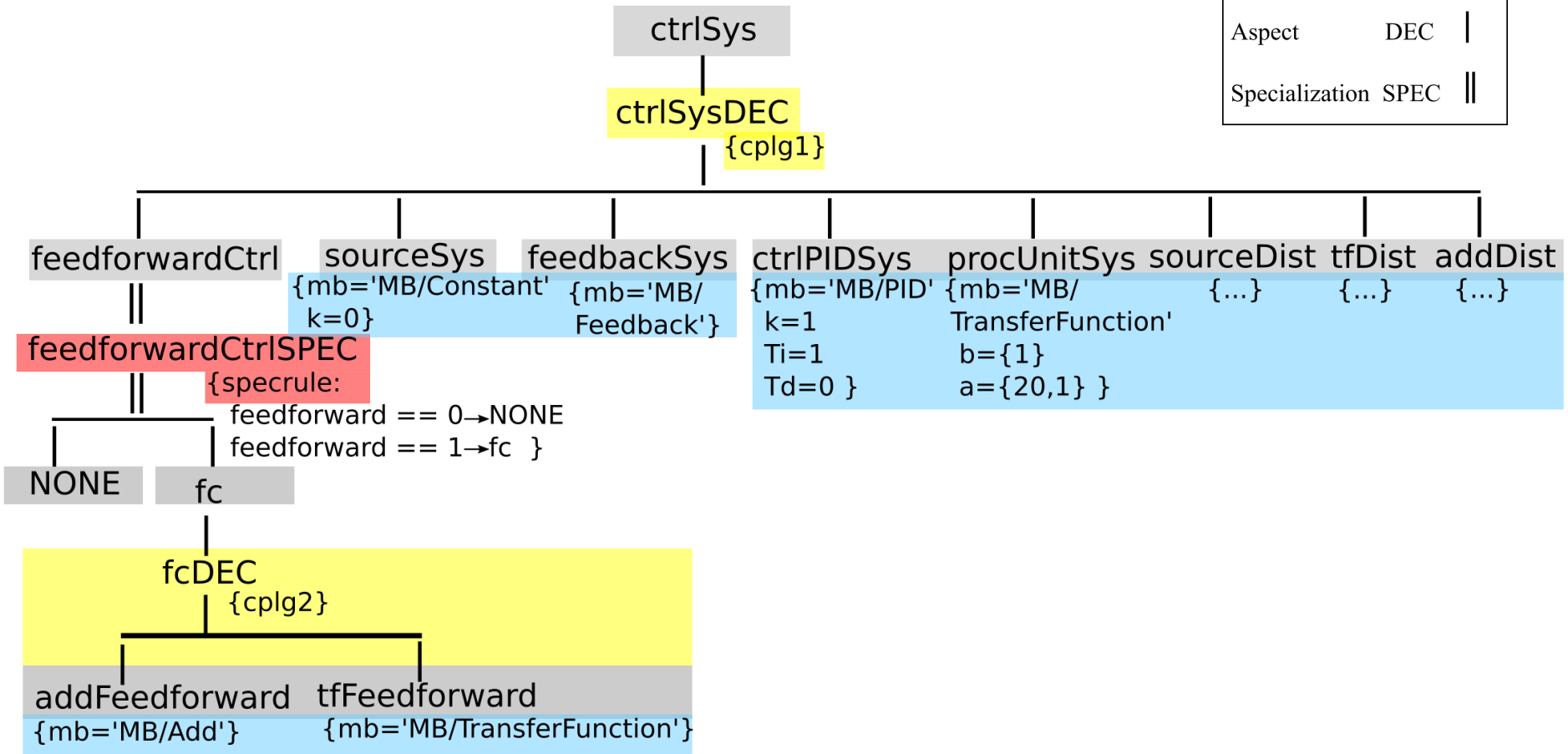




# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	

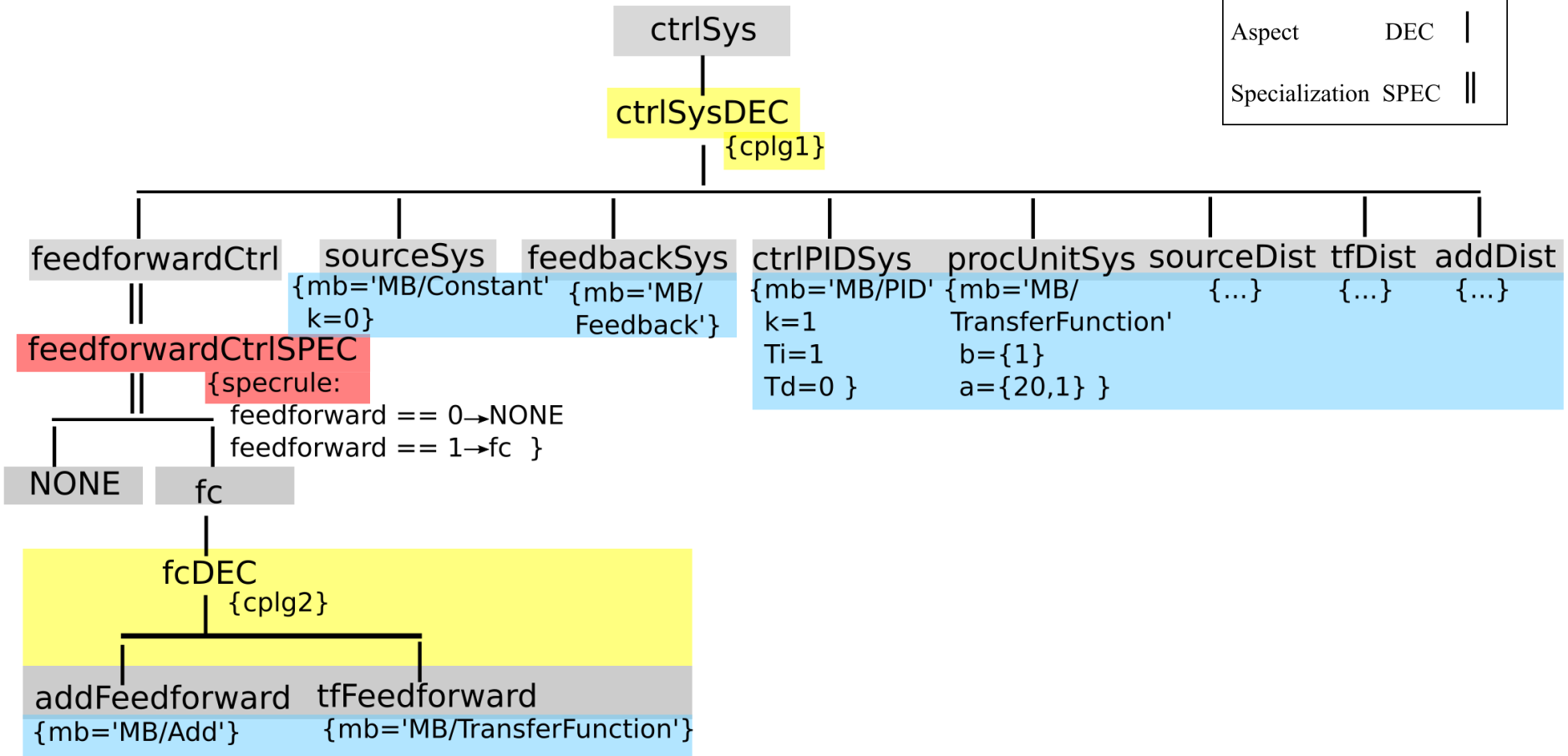




# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	

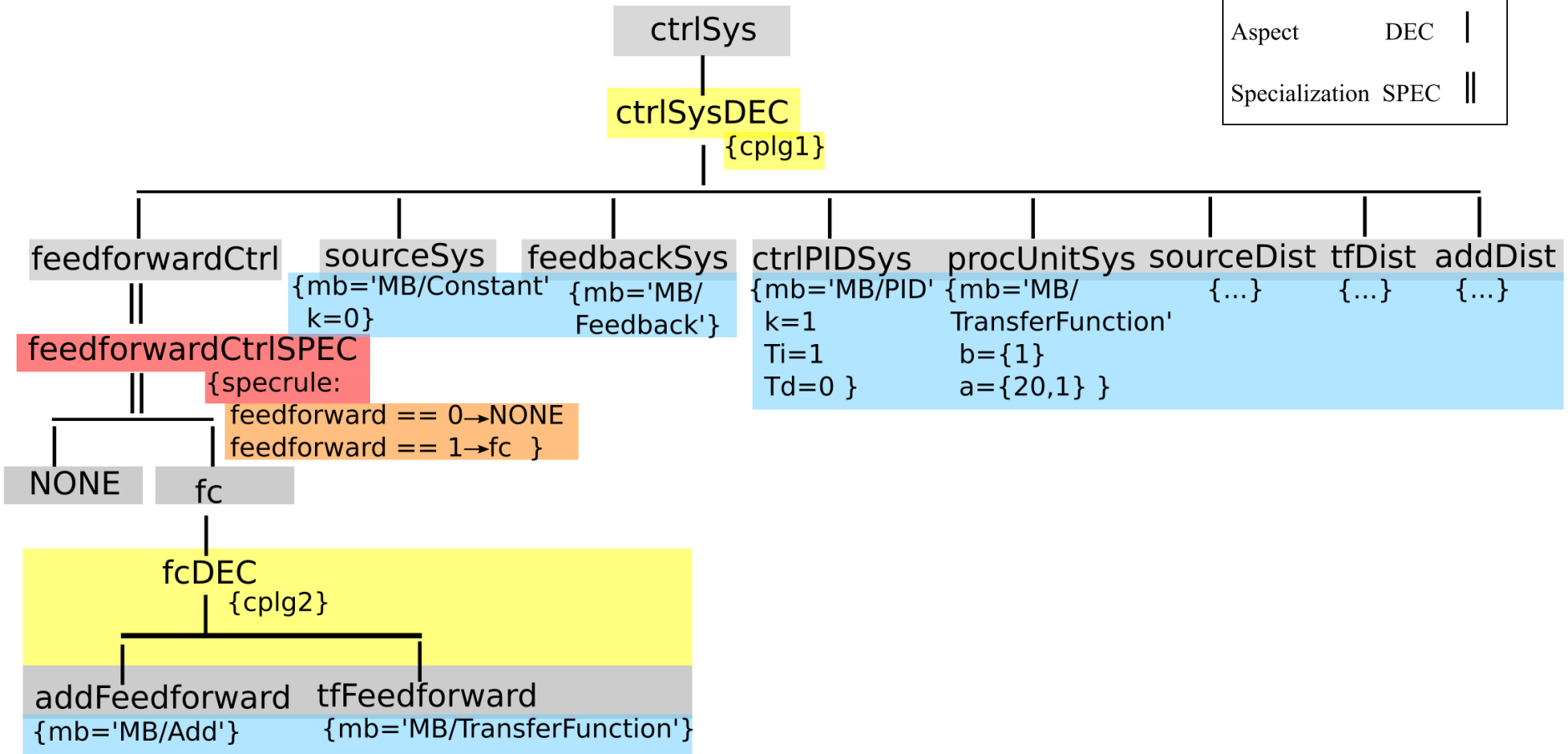




# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	

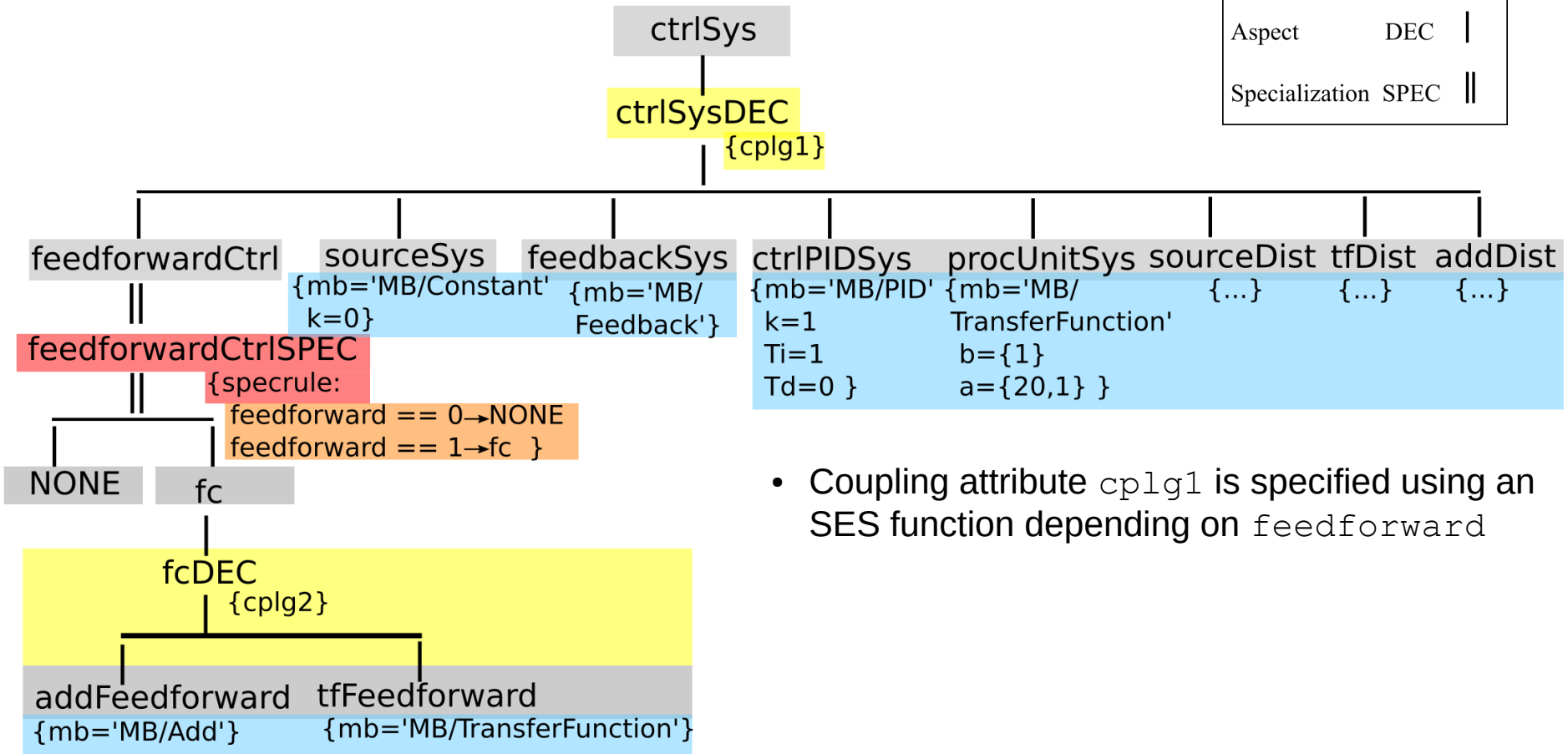




# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	



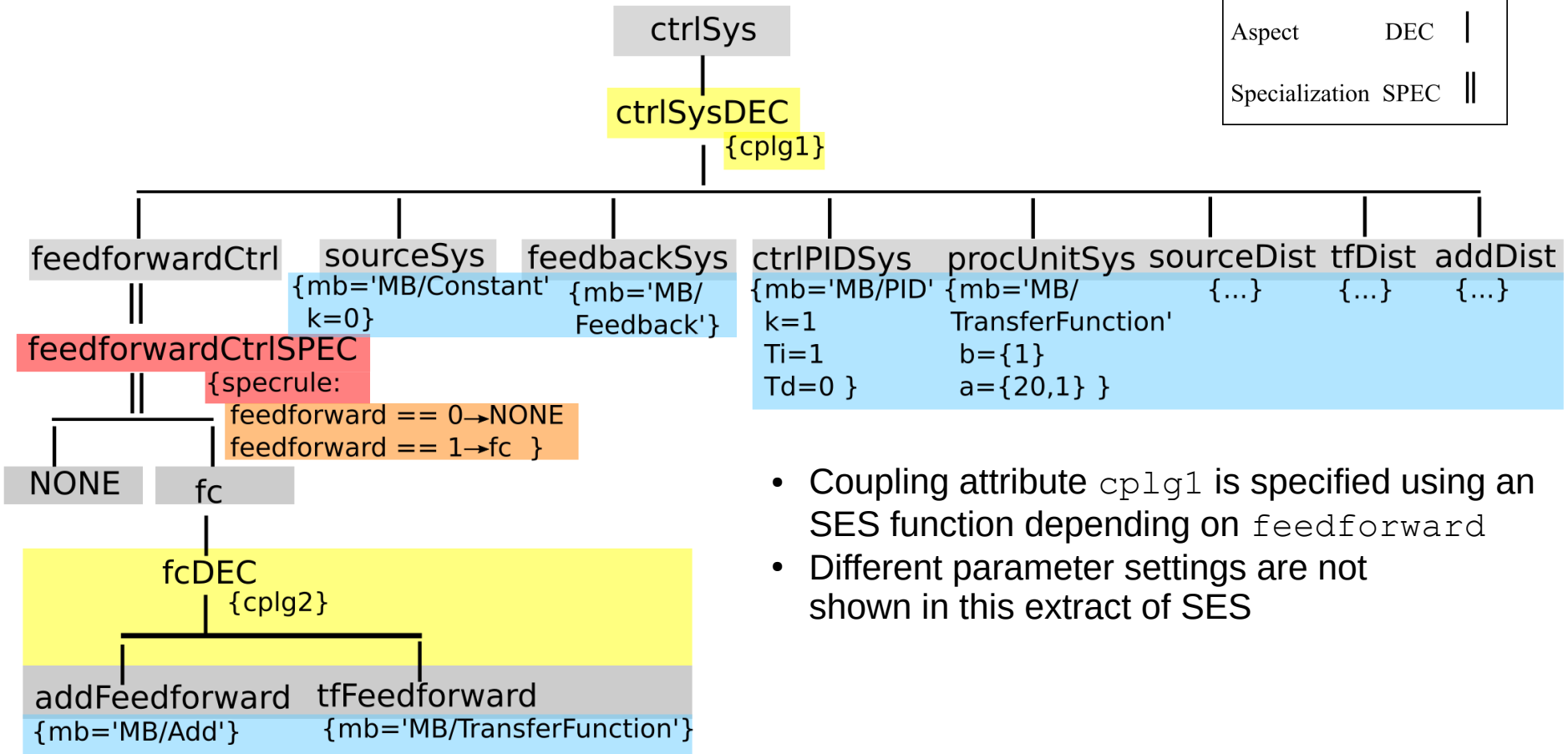
- Coupling attribute `cplg1` is specified using an SES function depending on `feedforward`



# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	



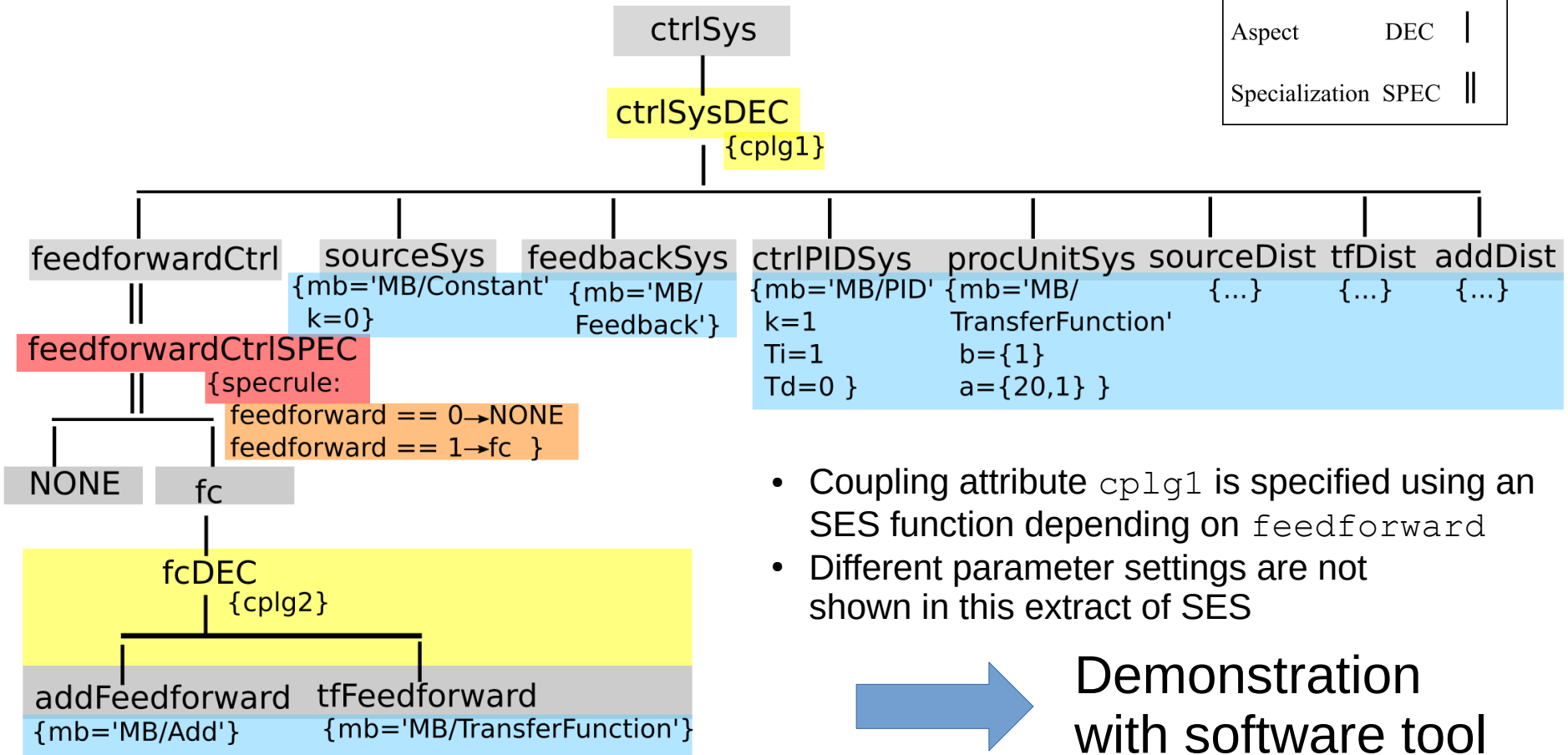
- Coupling attribute `cplg1` is specified using an SES function depending on `feedforward`
- Different parameter settings are not shown in this extract of SES



# More Detailed Extract of the SES

**SES** SESVAR={feedforward}  
 SemanticCondition={feedforward in [0,1]}

Type	Key	Suffix	Edge
Aspect		DEC	
Specialization		SPEC	



- 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/](https://github.com/cea-wismar/SESMB_Inf_Python/)

- Tools

- **SESToPy** → SES editor and IDE

- **SESViewEI** → SES tree viewer

- SESMoPy

- SESEuPy

- SESEcPy





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

- 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
- Merging



# Connection of SESToPy and SESViewEI

The image shows two software windows side-by-side. The left window is titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar with 'File', 'Edit', 'Merge', and 'Transformation'. Below the menu is a toolbar with icons for 'Open', 'Save', 'Empty Current Model', 'Prune', and 'Flatten'. A status bar indicates 'Model 1: not saved'. The main area is divided into three panes: 'Global Settings' on the left, 'Hierarchy Model' in the center, and 'Node Specific Properties' on the right. The 'Global Settings' pane has an 'Information' section and radio buttons for 'SES / PES' (with 'SES' selected), 'incompletely pruned PES', 'PES', and 'flattened PES'. There is also a text area for 'SES comment'. The 'Hierarchy Model' pane shows a 'Node' dropdown set to 'Entity Node' and a 'Tree' view. The 'Node Specific Properties' pane has a 'Default' section and a list of attributes: 'Attributes', 'Aspectrule', 'Number of Replications', 'Coupling', and 'Specrule'. At the bottom left, there are sections for 'SES Variables', 'Semantic Conditions' (checked), 'Selection Constraints', and 'SES Functions'. A 'Help' button is also present. The right window is titled 'SESViewEI' and displays 'SESViewEI - System Entity Structure View in Node.js / Electron'. It shows a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Exit the program here: [Exit]'. Below the message are several controls: 'Tree without Icons?' (checkbox), 'Save Tree as SVG' (button), 'Choose Font Size/Weight for the Tree: 10 normal' (dropdowns), 'Nodename [left] [right] of icon: Node [has] [has no] subtree.' (text), and 'Show/Hide Node Attributes' (button). At the bottom left of the SESToPy window, the text '2019-12-18 / 12:46:14 / Version 2019.12.18' is visible.



# Create SES Tree with SESToPy

The image displays two software windows side-by-side. The left window is SESToPy, titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main interface is divided into several panels: 'Global Settings' with radio buttons for 'SES / PES' (selected), 'incompletely pruned PES', 'PES', and 'flattened PES', and a text area for 'SES comment'; 'Hierarchy Model' showing a tree view with nodes 'EntityRoot' (Entity), 'Descriptive' (Aspect), 'Object1' (Entity), and 'Object2' (Entity); and 'Node Specific Properties' with a table for attributes.

Name	Value	var/fun	comment

The right window is SESViewEI, titled 'SESViewEI - System Entity Structure View in Node.js / Electron'. It displays a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this, there are controls for 'Tree without Icons?' (unchecked), 'Save Tree as SVG', 'Choose Font Size/Weight for the Tree: 15 normal', and 'Nodename [left] [right] of icon: Node [has] [has no] subtree.'. There are also buttons for 'SES variables', 'semantic conditions', and 'Show/Hide Node Attributes'. The main area shows a tree diagram with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Create SES Tree with SESToPy

The image displays two side-by-side windows from the SESToPy application. The left window, titled 'SESToPy - SES Tools in Python3 / PyQt5', shows the main interface with a menu bar (File, Edit, Merge, Transformation, ?), a toolbar (Open, Save, Empty Current Model, Prune, Flatten), and a status bar indicating 'Model 1: not saved'. The main area is divided into several panels: 'Global Settings' (Information, SES/PES options), 'Hierarchy Model' (Node selection, Tree view, and Node Specific Properties), and 'SES comment'. The 'Tree' view shows a hierarchical structure: EntityRoot (Entity) containing Descriptive (Aspect) which contains Object1 and Object2 (both Entity). The 'Node Specific Properties' panel for 'Entity' shows fields for Name, Value, var/fun, and comment, along with 'Insert', 'Delete', and 'Help' buttons. The right window, titled 'SESViewEI - System Entity Structure View in Node.js / Electron', shows a socket server interface. It includes a status message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this are controls for 'Tree without Icons?' (unchecked), 'Save Tree as SVG', 'Choose Font Size/Weight for the Tree: 15 normal', and 'Nodename [left] [right] of icon: Node [has] [has no] subtree.'. The main area of this window displays a visual tree diagram with 'EntityRoot' at the top, 'Descriptive' below it, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Create SES Tree with SESToPy

The image displays two software windows side-by-side. The left window is SESToPy, titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main interface is divided into several panels: 'Global Settings' with 'Information' and 'SES / PES' options (radio buttons for SES, 'incompletely pruned PES', PES, and 'flattened PES', plus a text area for 'SES comment'); 'Hierarchy Model' showing a tree view with columns for 'Node', 'Type', 'MB', and 'atr', containing 'EntityRoot' (Entity), 'Descriptive' (Aspect), 'Object1' (Entity), and 'Object2' (Entity); and 'Node Specific Properties' with a table for 'Attributes' (Name, Value, var/fun, comment) and input fields for 'Name' and 'Value', along with 'Insert', 'Delete', and 'Help' buttons. The right window is SESViewEI, titled 'SESViewEI - System Entity Structure View in Node.js / Electron'. It displays a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this are controls for 'Tree without Icons?' (checkbox), 'Save Tree as SVG' (button), 'Choose Font Size/Weight for the Tree:' (15 normal), and 'Nodename [left] [right] of icon: Node [has] [has no] subtree.' (radio buttons). There are also buttons for 'SES variables', 'semantic conditions', and 'Show/Hide Node Attributes'. The main area shows a tree diagram with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Create SES Tree with SESToPy

The image displays two side-by-side windows from the SESToPy application. The left window, titled 'SESToPy - SES Tools in Python3 / PyQt5', shows a GUI for creating and managing an SES tree. It includes a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main area is divided into several panels: 'Global Settings' (Information, SES/PES options), 'Hierarchy Model' (Tree view showing EntityRoot, Descriptive, Object1, and Object2), and 'Node Specific Properties' (Default, Attributes table, Name/Value fields, and buttons for Insert, Delete, and Help). A red box highlights the 'Object2' node in the tree, with an arrow pointing to it and the text 'Press F2 key to rename'. The right window, titled 'SESViewEI - System Entity Structure View in Node.js / Electron', shows a web-based visualization of the tree. It includes a title bar, a status bar indicating 'A socket server is started on port 54545...', and a main area displaying a tree diagram with nodes 'EntityRoot', 'Descriptive', 'Object1', and 'Object2'. The 'Descriptive' node is highlighted in green, and the 'Object1' and 'Object2' nodes are highlighted in blue. The tree diagram is a simple hierarchical structure with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Create SES Tree with SESToPy

The image displays two software windows side-by-side. The left window is SESToPy, titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main area is divided into several panels: 'Global Settings' with 'Information' and 'SES / PES' options (SES, incompletely pruned PES, PES, flattened PES, SES comment); 'Hierarchy Model' showing a tree structure with nodes like EntityRoot, Descriptive, Object1, and Object2; and 'Node Specific Properties' with fields for Name, Value, var/fun, and comment, along with 'Insert', 'Delete', and 'Help' buttons. A text annotation 'Press F2 key to rename' points to the 'Object2' node in the tree. The right window is 'SESViewEI - System Entity Structure View in Node.js / Electron'. It displays a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this are controls for 'Tree without Icons?' (unchecked), 'Save Tree as SVG', 'Choose Font Size/Weight for the Tree: 15 normal', and 'Nodename [left] [right] of icon: Node [has] [has no] subtree.'. A diagram shows a tree structure with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Create SES Tree with SESToPy

The image displays two side-by-side windows. The left window is SESToPy, titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main area is divided into several panels: 'Global Settings' (Information, SES/PES options), 'Hierarchy Model' (a tree view with nodes like EntityRoot, Descriptive, Object1, and Object2), and 'Node Specific Properties' (Default, Attributes, Name, Value, etc.). A red box highlights the 'Entity Node' dropdown in the Hierarchy Model, and an arrow points to the 'Object2' node with the text 'Press F2 key to rename'. The right window is SESViewEI, titled 'SESViewEI - System Entity Structure View in Node.js / Electron'. It shows a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this are controls for 'Tree without Icons?' (unchecked), 'Save Tree as SVG' (button), 'Choose Font Size/Weight for the Tree: 15 normal' (dropdown), and 'Nodename [left][right] of icon: Node [has][has no] subtree.' (checkboxes). A tree diagram is shown with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.





# Create SES Tree with SESToPy

The image displays two software windows side-by-side. The left window is SESToPy, titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main interface is divided into several panels: 'Global Settings' (Information, SES/PES options), 'Hierarchy Model' (Node dropdown, Tree view, and icons), and 'Node Specific Properties' (Default, Attributes table, Name/Value fields, and Action buttons like Insert, Delete, Help). A text annotation 'Press F2 key to rename' points to the 'Object2' node in the tree. The right window is 'SESViewEI - System Entity Structure View in Node.js / Electron'. It shows a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this are controls for 'Tree without Icons?' (checkbox), 'Save Tree as SVG' (button), 'Choose Font Size/Weight for the Tree:' (15 normal), and 'Nodename [left] [right] of icon: Node [has] [has no] subtree.' (radio buttons). A tree diagram is displayed with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Create SES Tree with SESToPy

The image displays two software windows side-by-side. The left window is SESToPy, titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main interface is divided into several panels: 'Global Settings' with 'Information' and 'SES / PES' options (SES, incompletely pruned PES, PES, flattened PES, SES comment); 'Hierarchy Model' showing a tree structure with nodes like EntityRoot, Descriptive, Object1, and Object2; and 'Node Specific Properties' with fields for Name, Value, var/fun, and comment, along with 'Insert', 'Delete', and 'Help' buttons. A text annotation 'Press F2 key to rename' points to the 'Object2' node in the tree. The right window is 'SESViewEI - System Entity Structure View in Node.js / Electron'. It displays a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this are controls for 'Tree without Icons?' (unchecked), 'Save Tree as SVG' (button), 'Choose Font Size/Weight for the Tree: 15 normal' (dropdown), and 'Nodename [left] [right] of icon: Node [has] [has no] subtree.' (checkboxes). A 'Show/Hide Node Attributes' button is also present. The main area shows a hierarchical tree diagram with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Create SES Tree with SESToPy

The image displays two software windows side-by-side. The left window is SESToPy, titled 'SESToPy - SES Tools in Python3 / PyQt5'. It features a menu bar (File, Edit, Merge, Transformation, ?), a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten, and a status bar indicating 'Model 1: not saved'. The main interface is divided into several panels: 'Global Settings' with 'Information' and 'SES / PES' options (SES, incompletely pruned PES, PES, flattened PES, SES comment); 'Hierarchy Model' showing a tree structure with nodes like EntityRoot, Descriptive, Object1, and Object2; and 'Node Specific Properties' with fields for Name, Value, var/fun, and comment, along with 'Insert', 'Delete', and 'Help' buttons. A text annotation 'Press F2 key to rename' points to the 'Object2' node in the tree. The right window is 'SESViewEI - System Entity Structure View in Node.js / Electron'. It displays a message: 'A socket server is started on port 54545. SESToPy can connect to this server now. Save received string as XML file? [checked] - Exit the program here: [Exit]'. Below this are controls for 'Tree without Icons?' (unchecked), 'Save Tree as SVG', 'Choose Font Size/Weight for the Tree: 15 normal', and 'Nodename [left] [right] of icon: Node [has] [has no] subtree.' There are also buttons for 'SES variables', 'semantic conditions', and 'Show/Hide Node Attributes'. The main area shows a hierarchical tree diagram with 'EntityRoot' at the top, 'Descriptive' as a child, and 'Object1' and 'Object2' as children of 'Descriptive'.



# Edit Entity Node with SESToPy

The screenshot shows the SESToPy application window with the 'Edit Entity Node' dialog box open. The dialog box is titled 'Entity Node' and contains a table of attributes. The 'Attributes' section is highlighted with an orange border. Below the table, there are input fields for 'Name' and 'Value', and buttons for 'Insert', 'Delete', and 'Help'.

Global Settings

Information

SES / PES

- SES
- incompletely pruned PES
- PES
- flattened PES

SES comment

SES Variables

Semantic Conditions

Selection Constrains

SES Functions

Model 1: not saved

Connection to server program taking the XML tree: Server IP  Server Port

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

Hierarchy Model

Node: Entity Node

Tree	Type	MB	atr	ars	cpl	srs	uid
EntityRoot	Entity						1
Descriptive	Aspect						2
Object1	Entity						3
Object2	Entity		x				4

Node Specific Properties

Default

Name	Value	var/fun	comment
1 var1	1		

Name:

Value:

Aspectrule

Number of Replications

Coupling

Specrule

2021-02-02 / 19:04:14 / Version 2021.02.02



# Edit Specialization Node with SESToPy

The screenshot shows the SESToPy application window with the following components:

- Global Settings:** Information and SES Variables sections. The SES Variables table is highlighted with an orange box:

Name	Value	Comment
1 var1	1	

- Hierarchy Model:** A tree view showing the model structure. The 'Descriptive' node is selected, and its 'Spec' child is highlighted with an orange box:

Tree	Type	MB	atr	ars	cpl	srs	uid
EntityRoot	Entity						1
Descriptive	Spec					x	2
Object1	Entity						3
Object2	Entity		x				4

- Node Specific Properties:** A panel on the right showing properties for the selected node. The 'Specrule' table is highlighted with an orange box:

Node	uid	Condition	result	comment
1 Object1	3	var1==1	T	
2 Object2	4	var1==2	F	

At the bottom left, the status bar shows: 2021-02-02 / 19:05:48 / Version 2021.02.02



# SESToPy Open the Feedback Example

The screenshot shows the SESToPy application interface. The main window has a menu bar with 'File', 'Edit', 'Merge', and 'Transformation'. The 'Open' button in the 'File' menu is highlighted with an orange box. The 'Open an SES from JSON' dialog box is open, showing the path 'SESToPy > Examples > FeedbackControl' in the address bar. The file 'Feedback.jsonsestree' is selected in the file list. The 'Dateiname:' field is empty, and the file type is set to 'JSON SES Tree (\*.jsonsestree)'. The 'Öffnen' (Open) button is highlighted with a blue box.

Name	Änderungsdatum	Typ
Feedback.jsonsestree	18.01.2020 17:52	JSONSESTREE-Datei



# SESToPy Feedback Example

The screenshot shows the SESToPy application window. At the top, it says "SESToPy - SES Tools in Python3 / PyQt5". Below that is a menu bar with "File", "Edit", "Merge", and "Transformation?". There are icons for "Open", "Save", "Empty Current Model", "Prune", and "Flatten".

The main area is titled "Model 1: Feedback" and shows a connection to a server program taking the XML tree. The server IP is "127.0.0.1" and the server port is "54545". There are "Connect" and "Disconnect" buttons.

Below this, there are tabs for "Model 1" through "Model 10".

The interface is divided into several sections:

- Global Settings:** Includes "Information" and "SES Variables" sections.
- Hierarchy Model:** Shows a tree view of the model structure. The selected node is "Entity Node". The tree includes nodes like "ctrlSys", "ctrlSysDEC", "feedforwardCtrl", "feedforwardCtrlSPEC", "fc", "fcDEC", "tfFeedforward", "addFeedforward", "NONE", "sourceSys", "feedbackSys", "ctrlPIDSys", "procUnitSys", "sourceDist", "tfDist", and "addDist".
- Node Specific Properties:** Shows a table with columns "Name", "Value", "var/fun", and "r/mme". The table contains one row: "1 mb 'MB/Add'".
- Form Fields:** There are input fields for "Name" and "Value", and buttons for "Insert", "Delete", and "Help".
- Checkboxes:** There are checkboxes for "Semantic Conditions" (checked), "Selection Constraints", and "SES Functions".

The screenshot shows the SESViewE1 application window. The title is "SESViewE1 - System Entity Structure View in Node.js / Electron".

At the top, it says "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:

Below this, there are several controls:

- "Tree without Icons?
- "Save Tree as SVG" button
- "Choose Font Size/Weight for the Tree: 12 | normal" dropdown
- "Nodename [left] [right] of icon: Node [has] [has no] subtree." text
- "Show/Hide Node Attributes" button

There are two tables:

SES variables	semantic conditions
feedforward = 1, comment:	feedforward in [0,1], result: T

Below the tables is a tree diagram showing the system entity structure. The root node is "ctrlSys". It has a child "ctrlSysDEC". "ctrlSysDEC" has children "feedforwardCtrl", "sourceSys", "feedbackSys", "ctrlPIDSys", "procUnitSys", "sourceDist", and "tfDist". "feedforwardCtrl" has a child "feedforwardCtrlSPEC". "feedforwardCtrlSPEC" has children "fc" and "NONE". "fc" has a child "fcDEC". "fcDEC" has children "tfFeedforward" and "addFeedforward".

At the bottom, there are input fields for "Name" (var2) and "Value" (3), and buttons for "Insert", "Delete", and "Help". There are also sections for "Aspectrule", "Number of Replications", "Coupling", and "Specrule".



# SESToPy XML Export / Import

The screenshot shows the SESToPy application window titled "SESToPy - SES Tools in Python3 / PyQt5". The "File" menu is open, and the "Import from XML" and "Export to XML" options are highlighted with a red rectangle. The menu items are: File, Edit, Merge, Transformation, ?, Open (Shift), Save (Shift), Save As (Shift), Empty Current Model (Shift), Import from XML, and Export to XML. Below the menu, a list of file paths is visible, including examples like "FeedbackControl/Feedback.jsonsestree" and "Example03\_FeedbackControl\_FMI/Feedback\_FPES.jsonsestree". The main workspace displays a hierarchical tree structure of entities, with "addFeedforward" selected. The right-hand side of the interface features a configuration panel with fields for "Name" (var2) and "Value" (3), along with "Insert", "Delete", and "Help" buttons. At the bottom, there are sections for "Semantic Conditions", "Selection Constraints", and "SES Functions". The status bar at the very bottom indicates the file path and the last saved time: "2021-02-02 / 19:12:17 / Version 2021.02.02 / File: C:/Users/Win10/SESMB\_Infrastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree / Last saved: 2020-01-18 - 17:52:4...".





# SESToPy SES Variables

The screenshot shows the SESToPy application window. The main interface is divided into several panels:

- Global Settings:** Includes a menu bar (File, Edit, Merge, Transformation, ?) and a toolbar with icons for Open, Save, Empty Current Model, Prune, and Flatten.
- Model Selection:** A row of tabs labeled Model 1 through Model 10. Model 1 is selected and titled "Model 1: Feedback".
- Connection:** A status bar at the top right shows "Connection to server program taking the XML tree: Server IP 127.0.0.1 Server Port 54545" with Connect and Disconnect buttons.
- SES Variables Table:** Located in the left panel, it contains one entry:

Name	Value	Comment
1 feedforward	1	
- Hierarchy Model:** A tree view showing the model structure. The selected node is "Entity Node". The tree includes:
  - ctrlSys (Entity)
  - ctrlSysDEC (Aspect)
  - feedforwardCtrl (Entity)
  - feedforwardCtrlSPEC (Spec)
  - fc (Entity)
  - fcDEC (Aspect)
  - tfFeedforward (Entity) with MB 'MB/TransferFunction' and attribute x
  - addFeedforward (Entity) with MB 'MB/Add' and attribute x
  - NONE (Entity)
  - sourceSys (Entity) with MB 'MB/Constant' and attribute x
  - feedbackSys (Entity) with MB 'MB/Feedback' and attribute x
  - ctrlPIDSys (Entity) with MB 'MB/PID' and attribute x
  - procUnitSys (Entity) with MB 'MB/TransferFunction' and attribute x
  - sourceDist (Entity) with MB 'MB/Step' and attribute x
  - tfDist (Entity) with MB 'MB/TransferFunction' and attribute x
  - addDist (Entity) with MB 'MB/Add' and attribute x
- Node Specific Properties:** A panel on the right showing properties for the selected node. It includes a table for attributes:

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

Below the table are input fields for Name (var2) and Value (3), and buttons for Insert, Delete, and Help.



# SESToPy Semantic Conditions

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

The main interface is titled "Model 1: Feedback" and shows a connection to a server program with the XML tree. The server IP is "127.0.0.1" and the server port is "54545".

The interface is divided into several panels:

- Global Settings:** Includes "Information" and "SES Variables" sections.
- Semantic Conditions:** A table with columns "Semantic Condition" and "result". It contains one entry: "1 feedforward in [0,1] T". Below the table is a text input field for "Semantic Condition" and buttons for "Insert", "Delete", and "Help".
- Hierarchy Model:** A tree view showing the model structure. The selected node is "Entity Node". The tree includes nodes like "ctrlSys", "ctrlSysDEC", "feedforwardCtrl", "feedforwardCtrlSPEC", "fc", "fcDEC", "tfFeedforward", "addFeedforward", "NONE", "sourceSys", "feedbackSys", "ctrlPIDSys", "procUnitSys", "sourceDist", "tfDist", and "addDist".
- Node Specific Properties:** A panel for editing properties of the selected node. It includes a table for "Attributes" with columns "Name", "Value", "var/fun", and "comment". The table contains one row: "1 mb 'MB/Add'". Below the table are input fields for "Name" (value: "var2") and "Value" (value: "3"), and buttons for "Insert", "Delete", and "Help".



# SESToPy Selection Rules (here Specrule)

The screenshot shows the SESToPy application window. The main area displays a 'Hierarchy Model' tree with nodes like 'ctrlSys', 'ctrlSysDEC', 'feedforwardCtrl', and 'feedforwardCtrlSPEC'. The 'feedforwardCtrlSPEC' node is highlighted with an orange box. To the right, the 'Node Specific Properties' panel shows a 'Specrule' table with two rows, also highlighted with an orange box.

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



# SESToPy Attributes

The screenshot shows the SESToPy application window. The main area displays a 'Hierarchy Model' tree with nodes like 'ctrlSys', 'ctrlSysDEC', 'feedforwardCtrl', 'feedforwardCtrlSPEC', 'fc', 'fcDEC', 'tfFeedforward', 'addFeedforward', 'NONE', 'sourceSys', 'feedbackSys', 'ctrlPIDSys', 'procUnitSys', 'sourceDist', 'tfDist', and 'addDist'. The 'ctrlPIDSys' node is highlighted with an orange box. To the right, the 'Node Specific Properties' panel shows an 'Attributes' table with the following data:

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

Below the table are input fields for 'Name' (value: var2) and 'Value' (value: 3), along with 'Insert', 'Delete', and 'Help' buttons. The bottom status bar shows the file path: 'C:/Users/Win10/SESMB\_Infrastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree/'.



# SESToPy Coupling List

The screenshot shows the SESToPy application window. The main area displays a 'Hierarchy Model' tree with nodes like 'ctrlSys', 'ctrlSysDEC', 'feedforwardCtrl', 'feedforwardCtrlSPEC', 'fc', and 'fcDEC'. The 'fcDEC' node is selected and highlighted in orange. To the right, the 'Node Specific Properties' panel is open, showing a 'Coupling' table with 4 rows of data. Below the table are fields for 'source node' and 'sink node', and buttons for 'Insert', 'Delete', and 'Help'.

	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

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

The screenshot displays the SESToPy application window with the following components:

- Global Settings:** Information, SES Variables, Semantic Conditions (checked), Selection Constraints.
- SES Functions:** A code editor showing a function definition for `cpifcn`. The function takes `feedforward` and `children` as arguments and defines a list of couplings (`cplg`) based on the `feedforward` flag. It includes comments for fixed and variable couplings.
- Hierarchy Model:** A tree view showing the model structure. The selected node is `ctrlSysDEC`, which is an Aspect. Its children include `feedforwardCtrl` (Entity), `fc` (Entity), `fcDEC` (Aspect), `tfFeedforward` (Entity), and `addFeedforward` (Entity). Other entities listed include `sourceSys`, `feedbackSys`, `ctrlPIDSys`, `procUnitSys`, `sourceDist`, `tfDist`, and `addDist`.
- Node Specific Properties:** A panel for the selected `ctrlSysDEC` node. It shows the `Coupling` section with a table for defining couplings. The table has columns for `source`, `uid`, `port name / type`, `sink`, `uid`, `port name / type`, and `comment`. Below the table are fields for `source node` and `sink node`, and buttons for `Insert`, `Delete`, and `Help`. The `SES function` dropdown is set to `cpifcn(feedforward,CHILDREN)`. The `Aspectrule` field contains the expression: `[y / SPR', 'feedforwardCtrl', 'u2 / SPR', ']', '[feedforwardCtrl, 'y / SPR', 'procUnitSys', 'u / SPR', ']]`.

At the bottom of the window, the status bar shows: 2021-02-02 / 19:16:21 / Version 2021.02.02 / File: C:/Users/Win10/SESMB\_infastructure/SESToPy/Examples/FeedbackControl/Feedback.jsonsestree / Last saved: 2020-01-18 - 17:52:45



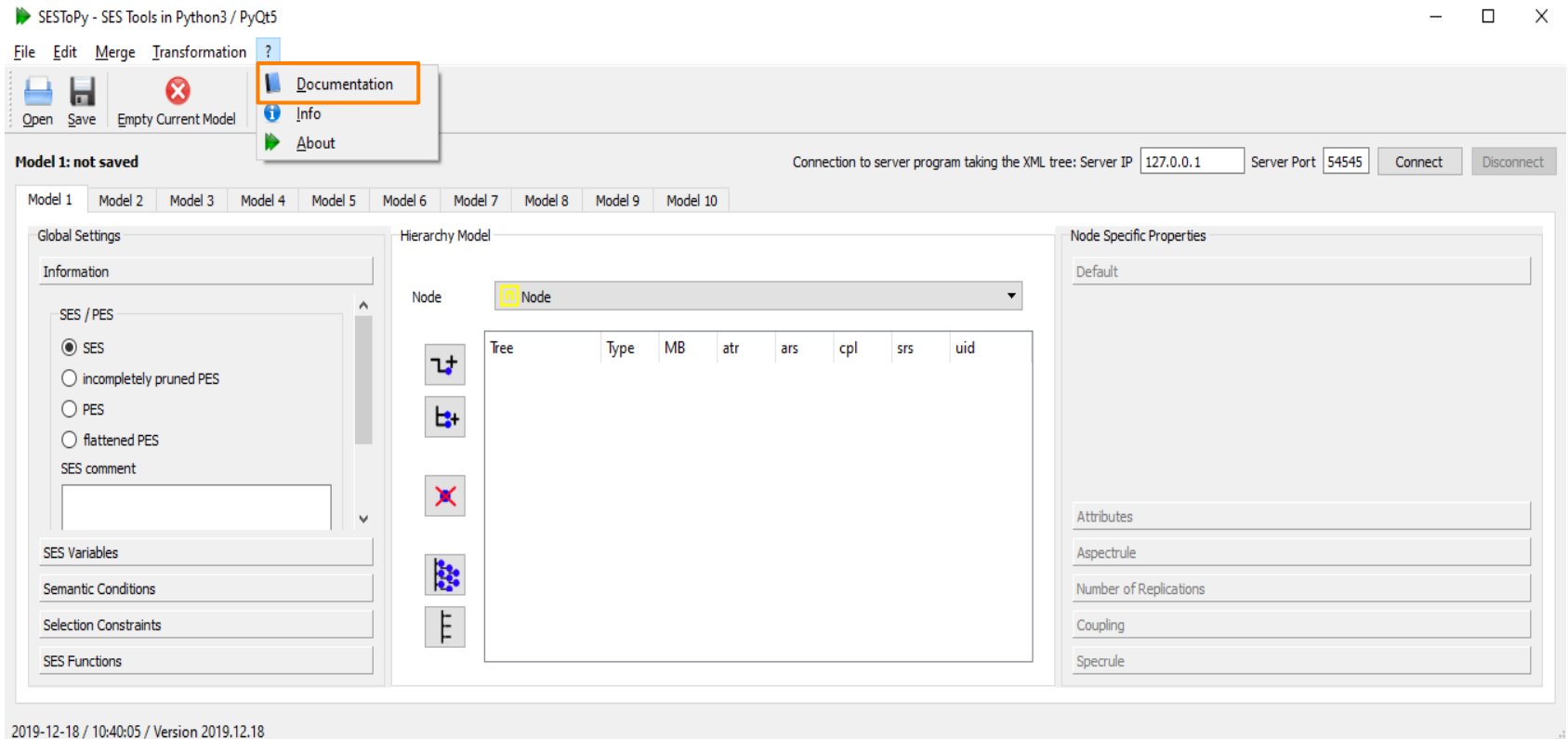
# SESToPy Merging

The screenshot shows the SESToPy application window with the following components:

- Menu:** The 'Merge' menu is open, showing options: 'Add SubSES', 'Save SubSES', 'Import SubSES from XML', and 'Export SubSES to XML'. The 'Merge' menu is highlighted with an orange box.
- Global Settings:** Includes sections for Information, SES Variables, Semantic Conditions (checked), Selection Constraints, and SES Functions.
- Code Editor:** Contains Python code for a function named 'cplfcn'. The code defines a list of children, appends couplings, and handles variable couplings based on the 'feedforward' flag.
- Hierarchy Model:** A tree view showing the model structure. The selected node is 'Entity Node'. The tree includes nodes like 'ctrlSys', 'ctrlSysDEC', 'feedforwardCtrl', 'feedforwardCtrlSPEC', 'fc', 'fcDEC', 'tfFeedforward', 'addFeedforward', 'sourceSys', 'feedbackSys', 'ctrlPIDSys', 'procUnitSys', 'sourceDist', 'tfDist', and 'addDist'. A 'NONE' node is currently selected.
- Node Specific Properties:** A panel for editing the selected node's properties. It includes fields for Name (var2), Value (3), and buttons for Insert, Delete, and Help. Below are sections for Aspectrule, Number of Replications, Coupling, and Specrule.



# SESToPy Documentation



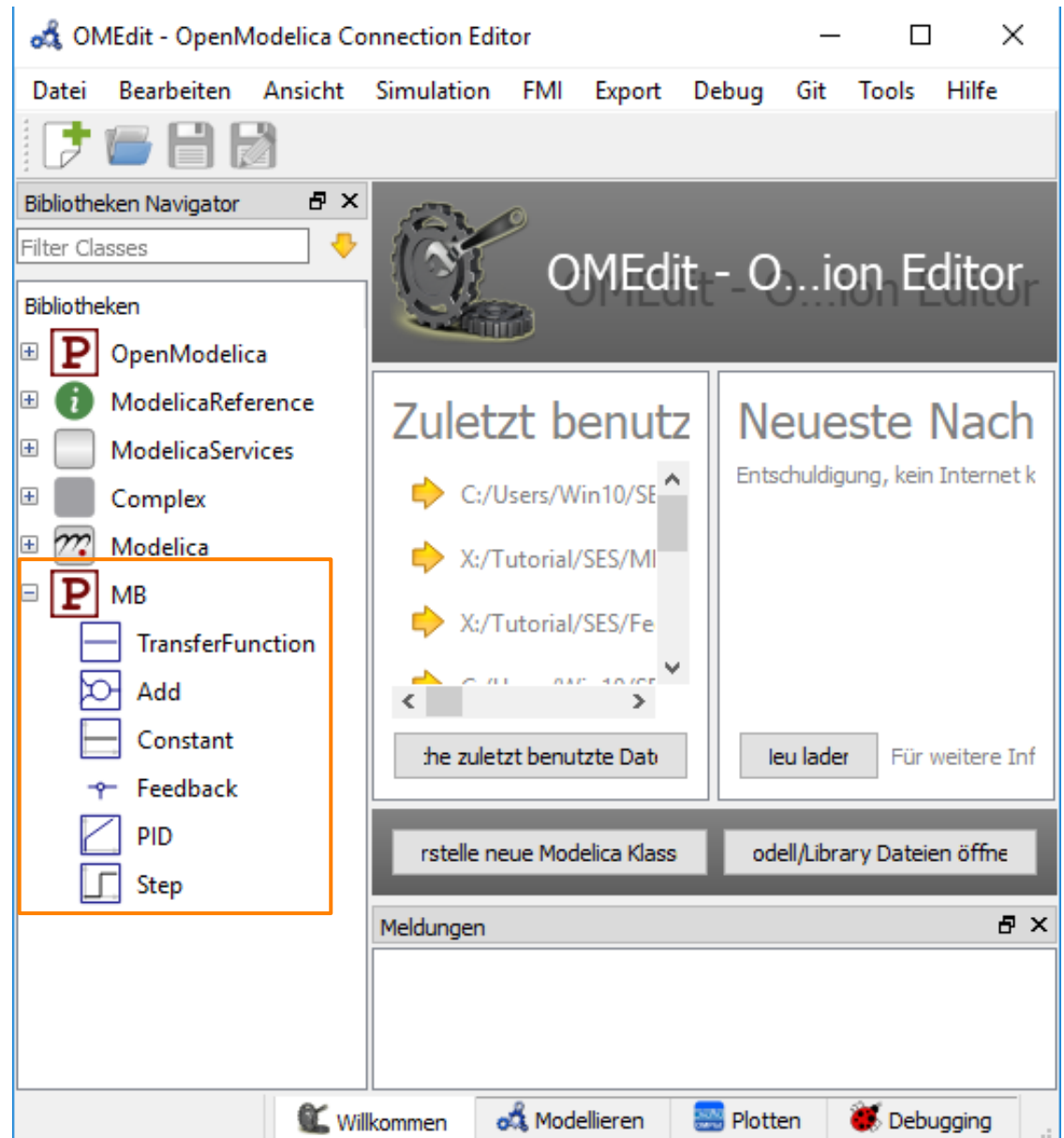
- See the documentation for more information





# OpenModelica MB

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



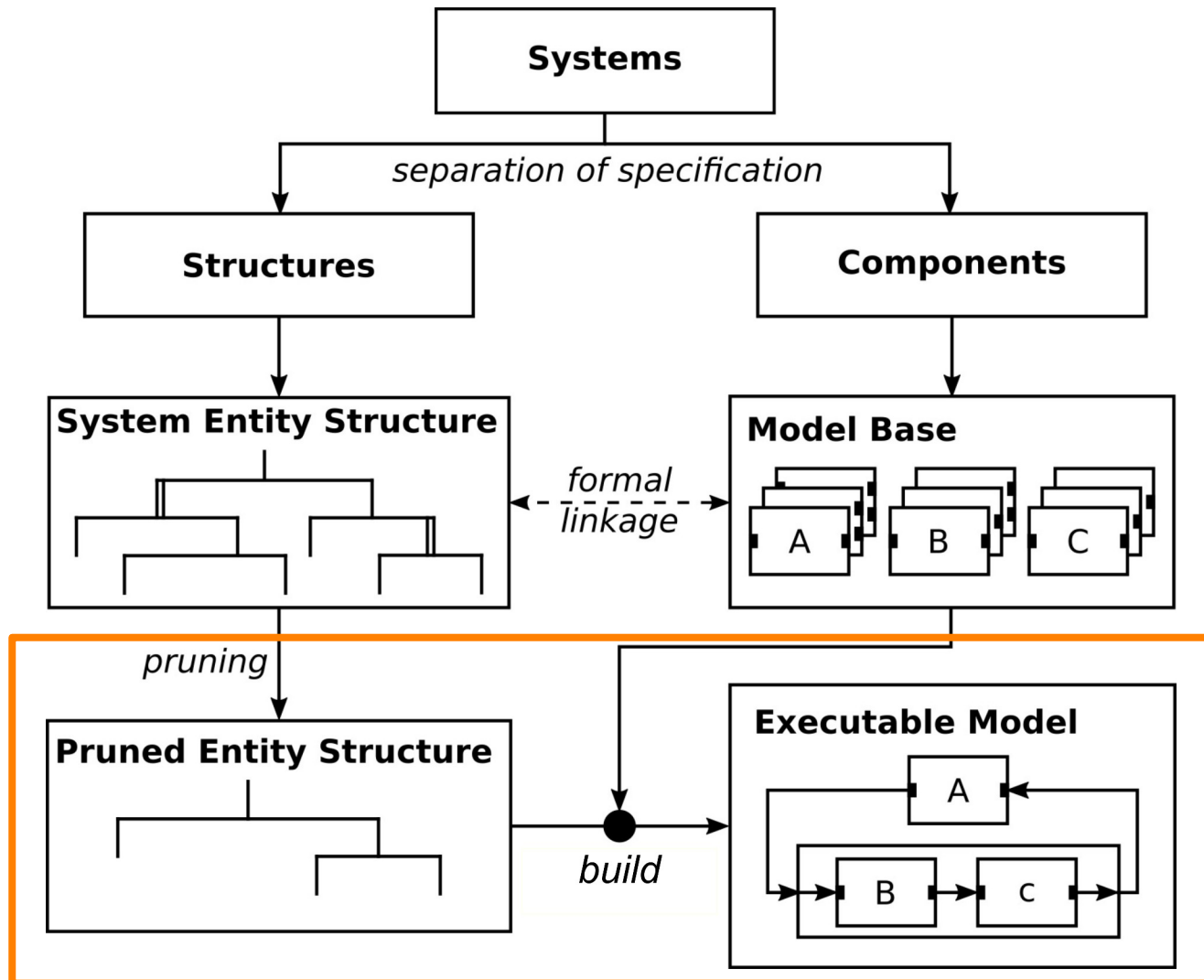


# Outline

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



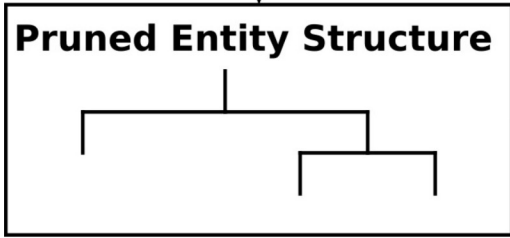
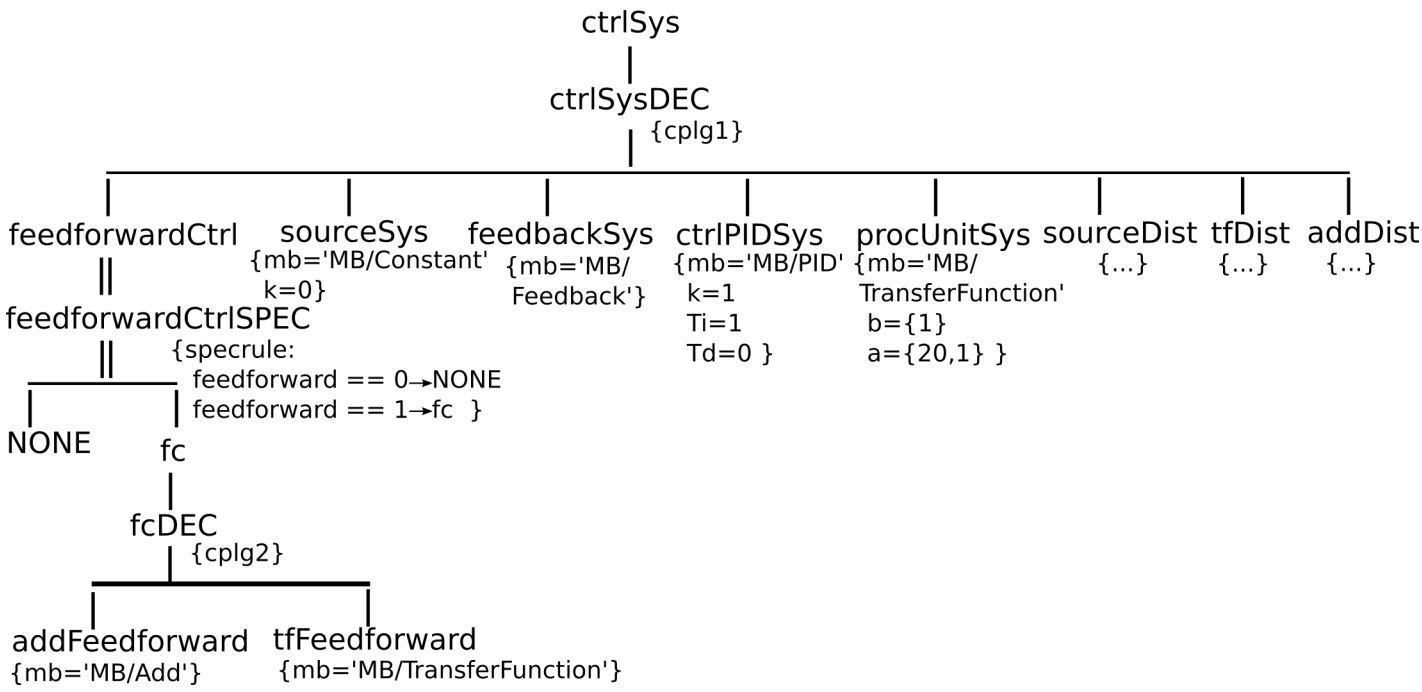
# Model Selection and Generation



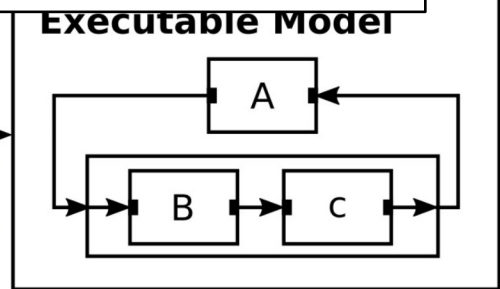


# Model Selection and Generation of Variant #1

**SES** SESVAR={feedforward}  
SemanticCondition={feedforward in [0,1]}



*build*

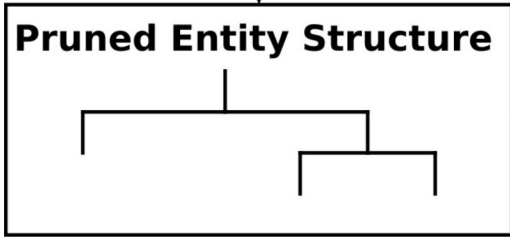
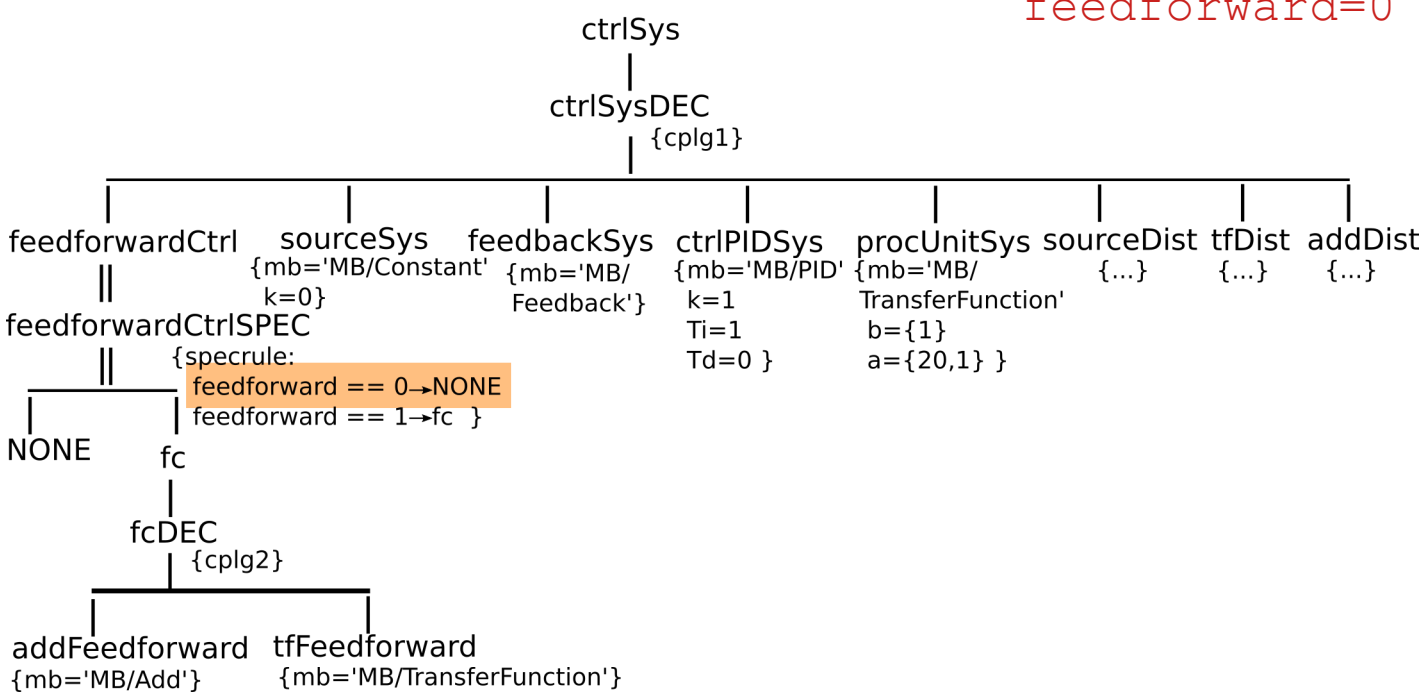




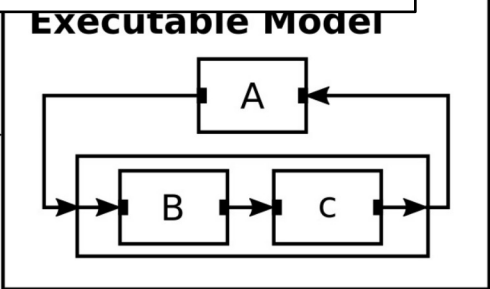
# Model Selection and Generation of Variant #1

**SES** SESVAR={feedforward}  
SemanticCondition={feedforward in [0,1]}

pruning with  
feedforward=0

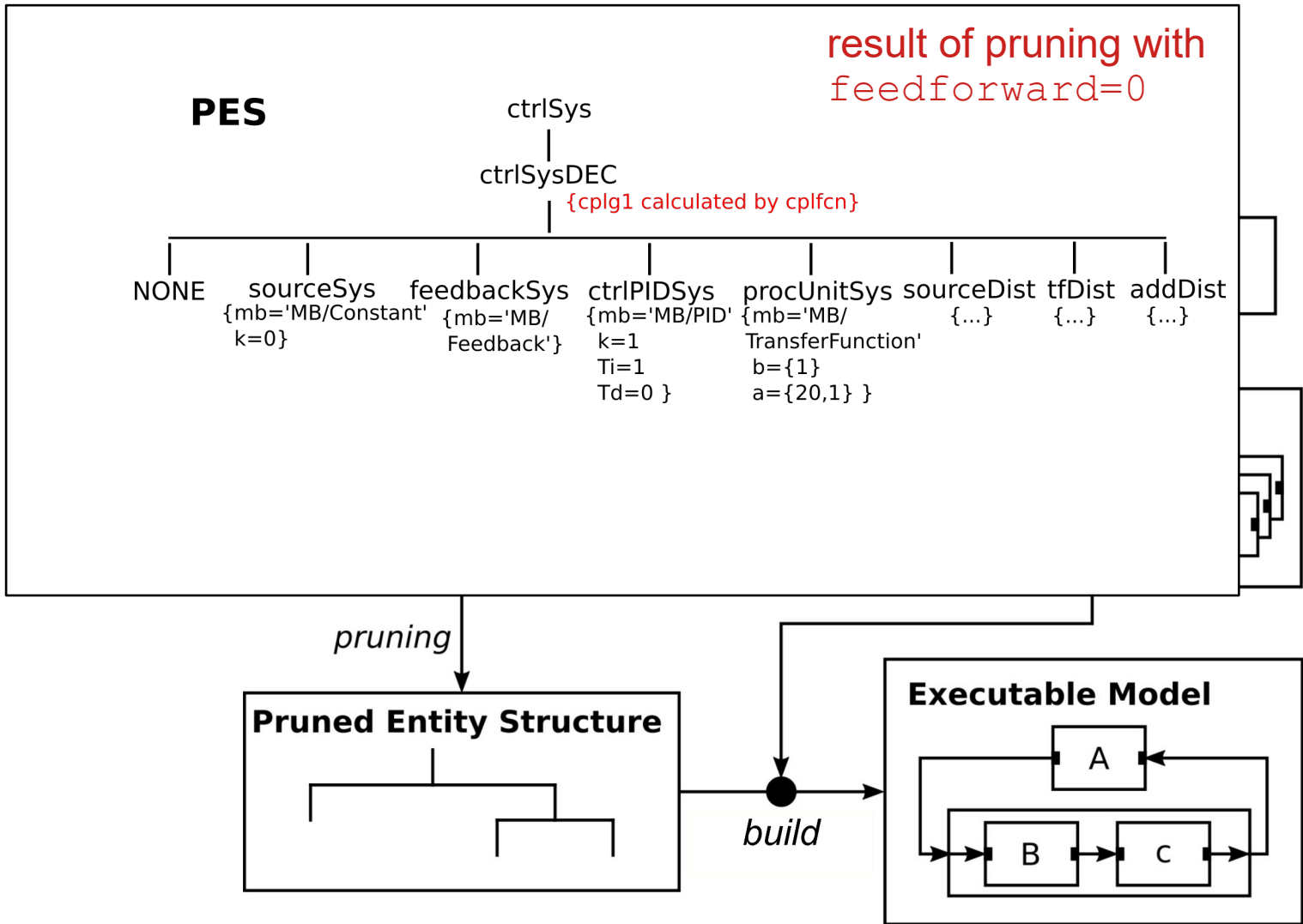


*build*



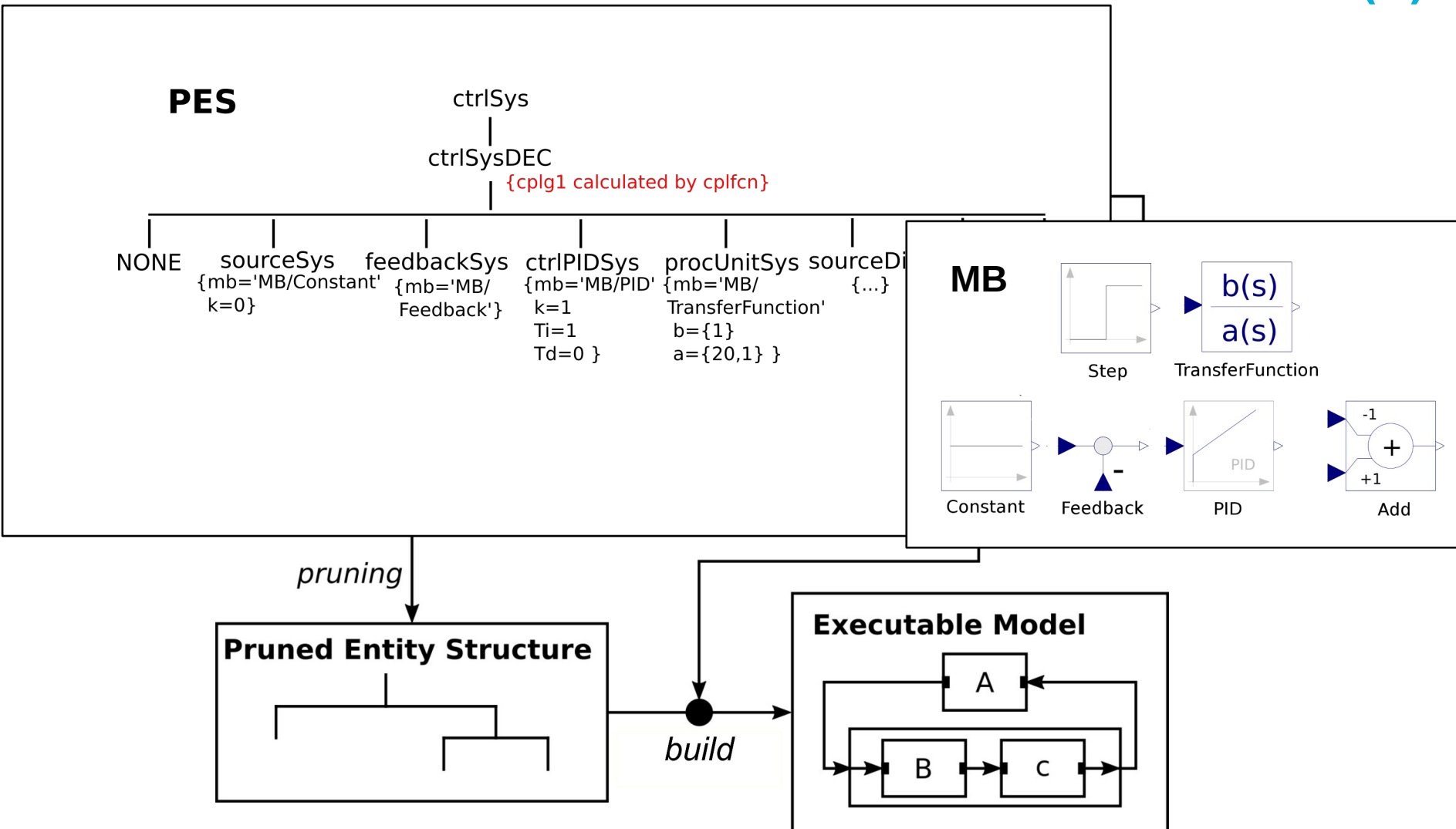


# Model Selection and Generation of Variant #1 (2)



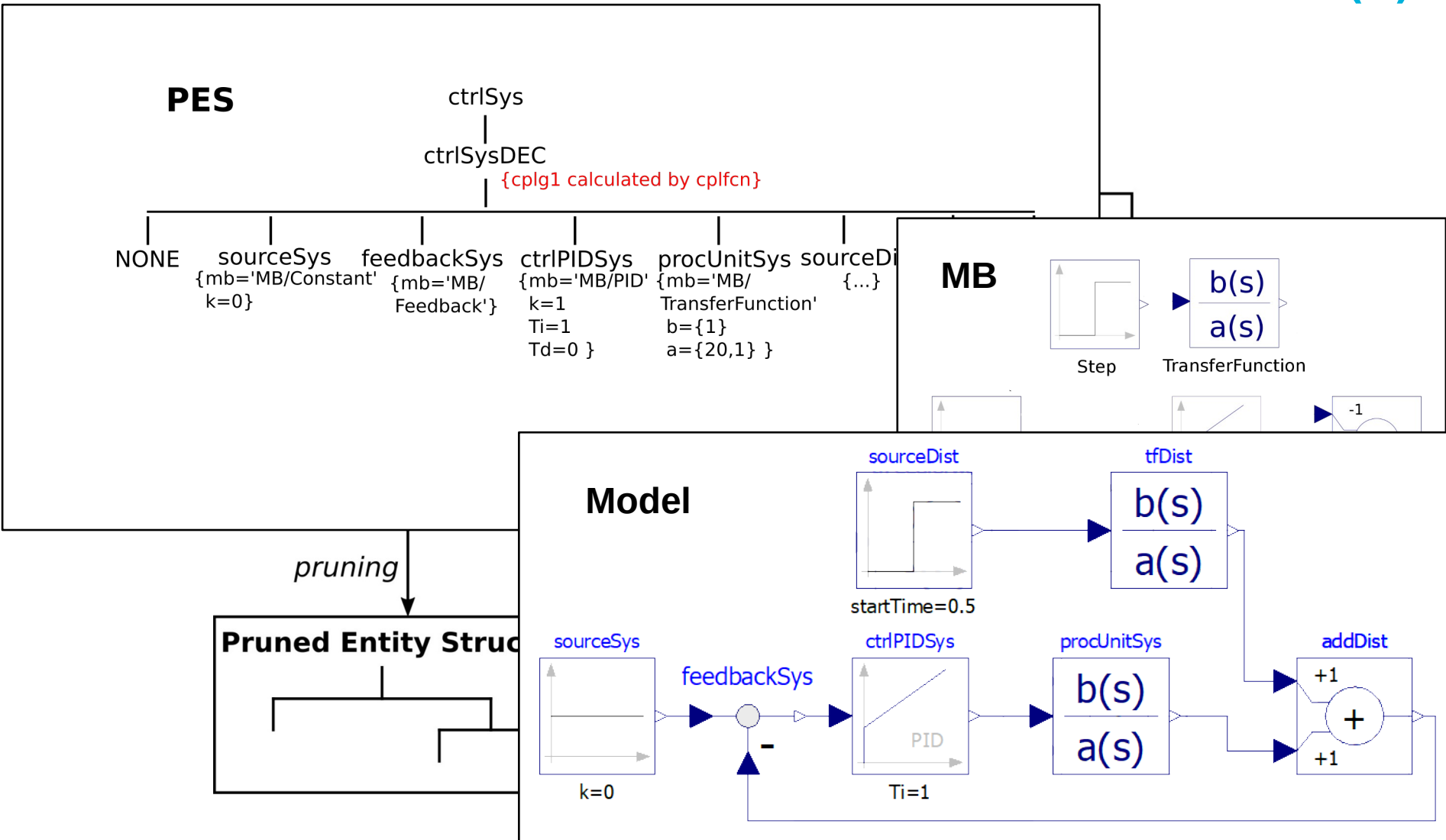


# Model Selection and Generation of Variant #1 (3)





# Model Selection and Generation of Variant #1 (4)

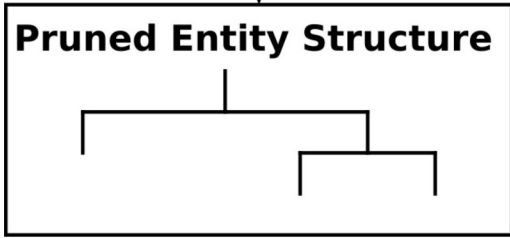
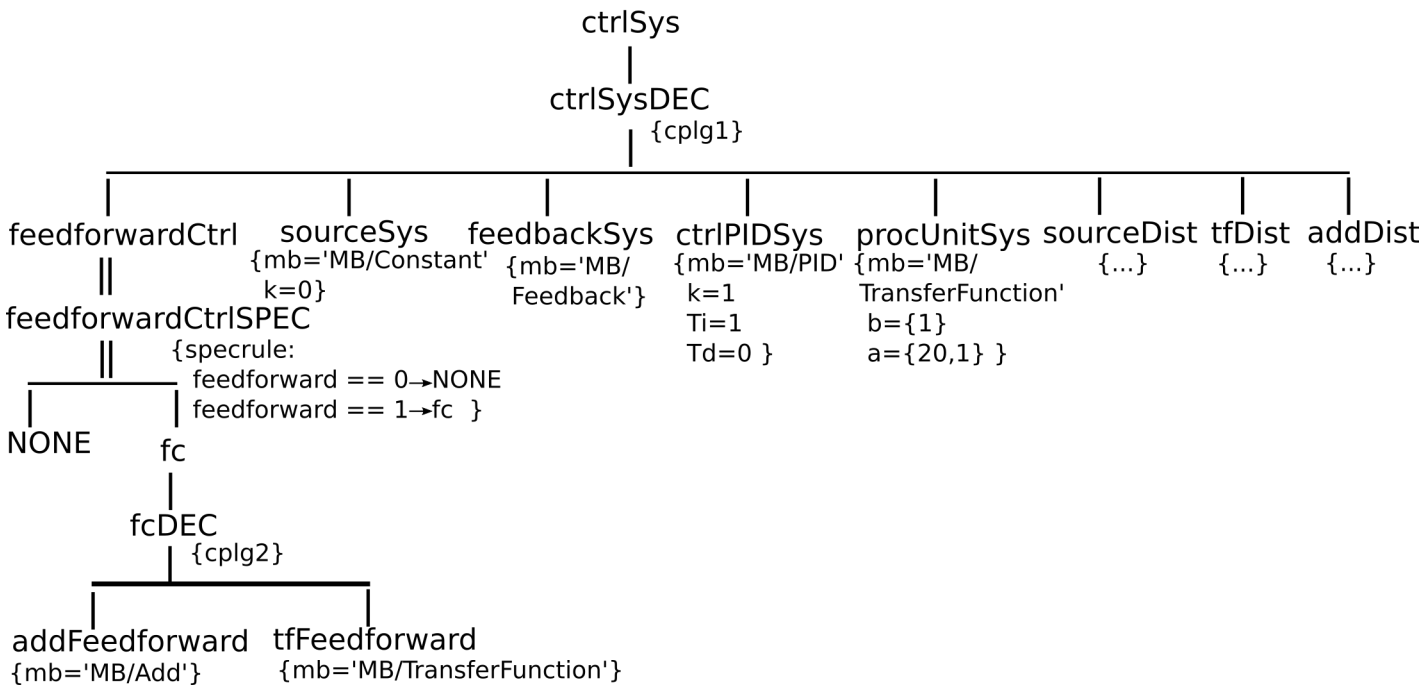




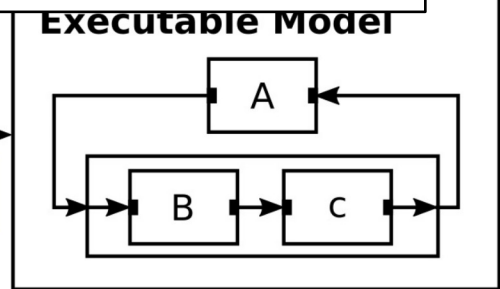


# Model Selection and Generation of Variant #2

**SES** SESVAR={feedforward}  
SemanticCondition={feedforward in [0,1]}



*build*

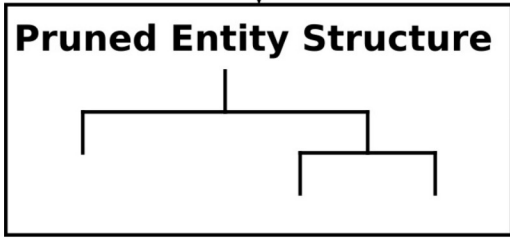
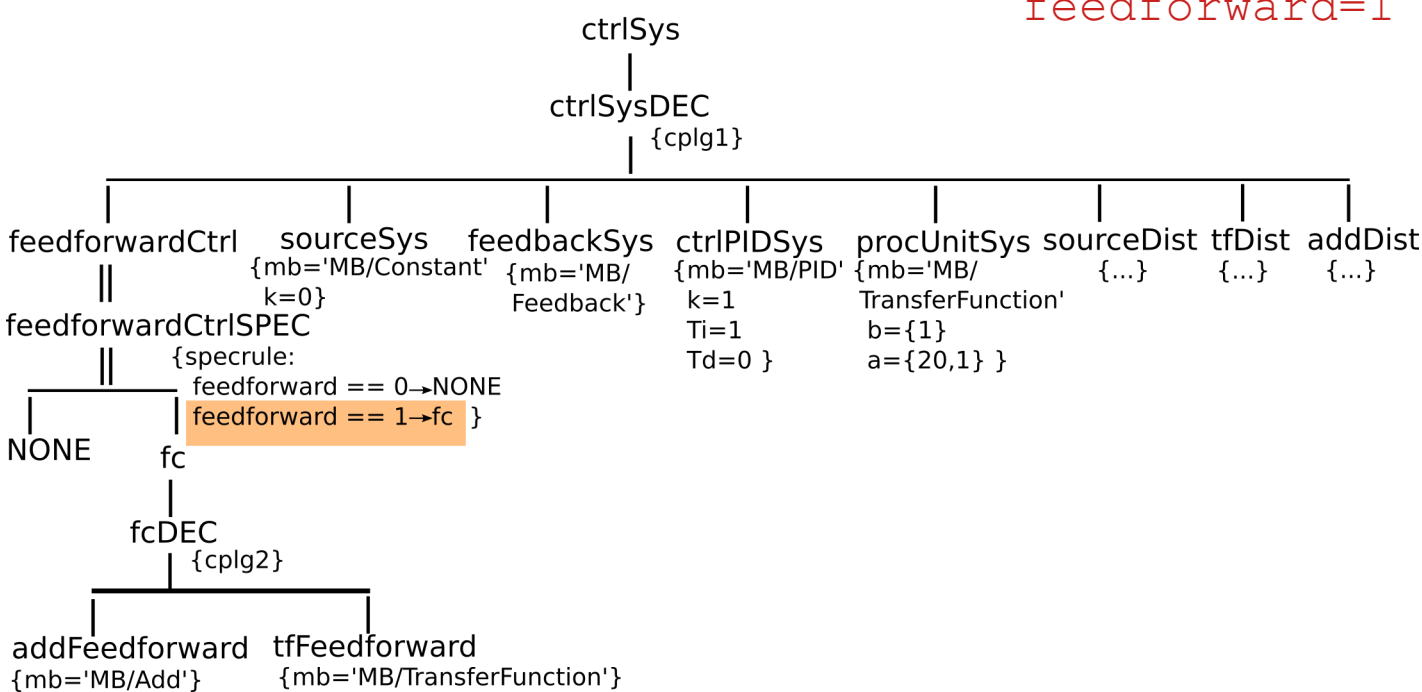




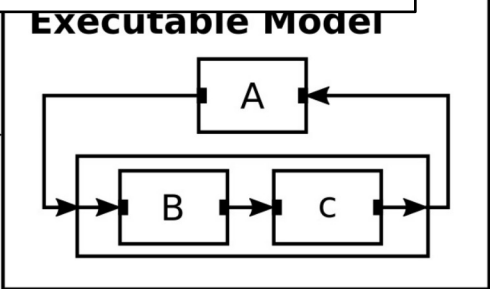
# Model Selection and Generation of Variant #2

**SES** SESVAR={feedforward}  
SemanticCondition={feedforward in [0,1]}

pruning with  
feedforward=1

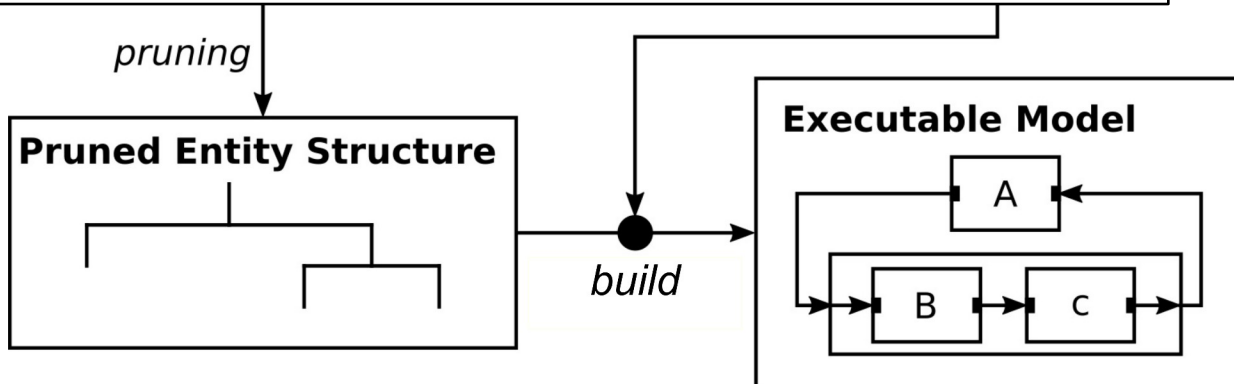
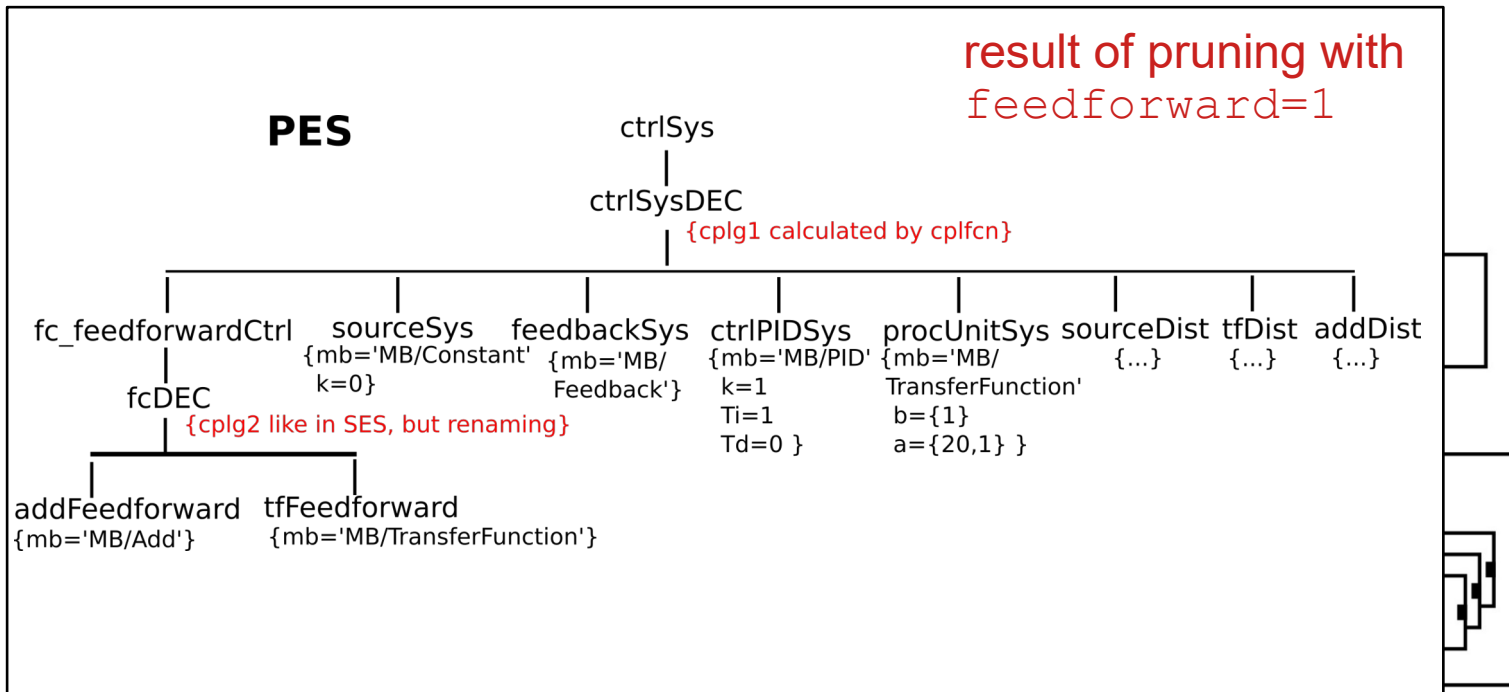


*build*



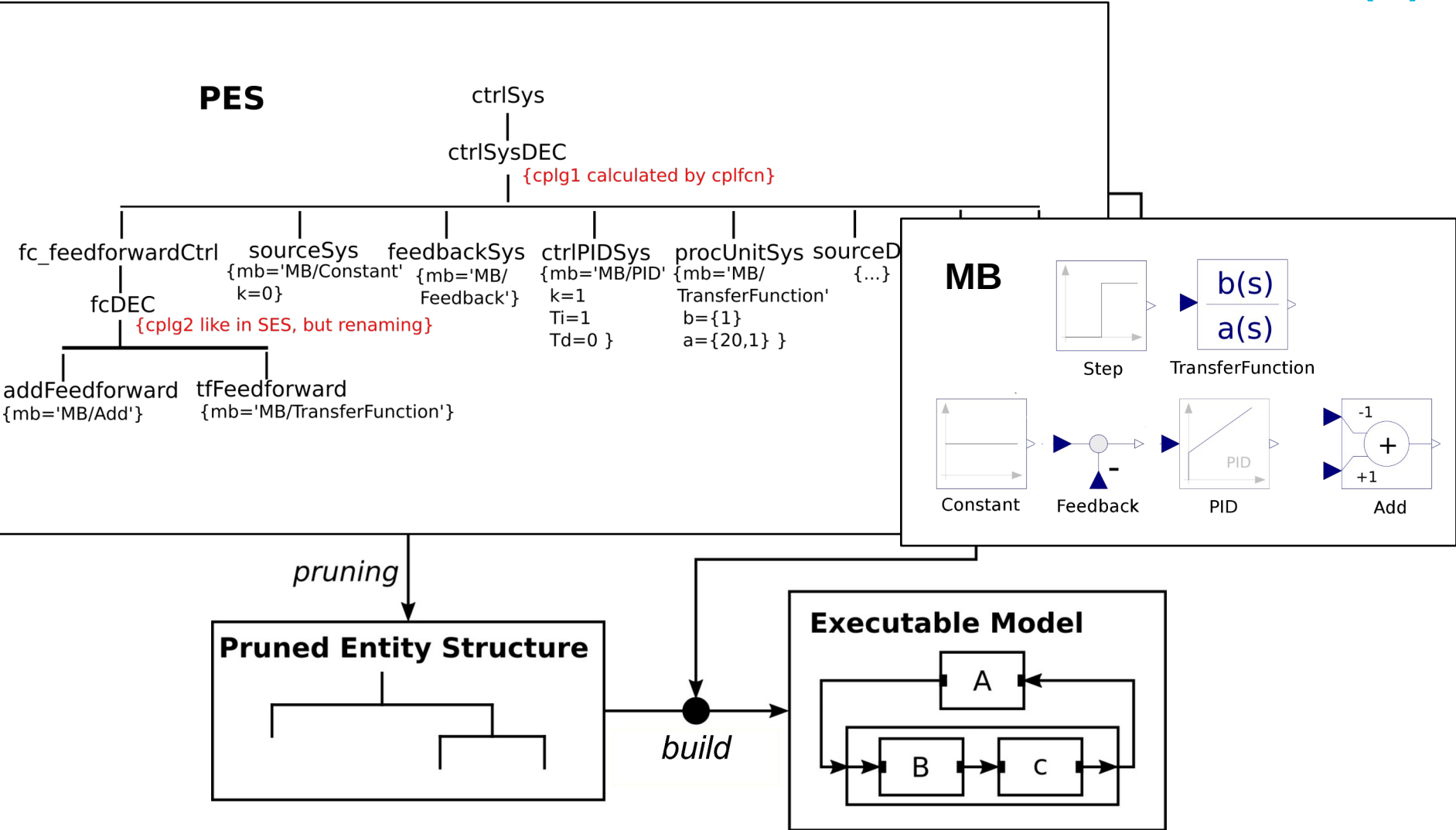


# Model Selection and Generation of Variant #2 (2)





# Model Selection and Generation of Variant #2 (3)





# Model Selection and Generation of Variant #2 (4)

**PES**

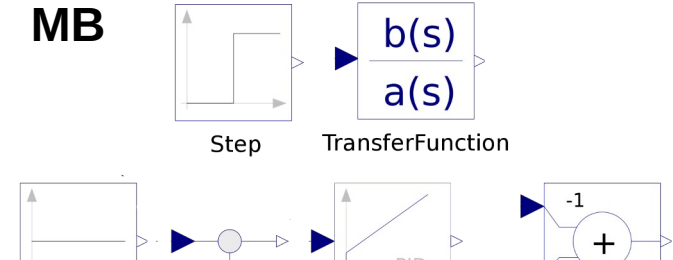
ctrlSys  
|  
ctrlSysDEC  
{cplg1 calculated by cplfcn}

fc\_feedforwardCtrl | sourceSys {mb='MB/Constant' k=0} | feedbackSys {mb='MB/Feedback'} | ctrlPIDSys {mb='MB/PID' k=1 Ti=1 Td=0} | procUnitSys {mb='MB/TransferFunction' b={1} a={20,1}} | sourced {...}

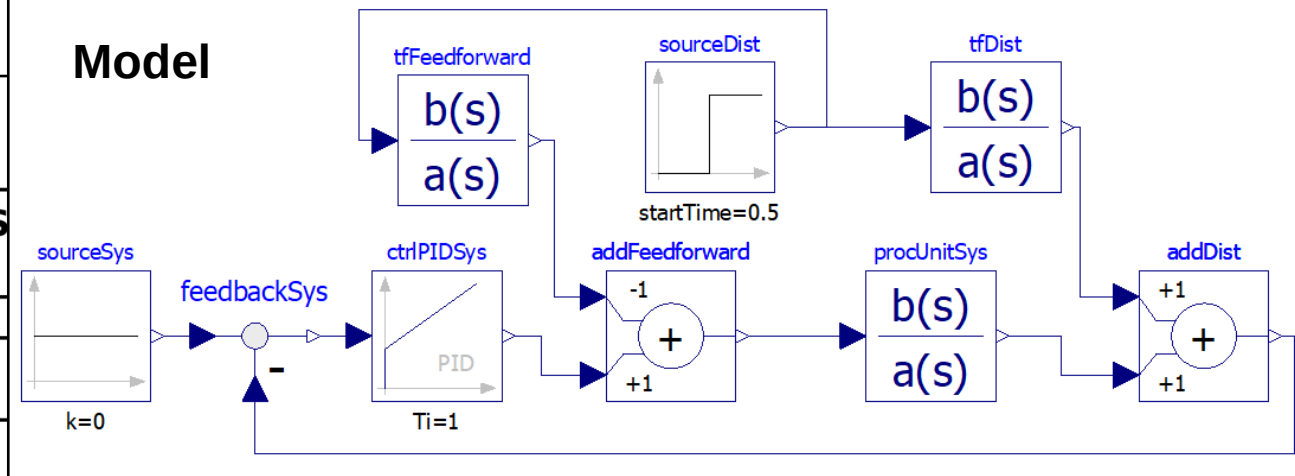
fcDEC {cplg2 like in SES, but renaming}

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

**MB**

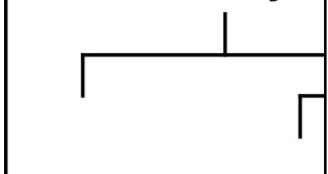


**Model**



*pruning*

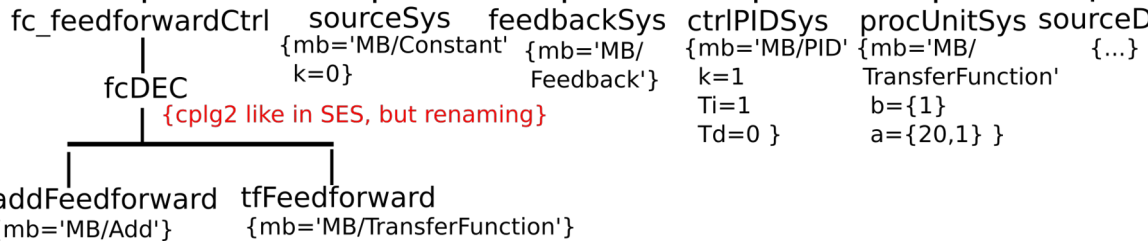
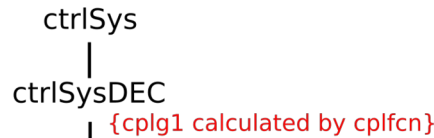
**Pruned Entity S**



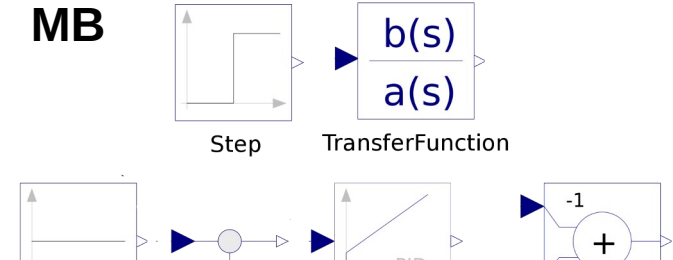


# Model Selection and Generation of Variant #2 (4)

**PES**



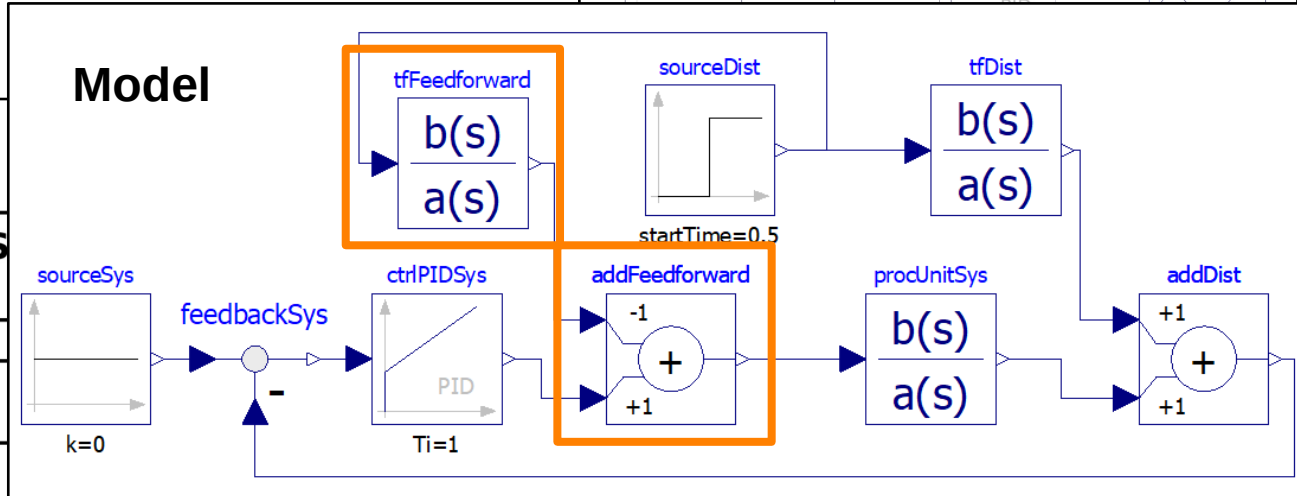
**MB**



*pruning*

**Pruned Entity S**

**Model**





## Python Toolset

- Available:

[https://github.com/cea-wismar/SESMB\\_Inf\\_Python/](https://github.com/cea-wismar/SESMB_Inf_Python/)

- Tools

- SESToPy → SES editor and IDE

- SESViewEl → SES tree viewer

- **SESMoPy** → Model builder 

- 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

The screenshot displays the SESToPy application window. The title bar reads 'SESToPy - SES Tools in Python3 / PyQt5'. The menu bar includes 'File', 'Edit', 'Merge', and 'Transformation'. The toolbar contains icons for 'Open', 'Save', 'Empty Current Model', 'Prune', and 'Flatten'. The main area shows 'Model 1: not saved' and a connection status for 'Server IP 127.0.0.1' and 'Server Port 54545'. A 'Global Settings' panel on the left has 'Information' and 'SES / PES' options. A 'Hierarchy Model' panel shows a tree view with 'Node' selected. A 'Node Specific Properties' panel is on the right. A file explorer dialog box is open, showing the path '< SESMoPy > Examples'. The dialog lists several folders and one file: 'Example01', 'Example02\_FeedbackControl', 'Example03\_FeedbackControl\_FMI', 'Example04\_RLC\_FMI', 'Example05\_MSD\_FMI', 'Example06\_Clock Software Generation', and 'Template\_for\_SESMoPy.jsonsestree'. The file 'Template\_for\_SESMoPy.jsonsestree' is selected. The dialog also shows a 'Dateiname:' field and a file type dropdown set to 'JSON SES Tree (\*.jsonsestree)'. Buttons for 'Öffnen' and 'Abbrechen' are visible at the bottom of the dialog.



# Different Simulators & Interfaces can be set

The screenshot shows the SESToPy application window with the following components:

- Global Settings:** A table for SES Variables with two entries highlighted in orange:

Name	Value	Comment
mysim	"OpenModelica"	
myinterface	"native"	

- Hierarchy Model:** A tree view showing a hierarchy of nodes. The 'simMethod' node under 'expDCC' is highlighted in orange.

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

- Node Specific Properties:** A table for attributes with two entries highlighted in orange:

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

At the bottom of the window, the status bar displays: 2021-02-02 / 20:54:15 / Version 2021.02.02 / File: C:/Users/Win10/SESMB\_Infrastructure/SESMoPy/Examples/Template\_for\_SESMoPy.jsonsestree / Last saved: 2020-09-17 - 21:54:54



# Merge Feedback SES to Provisional Experimental Frame

The screenshot shows the SESToPy application window. The 'Merge' menu is open, with 'Add SubSES' highlighted. The main window displays a 'Hierarchy Model' tree with nodes like 'exp', 'expDEC', 'simMethod', 'ctrlSys', and 'expivemethod'. A table of 'SES Variables' is visible on the left, and a 'Node Specific Properties' panel is on the right.

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

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



# Merge Feedback SES to Provisional Experimental Frame (2)

The screenshot shows the SESToPy software interface. On the left, the 'Global Settings' panel contains a table of 'SES Variables':

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

The 'Hierarchy Model' panel shows a tree structure of nodes. A blue box highlights the following nodes:

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

The 'Node Specific Properties' panel on the right shows a table of attributes for the selected node:

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

At the bottom of the interface, the status bar displays: 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





# Prune & Flatten for feedforward=0

The screenshot shows the SESToPy application window. The menu bar includes 'File', 'Edit', 'Merge', and 'Transformation'. The 'Transformation' menu is open, showing 'Prune' and 'Flatten' options, which are highlighted with an orange box. The main workspace is titled 'Model 1: Template\_for\_SESMoPy'. It features a 'Global Settings' panel on the left with a table of SES Variables:

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

The 'Hierarchy Model' panel in the center shows a tree structure of nodes. The selected node is 'Entity Node'. The tree includes nodes like 'exp', 'expDEC', 'simMethod', 'ctrlSys', 'ctrlSysDEC', 'feedforwardCtrl', 'feedforwardCtrlSPEC', 'fc', 'fcDEC', 'tfFeedforward', 'addFeedforward', 'NONE', 'sourceSys', 'feedbackSys', 'ctrlPIDSys', 'procUnitSys', 'sourceDist', 'tfDist', 'addDist', and 'expMethod'. Each node has associated 'Type', 'MB', and 'atr' values.

The 'Node Specific Properties' panel on the right shows a table of attributes for the selected node:

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



# Flattened PES for feedforward=0

The screenshot shows the SESToPy application window. The 'Global Settings' panel on the left has 'flattened PES' selected under the 'SES / PES' section. The 'Hierarchy Model' panel in the center displays a tree structure with a table of nodes. The 'Node Specific Properties' panel on the right shows attributes for the selected node.

Tree	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			

Name	Value	var/fun	comment
PARAMVARY1	"ctrlPIDSys.k=[1,5]"		template: ...
PARAMVARY2	"ctrlPIDSys.Ti=[1,0,5]"		template: ...



# Save Flattened PES for feedforward=0

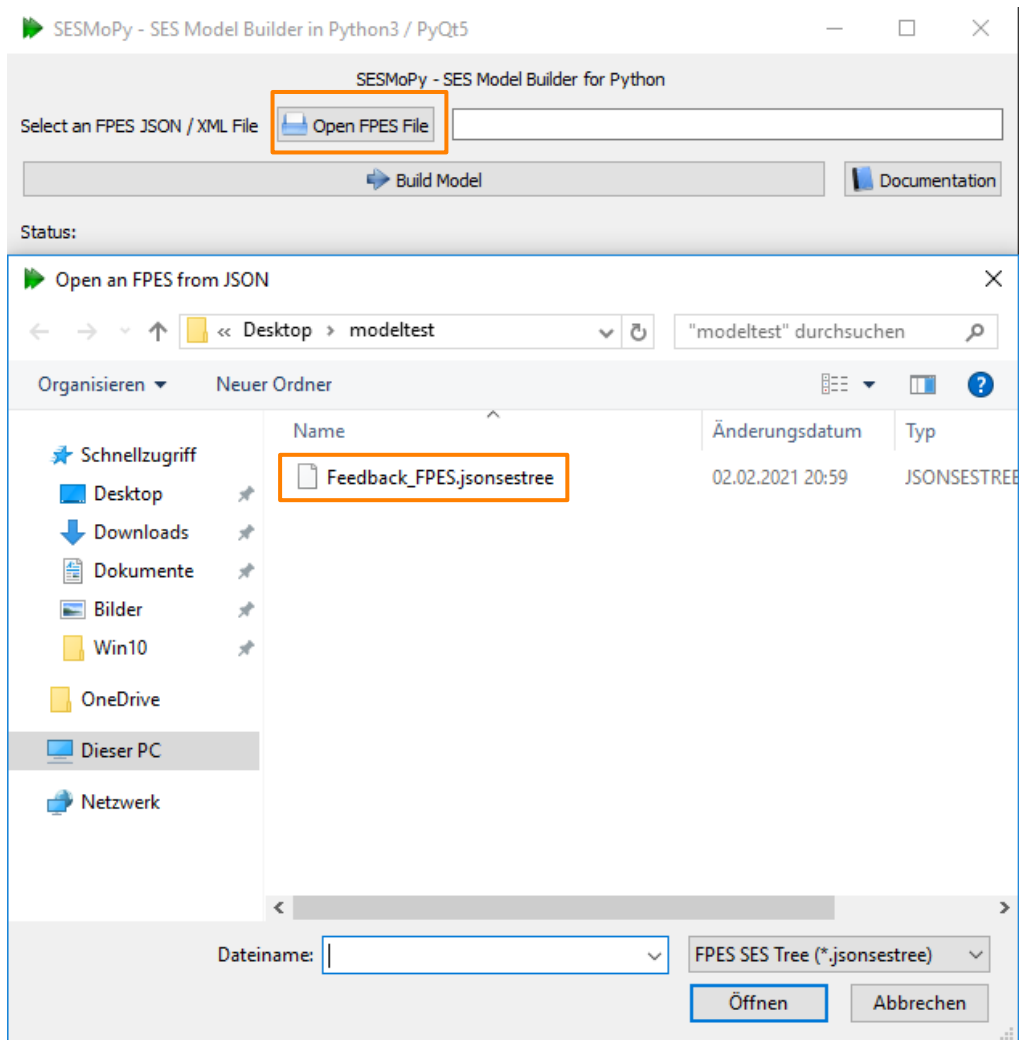
The screenshot displays the SESToPy application window. The top toolbar contains buttons for 'Open', 'Save', 'Empty Current Model', 'Prune', and 'Flatten'. The 'Save' button is highlighted with an orange box. A 'Save an SES as JSON' dialog box is open, showing the file path 'Dieser PC > Desktop > modeltest'. The file name 'Feedback\_FPES' is entered in the 'Dateiname:' field, and the file type is 'JSON SES Tree (\*.jsonsestree)'. The background shows the 'Global Settings' panel with 'flattened PES' selected under 'SES / PES'. The 'Model 3: not saved' tab is active, and the 'Global Specific Properties' panel is visible on the right.

Name	Value	var/fun	comment
1 PARAMVARY1	"ctrlPIDSys.k=[1,5]"		template: ...
2 PARAMVARY2	"ctrlPIDSys.Ti=[1,0,5]"		template: ...





# Open Flattened PES in SESMoPy





# Add MB in Modelfolder & Build Model

The screenshot displays two overlapping windows. The top window is the 'SESMoPy - SES Model Builder for Python' application. It features a text input field for 'Select an FPES JSON / XML File' containing the path 'C:/Users/Win 10/Desktop/modeltest/Feedback\_FPES.jsonsestree'. Below this is a 'Build Model' button, which is highlighted with an orange rectangular border. To the right of the 'Build Model' button is a 'Documentation' button. The bottom window is a Windows File Explorer showing the 'modeltest' folder. The file list contains two items: 'Feedback\_FPES.jsonsestree' and 'MB.mo'. The 'MB.mo' file is selected and highlighted with a blue background, and its name is also highlighted with an orange rectangular border. The status bar at the bottom of the File Explorer indicates '2 Elemente' and '1 Element ausgewählt (505 Bytes)'.

Name	Änderungsdatum	Typ
Feedback_FPES.jsonsestree	02.02.2021 20:59	JSONSESTREE
MB.mo	28.11.2018 21:18	MO-Datei



# Created Models

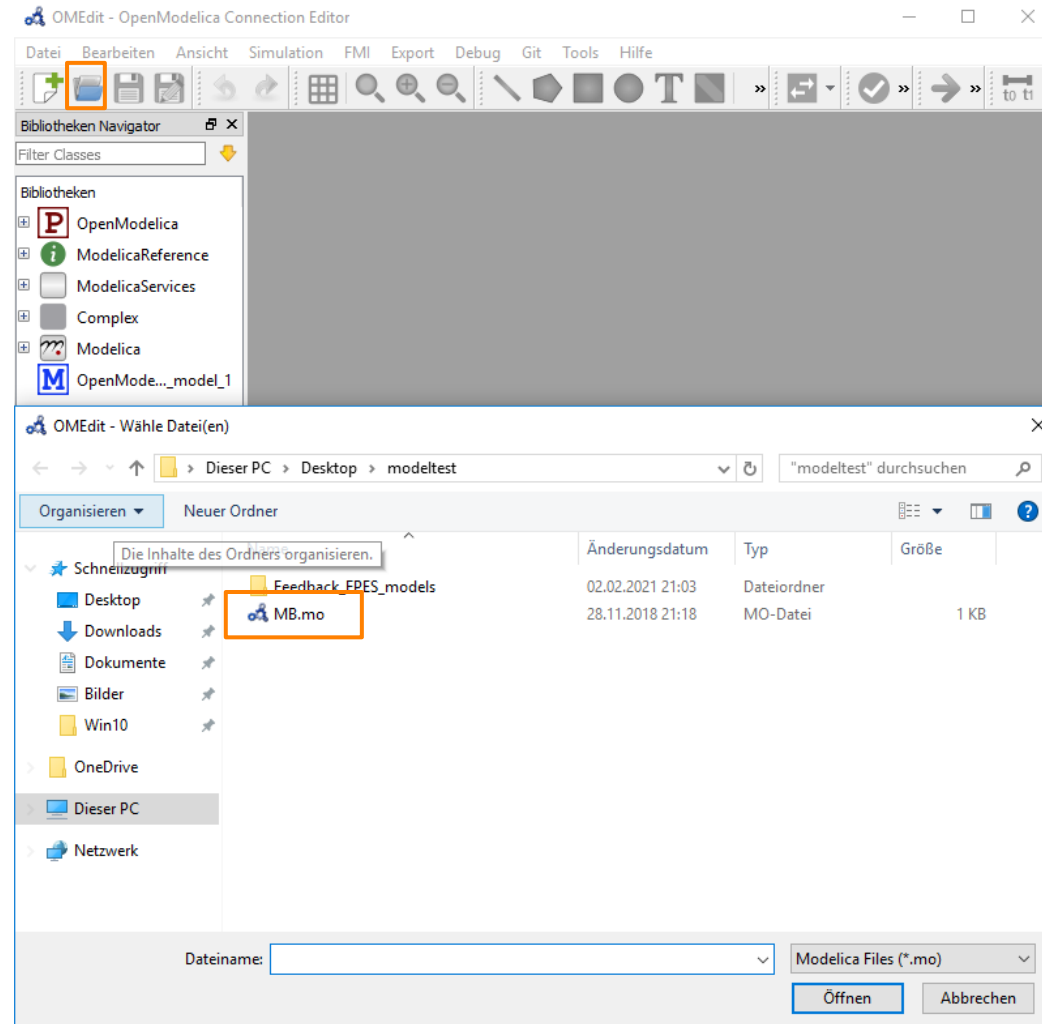
The screenshot shows a Windows File Explorer window titled 'Feedback\_FPES\_models'. The address bar indicates the path 'modeltest > Feedback\_FPES\_models'. The left sidebar shows the 'Schnellzugriff' (QuickTime) section with 'Dieser PC' selected, showing 4 elements. The main pane displays a list of files with columns for Name, Änderungsdatum (Change Date), and Typ (Type). The files listed are:

Name	Änderungsdatum	Typ
config.txt	02.02.2021 21:03	Textdokumen
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

The two MO files are highlighted with an orange box. The ribbon at the top includes tabs for 'Start', 'Freigeben', and 'Ansicht', and groups for 'Zwischenablage', 'Organisieren', 'Neu', and 'Öffnen'.



# Open Created Models in OpenModelica & Load MB





# Model & Setup Simulation in OpenModelica

The image shows two overlapping windows from the OpenModelica software. The top window is the 'OMEdit - OpenModelica Connection Editor' showing a library browser on the left and a code editor on the right. The code editor contains the following Modelica code:

```
1 model OpenModelica_Feedback_FPES_model_1
2   MB.Constant sourceSys(k=0);
3   MB.Feedback feedbackSys();
4   MB.PID ctrlPIDSys(k=1, Ti=1, Td=0);
5   MB.TransferFunction procUnitSys(b={1}, a={20, 1});
6   MB.Step sourceDist(startTime=0.5);
7   MB.TransferFunction tfDist(b={1}, a={10, 1});
8   MB.Add addDist();
9   equation
10  connect(sourceSys.y, feedbackSys.u1);
11  connect(feedbackSys.y, ctrlPIDSys.u);
12  connect(procUnitSys.y, addDist.u2);
13  connect(addDist.y, feedbackSys.u2);
14  connect(sourceDist.y, tfDist.u);
15  connect(tfDist.y, addDist.u1);
16  connect(ctrlPIDSys.y, procUnitSys.u);
17 end OpenModelica_Feedback_FPES_model_1;
```

The bottom window is the 'Simulation Setup - OpenModelica\_Feedback\_FPES\_model\_1' dialog. It has tabs for 'Allgemeine', 'Ausgabe', 'Simulationsflags', and 'Archived Simulations'. The 'Allgemeine' tab is active, showing simulation interval settings:

- Startzeit: 0
- Stoppzeit: 50
- Number of steps: 500 (selected)
- Interval: 0.002

Integration settings include:

- Methode: dassl
- Toleranz: 1e-6
- Jacobi-Matrix: (dropdown)
- DASSL/IDA Options:  Root Finding,  Restart After Event
- Initial Step Size: (input field)
- Maximum Step Size: (input field)

At the bottom, there are checkboxes for 'Save experiment annotation inside model', 'Save \_\_OpenModelica\_simulationFlags annotation inside model', and 'Simulieren' (checked). 'OK' and 'Abbrechen' buttons are at the bottom right.



# OpenModelica Simulation Results

OMEdit - OpenModelica Connection Editor

File Bearbeiten Ansicht Simulation FMI Export Debug Git Tools Hilfe

Bibliotheken Navigator

Filter Classes

Bibliotheken

- OpenModelica
- ModelicaReference
- ModelicaServices
- Complex
- Modelica
- OpenMode...\_model\_1
- MB

Plot : 1

Auto Scale Fit in View Save Print Grid Detailed Grid No Grid

sourceSys.y sourceDist.y addDist.y

time (s)

Meldungen

[1] 21:07:42 Übersetzung Warnung  
Assuming fixed start value for the following 3 variables:  
tfDist.x\_scaled[1]:VARIABLE(protected = true) "Scaled vector x" type: Real [1]  
procUnitSys.x\_scaled[1]:VARIABLE(protected = true) "Scaled vector x" type: Real [1]  
ctrlPIDSys.D.x:VARIABLE(start = ctrlPIDSys.D.x\_start) "State of block" type: Real [1]

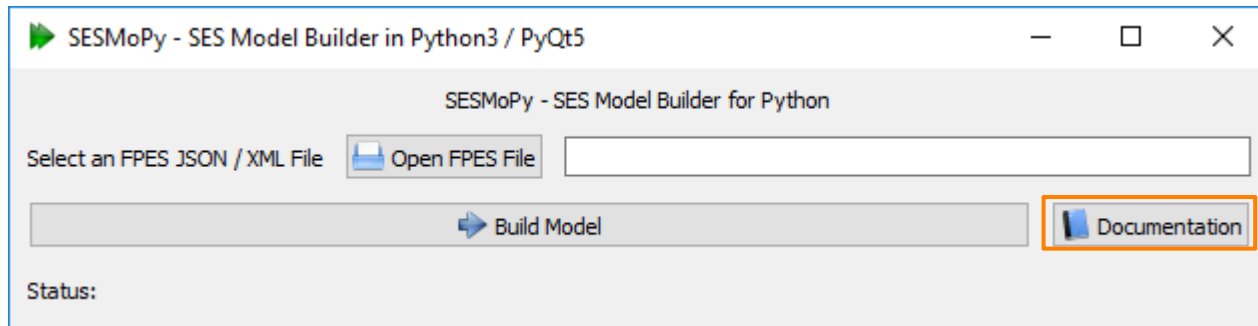
Variablen

- OpenMo...odel\_1
  - addDist
    - k1
    - k2
    - u1
    - u2
    - y
  - ctrlPIDSys
  - feedbackSys
  - procUnitSys
  - sourceDist
    - height
    - offset
    - startTime
    - y
  - sourceSys
    - k
    - y

Willkommen Modellieren Plotten Debugging



# 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) <sup>1,2</sup>

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

<sup>2</sup>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) <sup>1,2</sup>

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

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

<sup>2</sup>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) <sup>1,2</sup>

- FMI defines a standardized interface of components (models, blocks)
  - Reuse of components
    - (i) For **model exchange**
    - (ii) For co-simulation

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

<sup>2</sup>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) <sup>1,2</sup>

- 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

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

<sup>2</sup>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) <sup>1,2</sup>

- 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

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

<sup>2</sup>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) <sup>1,2</sup>

- 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

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

<sup>2</sup>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)

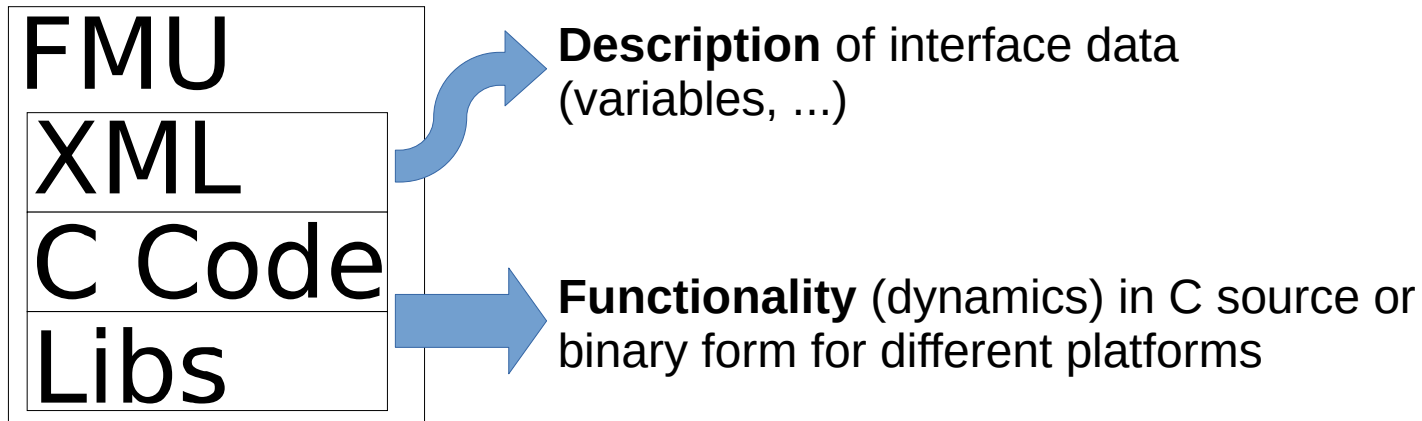
Component implementing FMI = Functional Mock-up Unit (FMU)  
a zipped file with fileextension .fmu



## Functional Mock-up Unit (FMU)

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

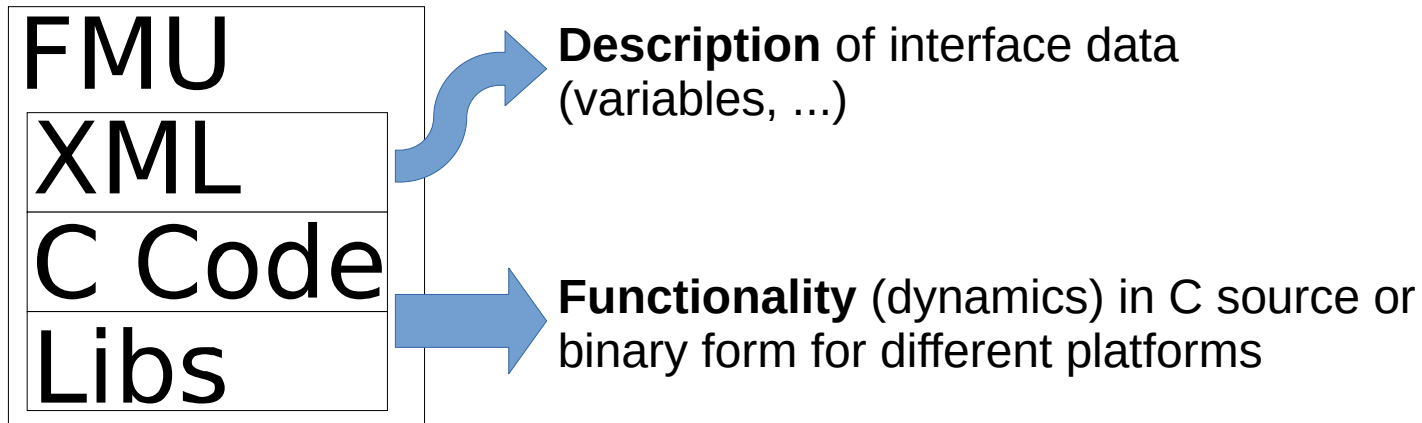
a zipped file with fileextension .fmu





# Functional Mock-up Unit (FMU)

Component implementing FMI = Functional Mock-up Unit (FMU)  
a zipped file with fileextension .fmu



Model Exchange

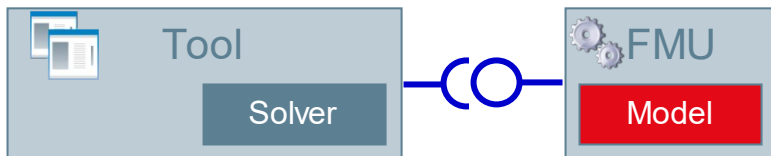


Figure taken from the FMI presentation on the website [www.fmi-standard.org](http://www.fmi-standard.org).



# 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

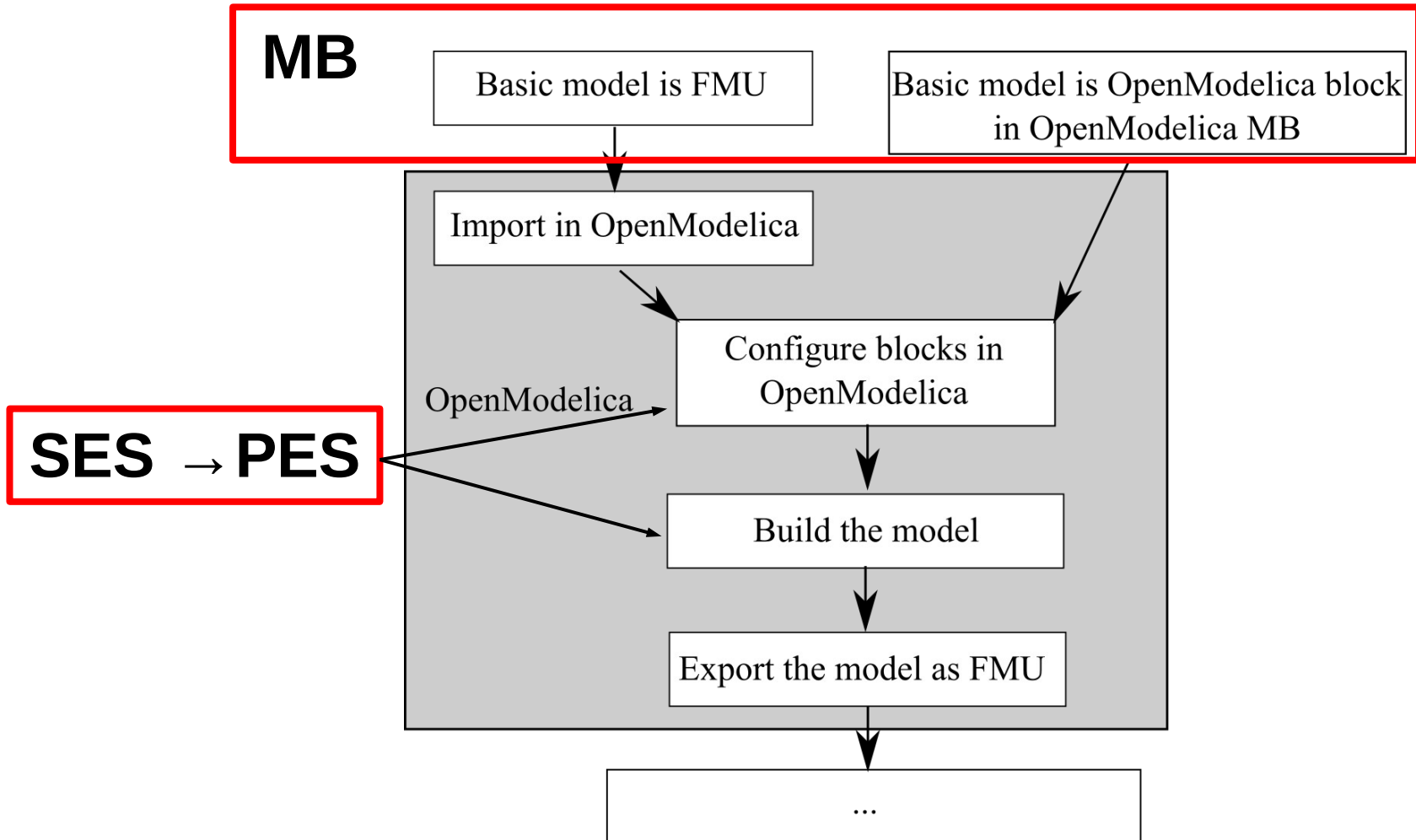


# Model Building – Support Different Simulators Using FMI

- **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



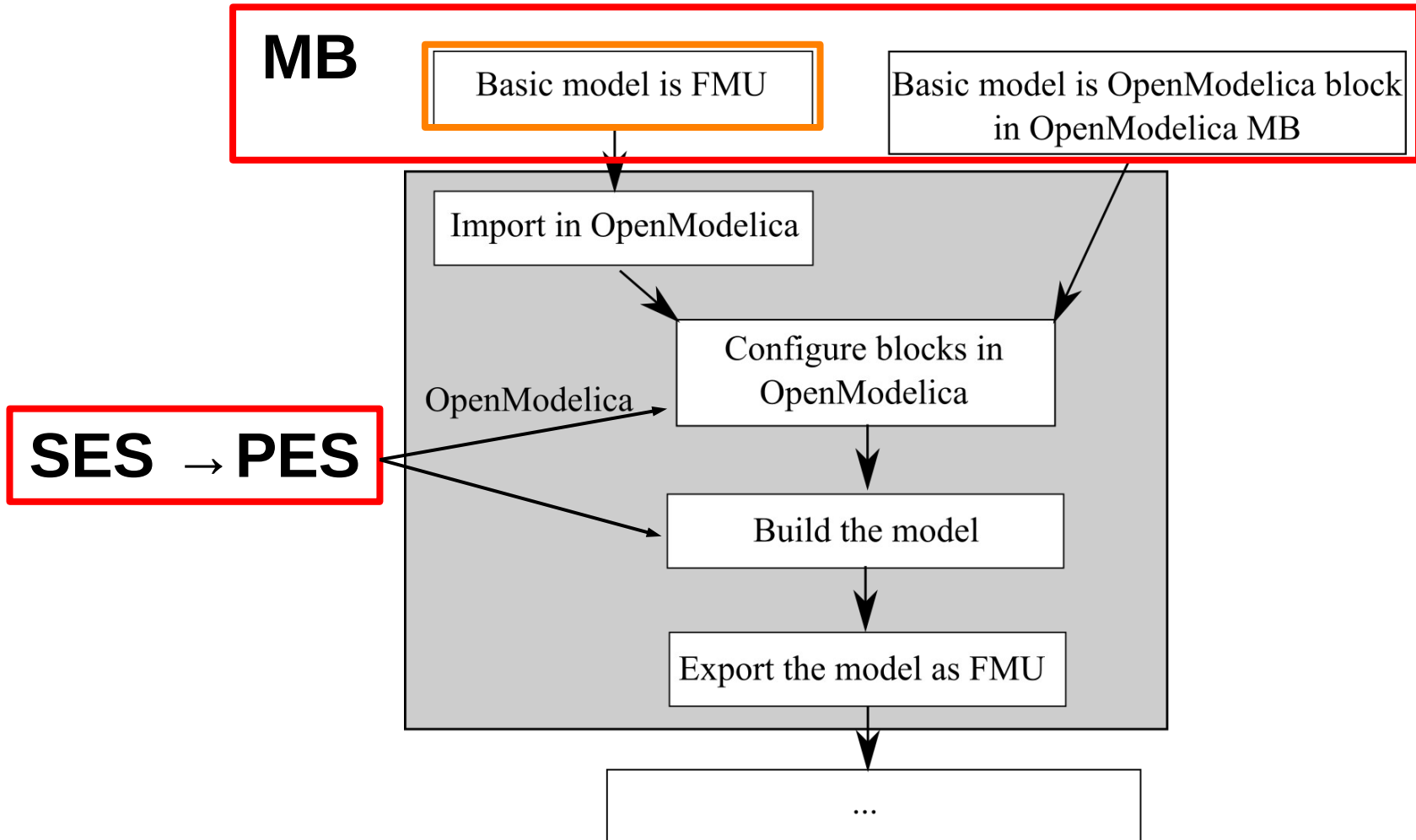
# Model Building Using FMI





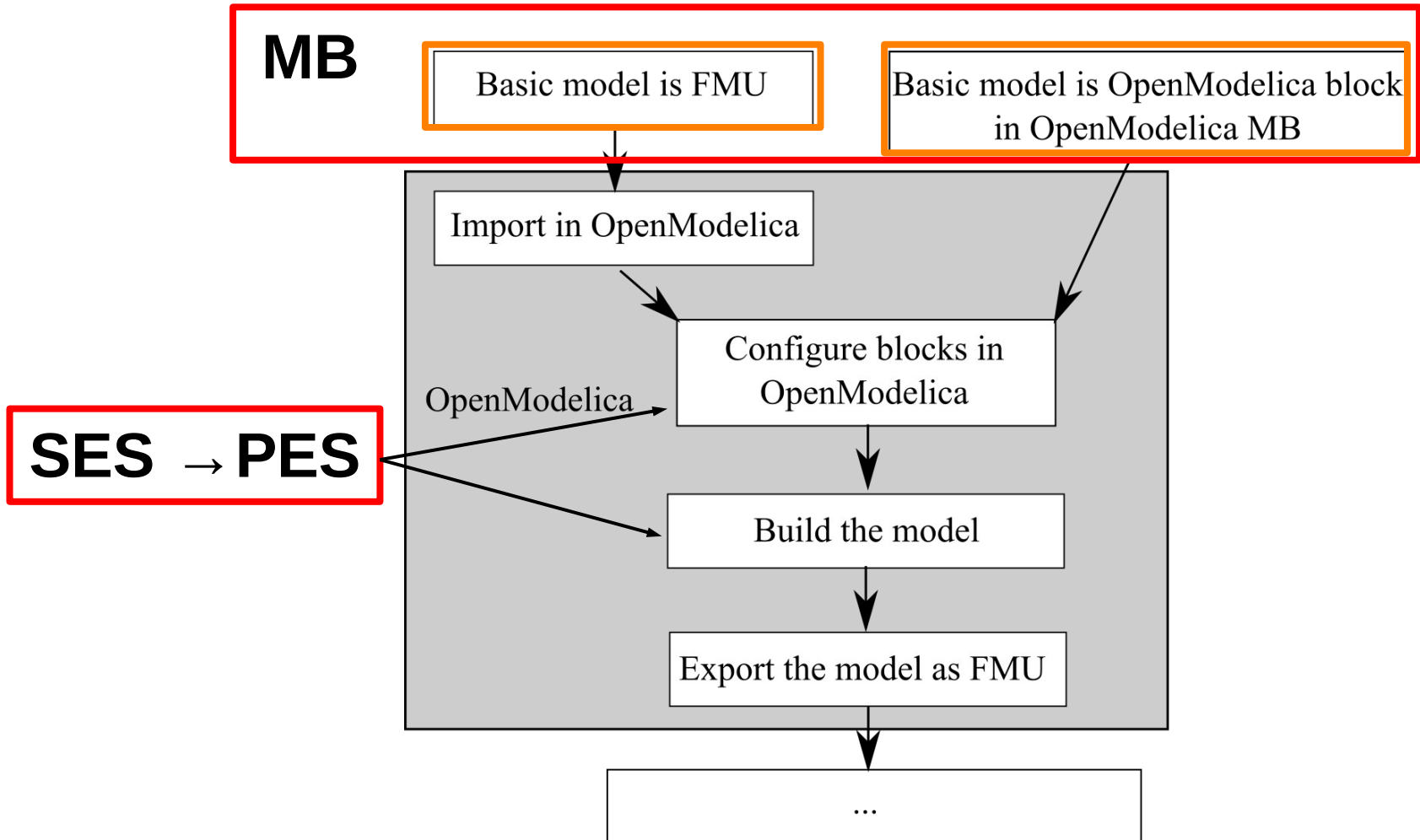


# Model Building Using FMI



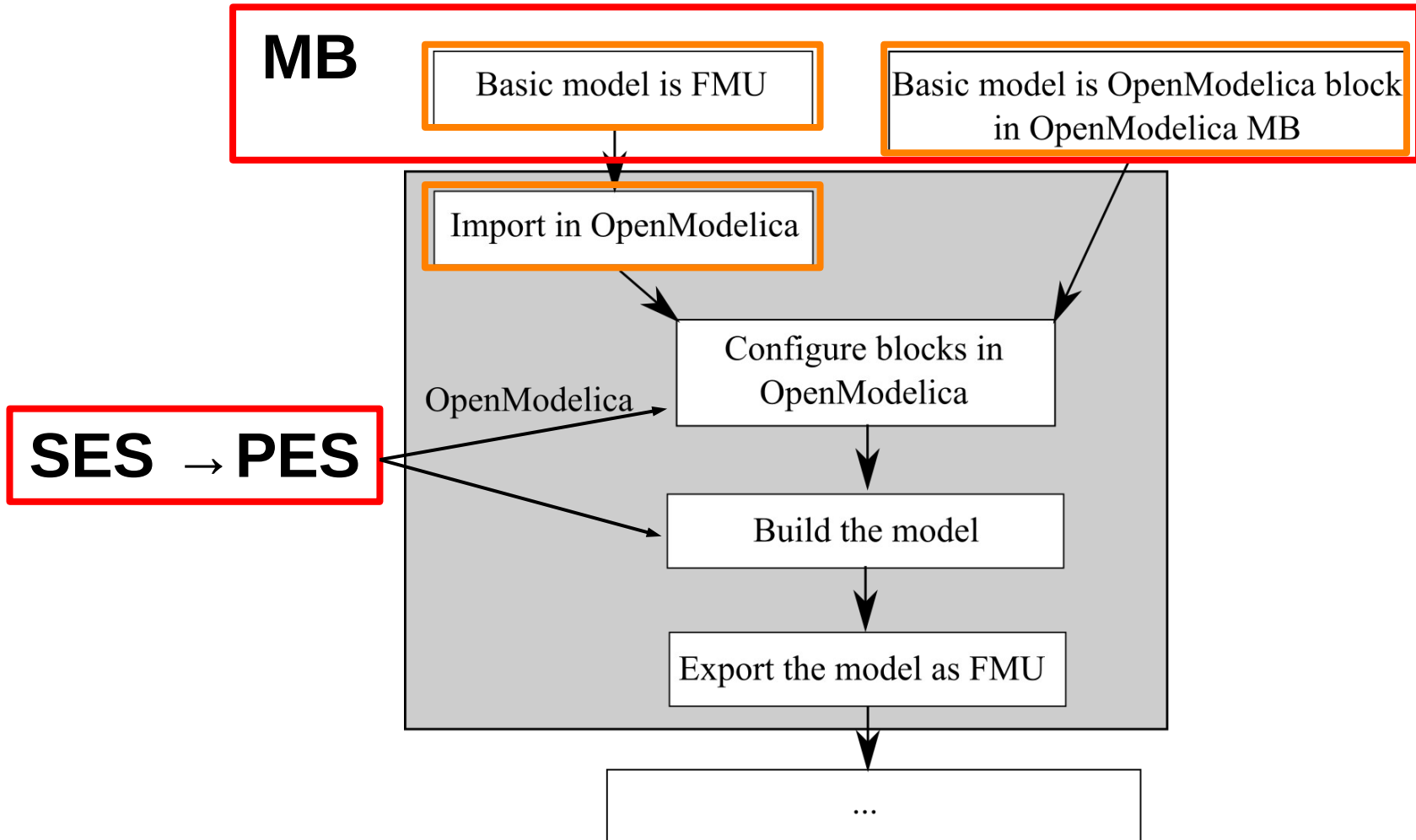


# Model Building Using FMI



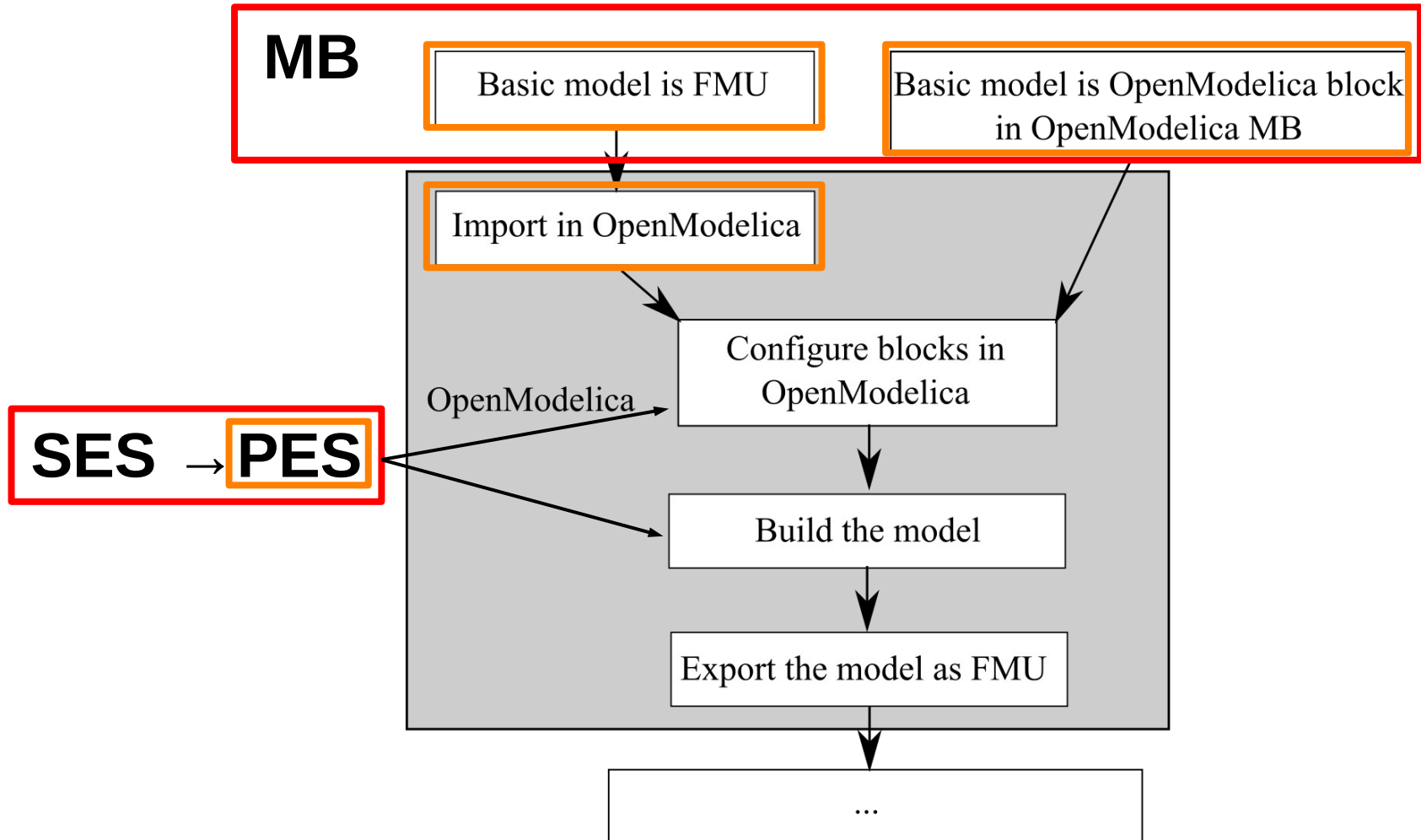


# Model Building Using FMI



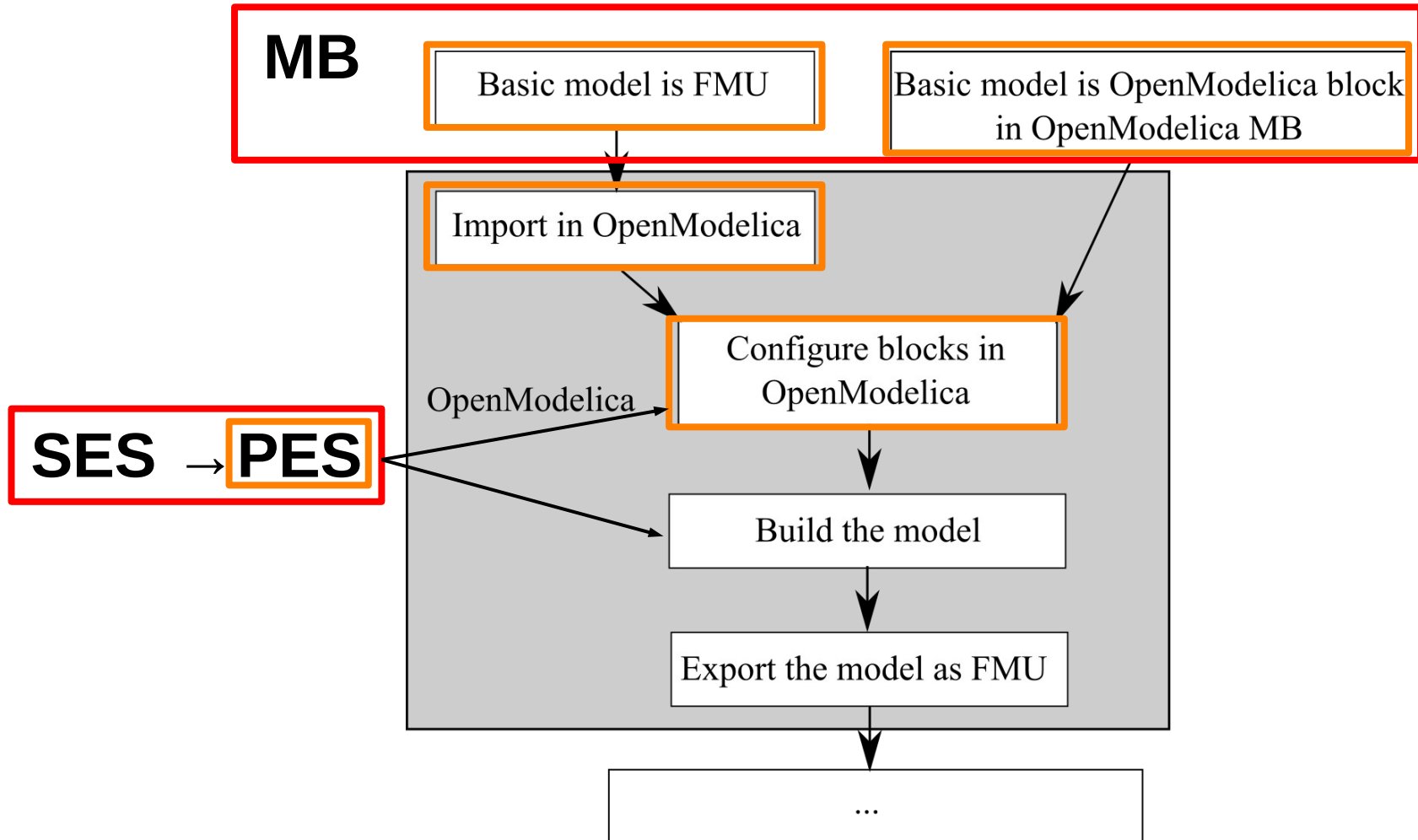


# Model Building Using FMI



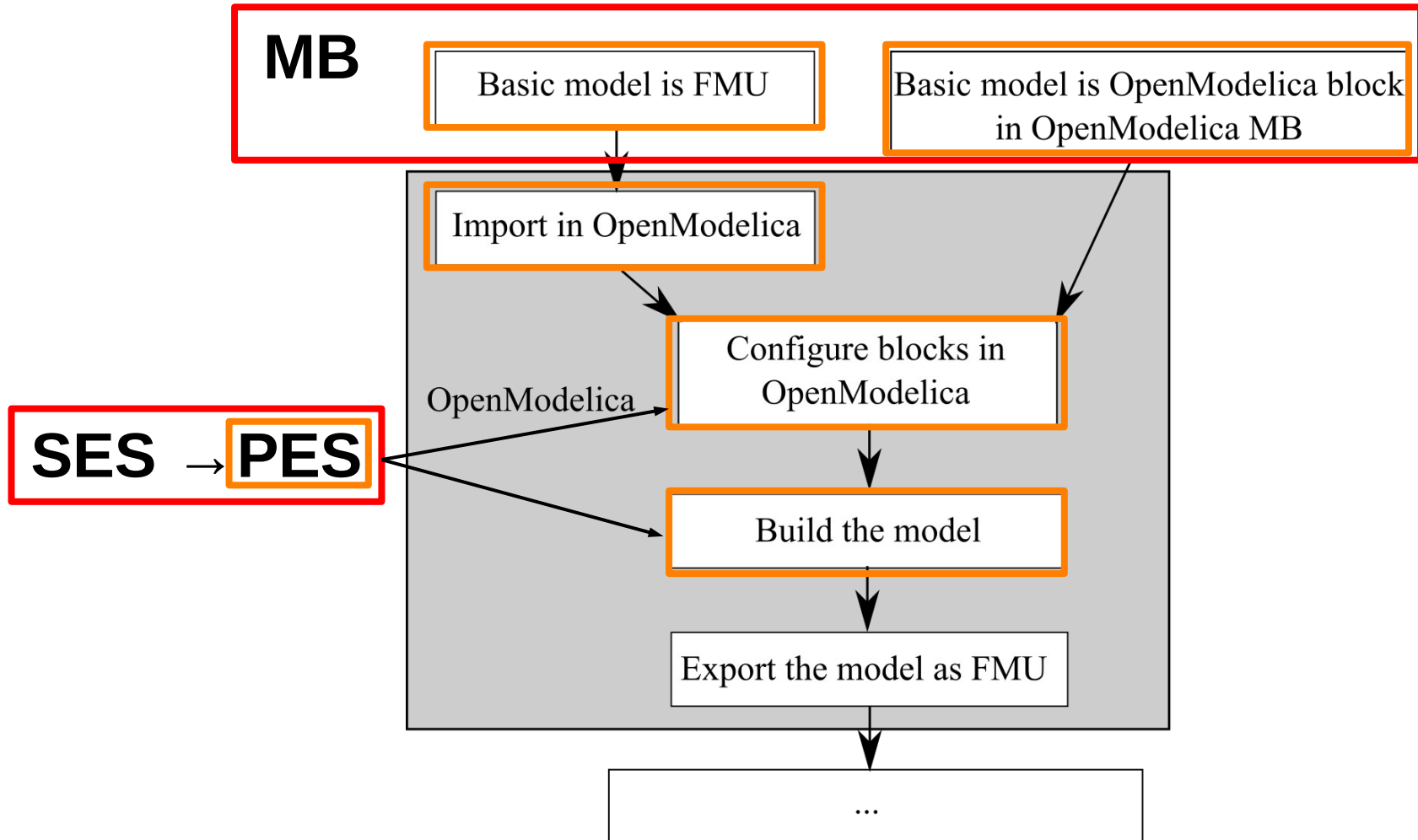


# Model Building Using FMI



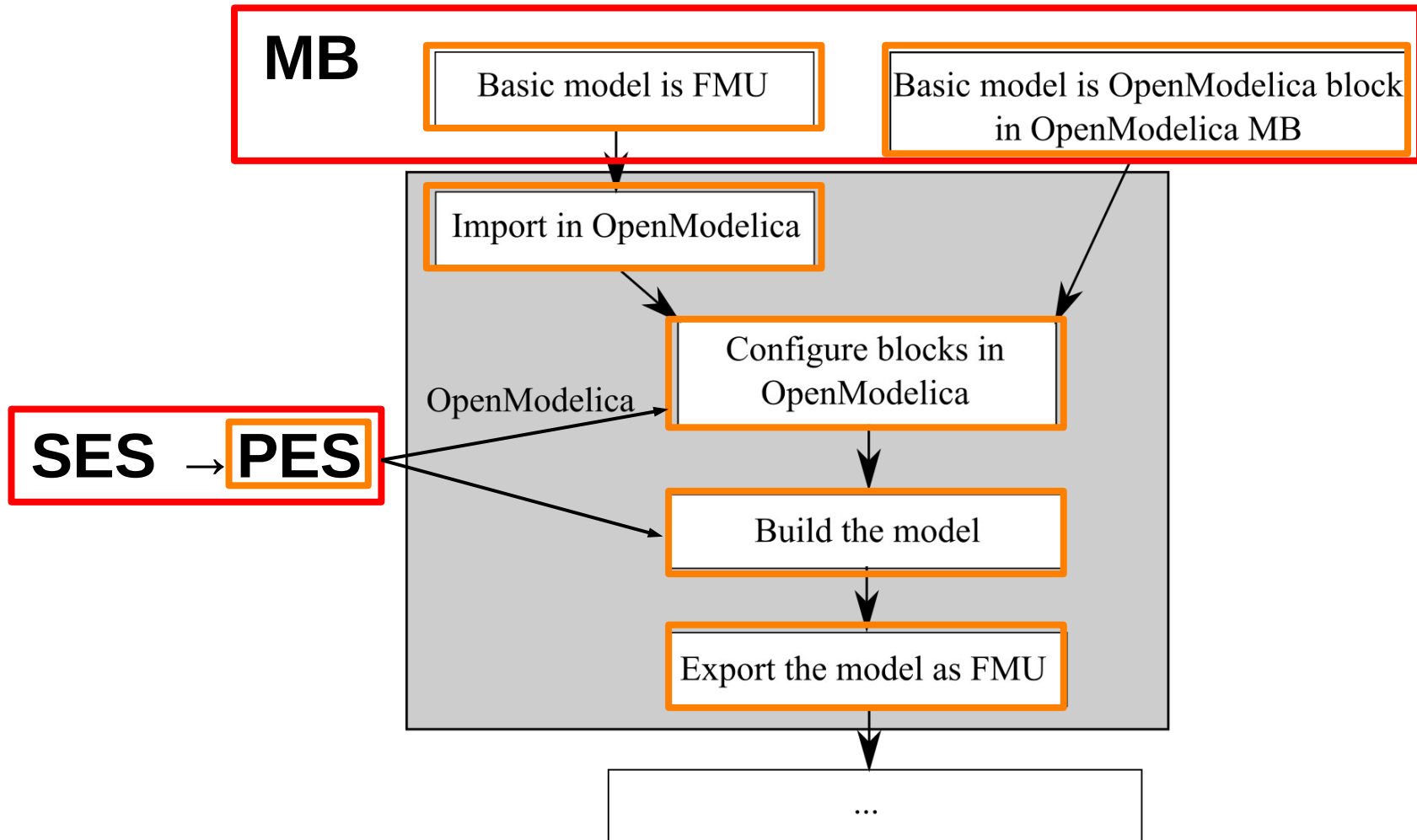


# Model Building Using FMI



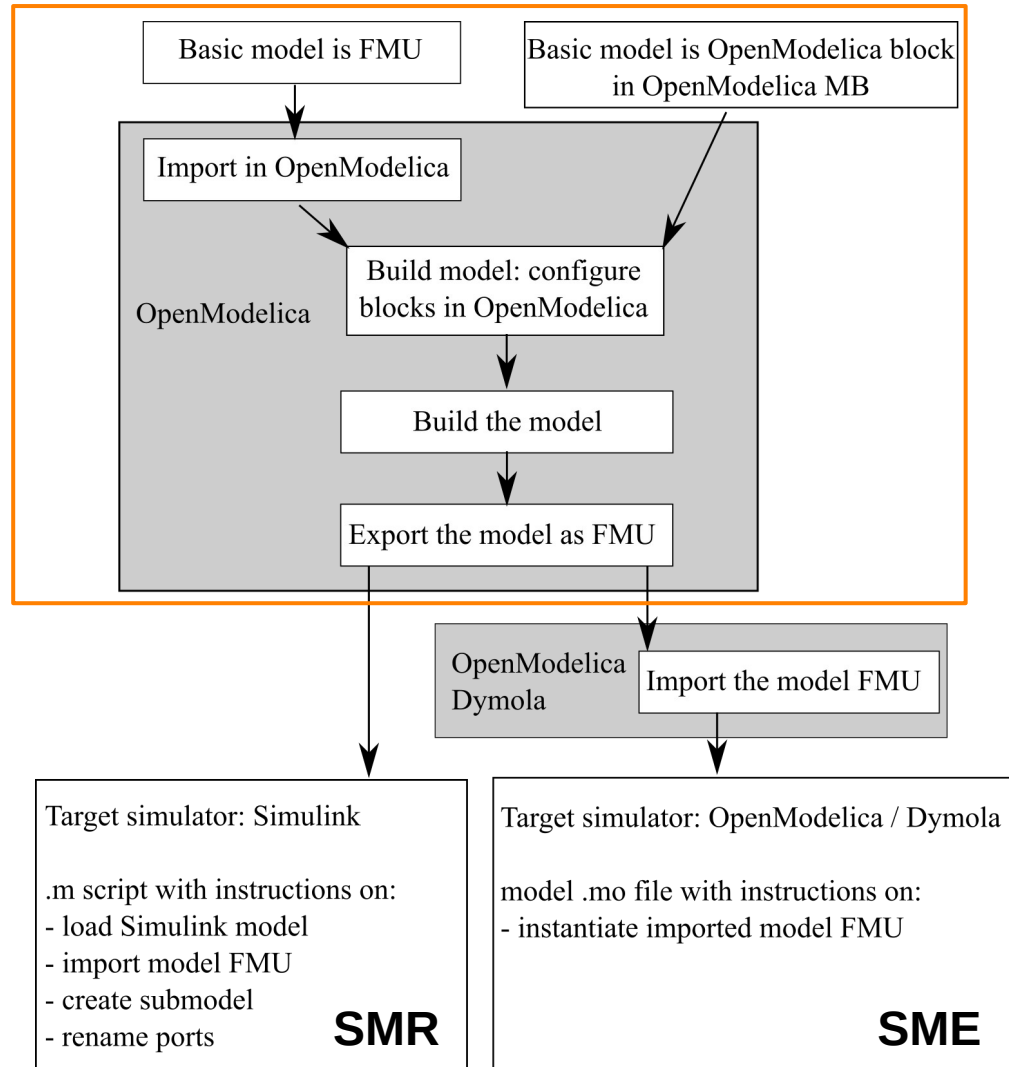


# Model Building Using FMI





# Model Building Using FMI (2)

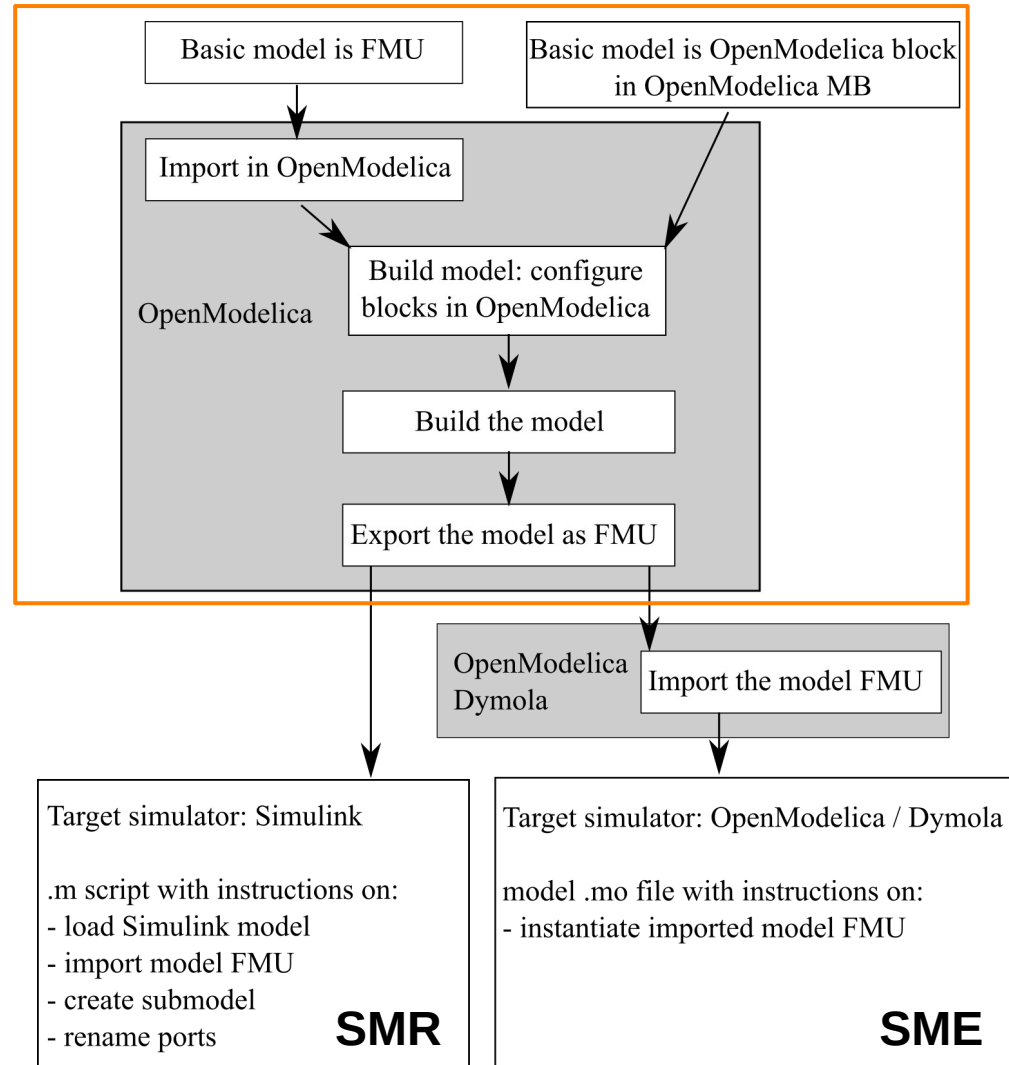






## Model Building Using FMI (2)

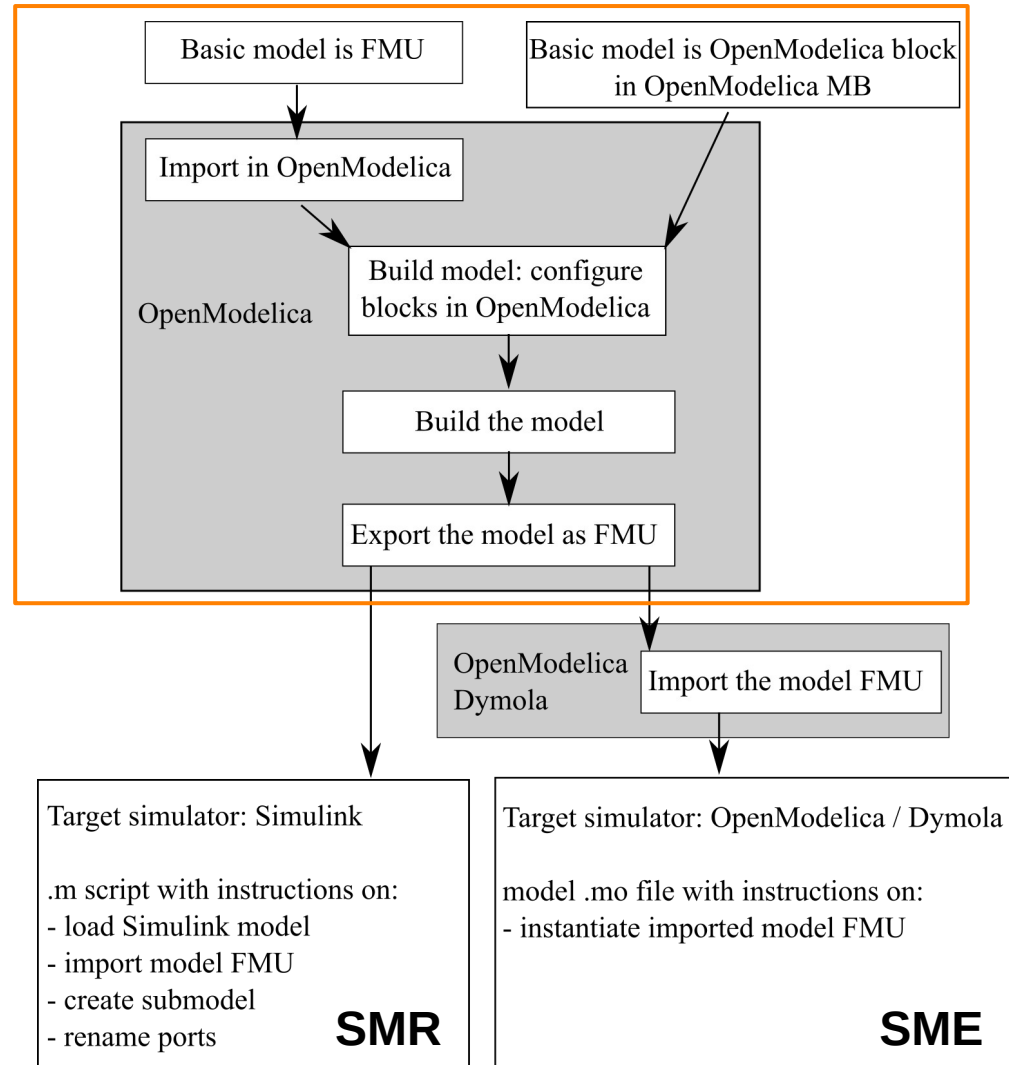
- Create simulator specific instructions on how to execute the model FMU





## Model Building Using FMI (2)

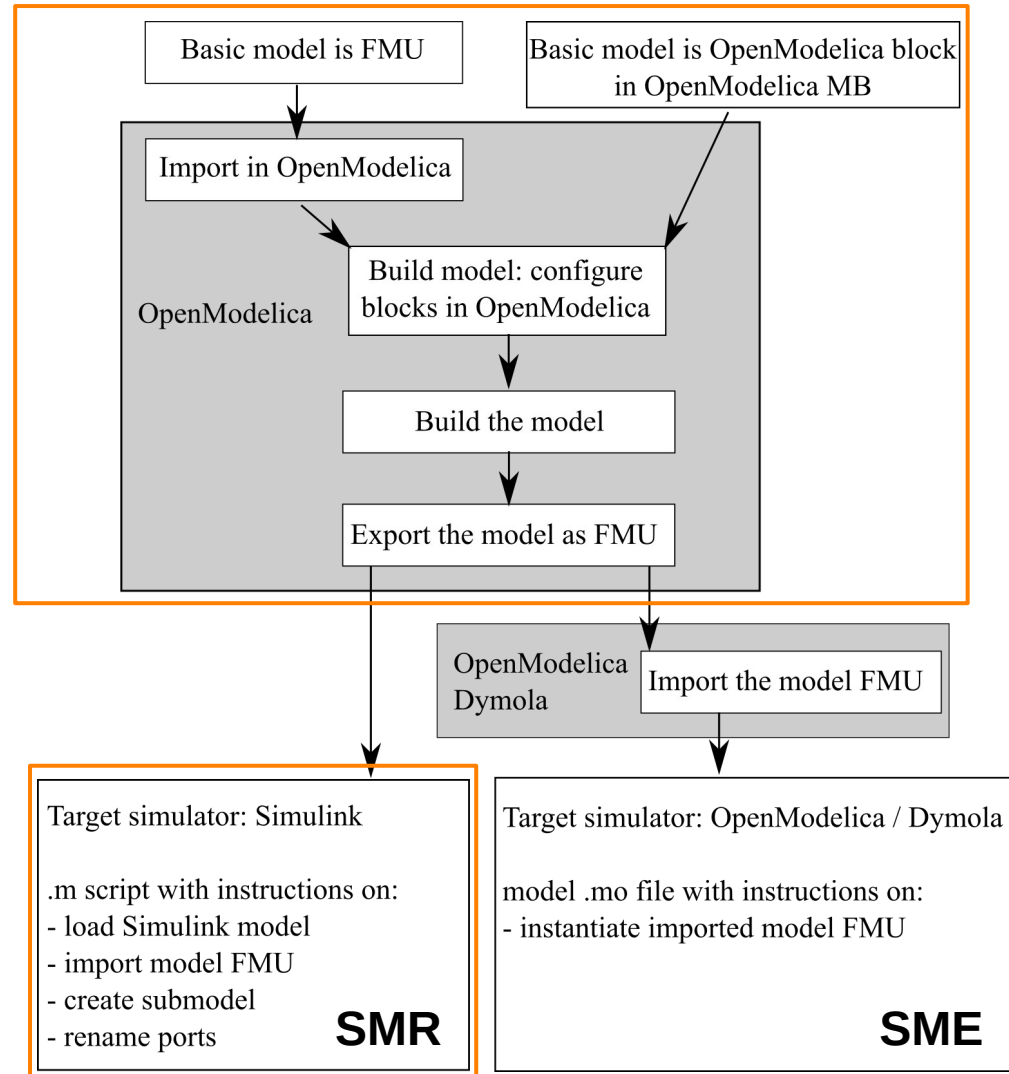
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **”Simulation Model Representation” (SMR)**





## Model Building Using FMI (2)

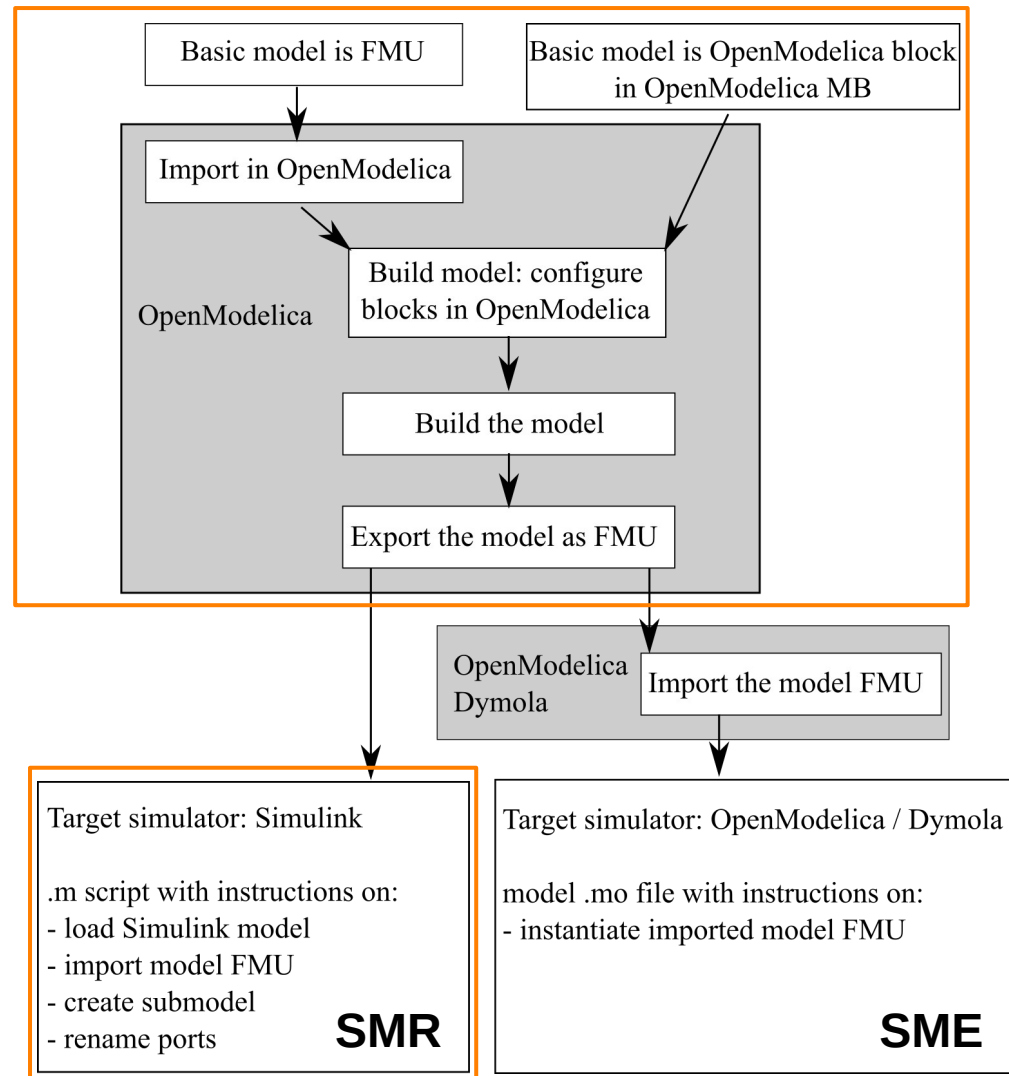
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **”Simulation Model Representation” (SMR)**





## Model Building Using FMI (2)

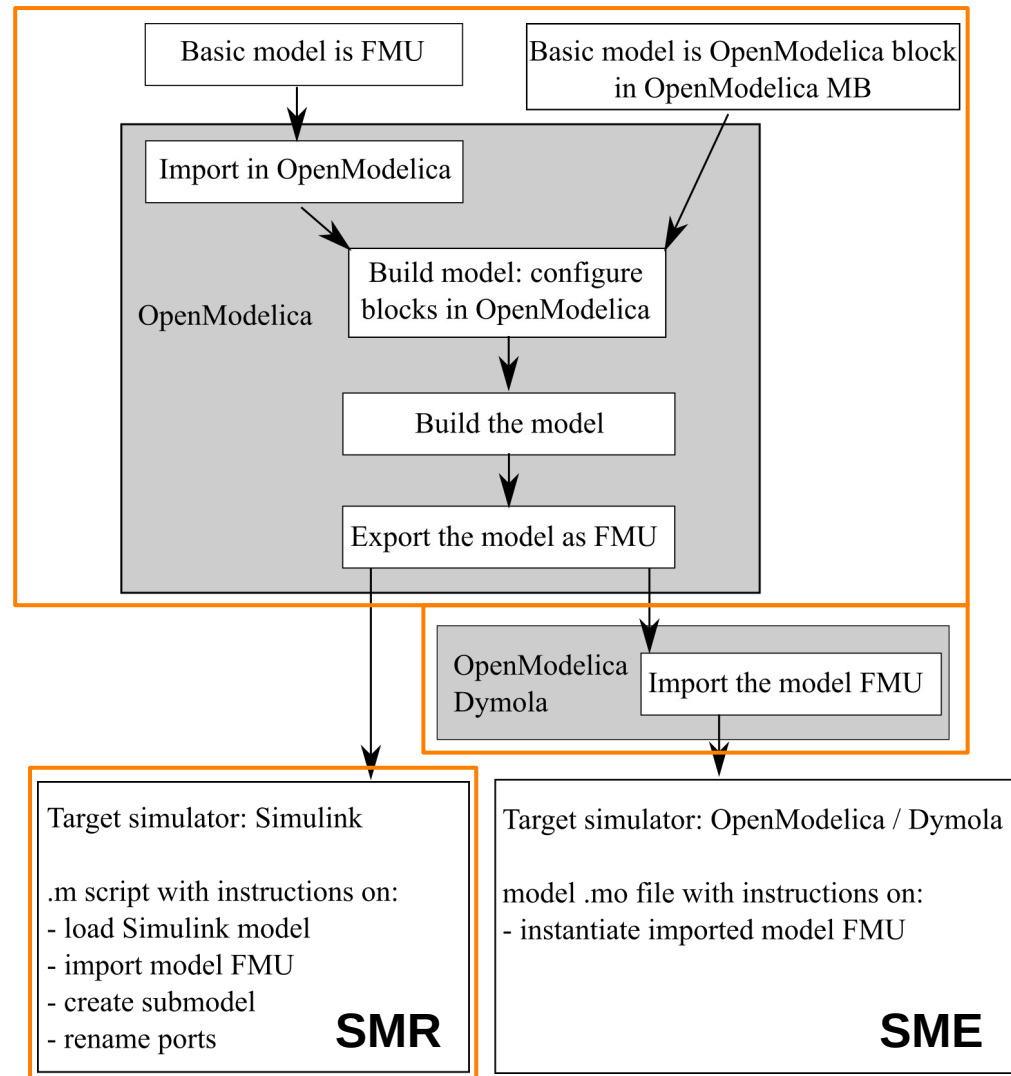
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **“Simulation Model Representation” (SMR)**
- *OpenModelica / Dymola* models are textfiles defining the executable model → **“Simulation Model Executable”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





## Model Building Using FMI (2)

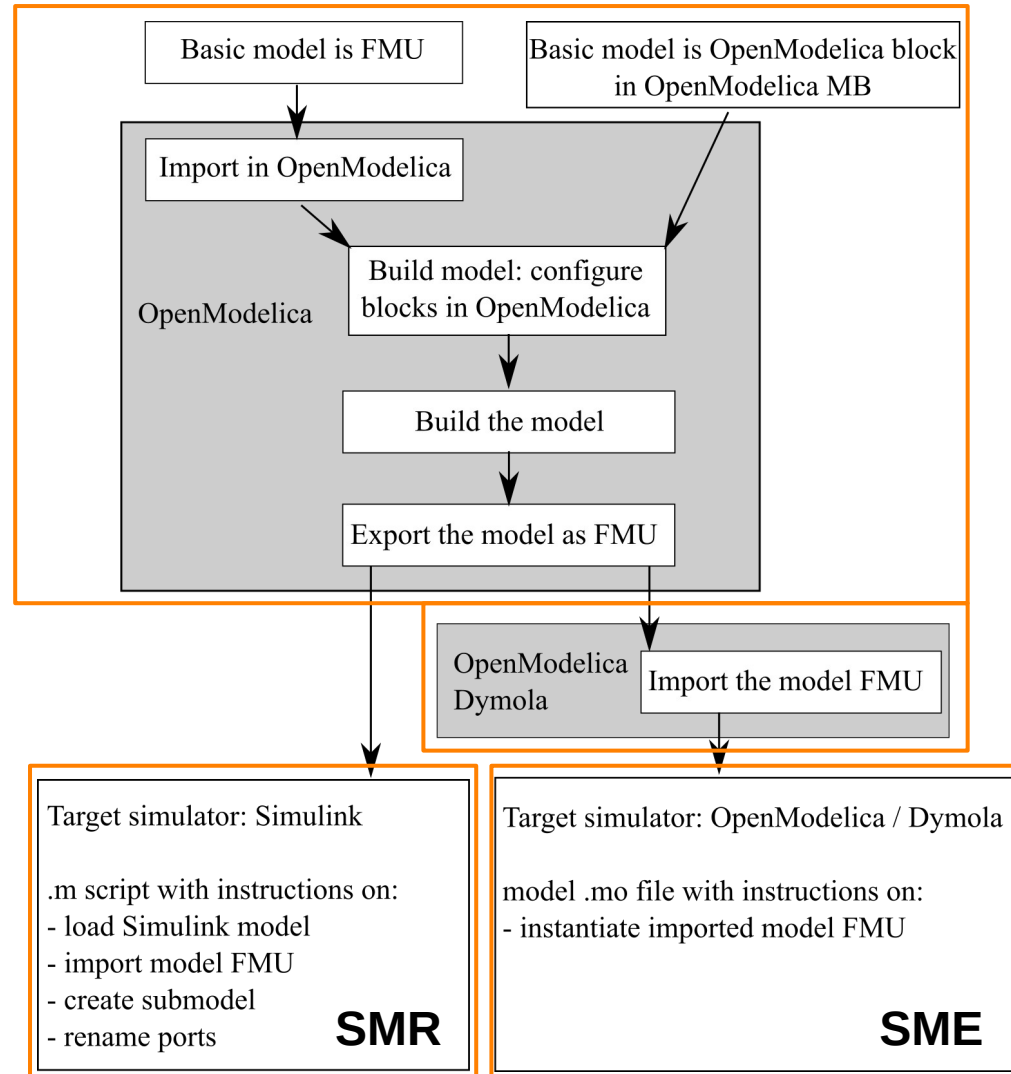
- **Create simulator specific instructions on how to execute the model FMU**
- *Simulink* models can be created and manipulated with a Matlab script → **“Simulation Model Representation” (SMR)**
- *OpenModelica / Dymola* models are textfiles defining the executable model → **“Simulation Model Executable”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





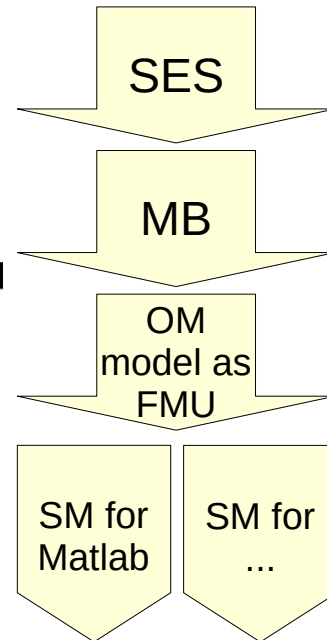
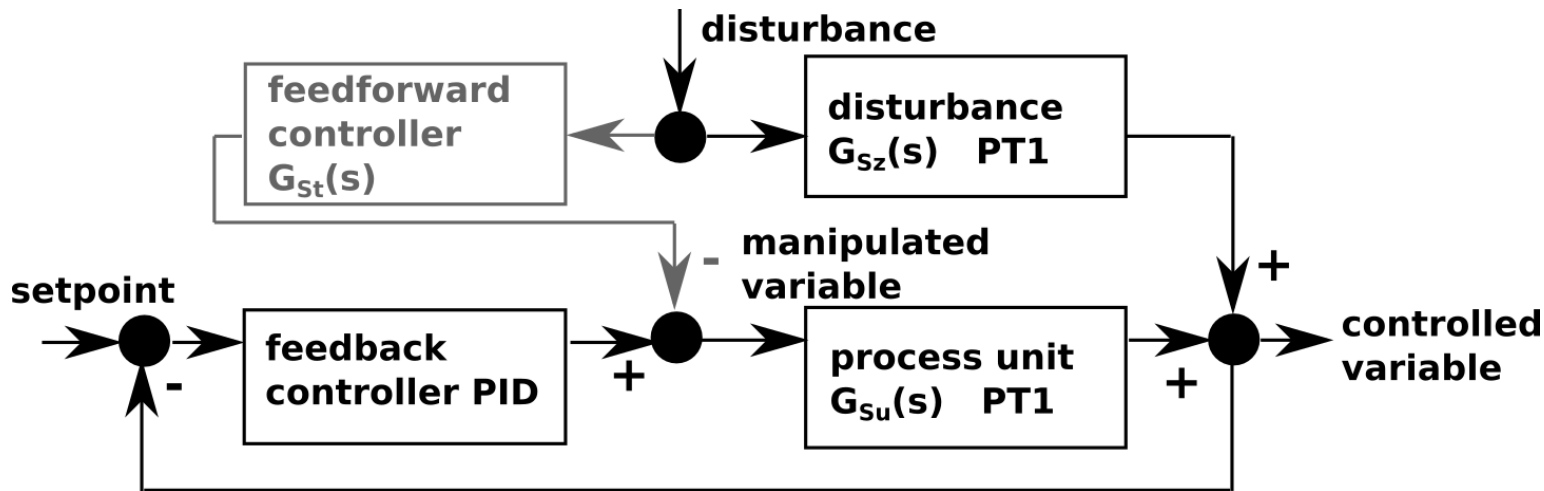
## Model Building Using FMI (2)

- **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)**





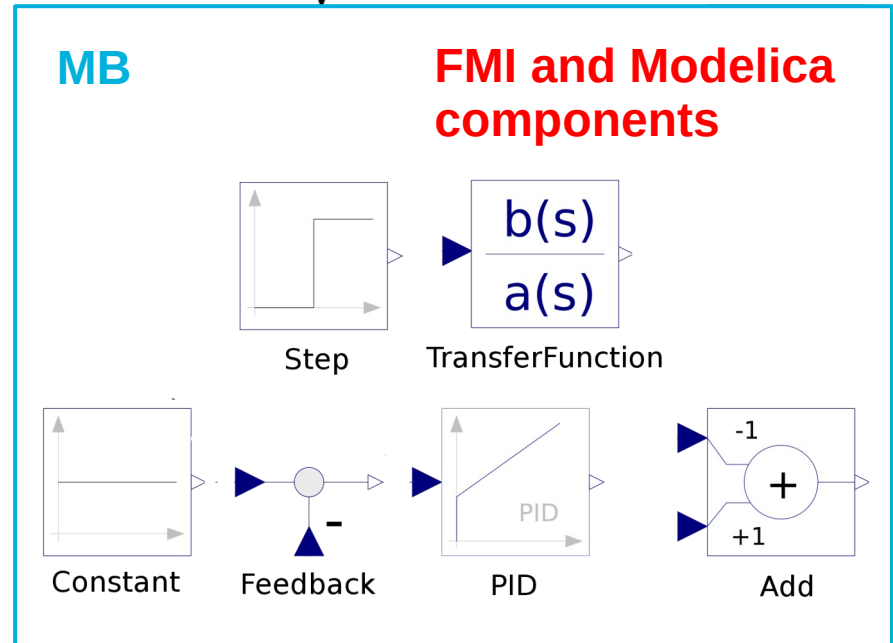
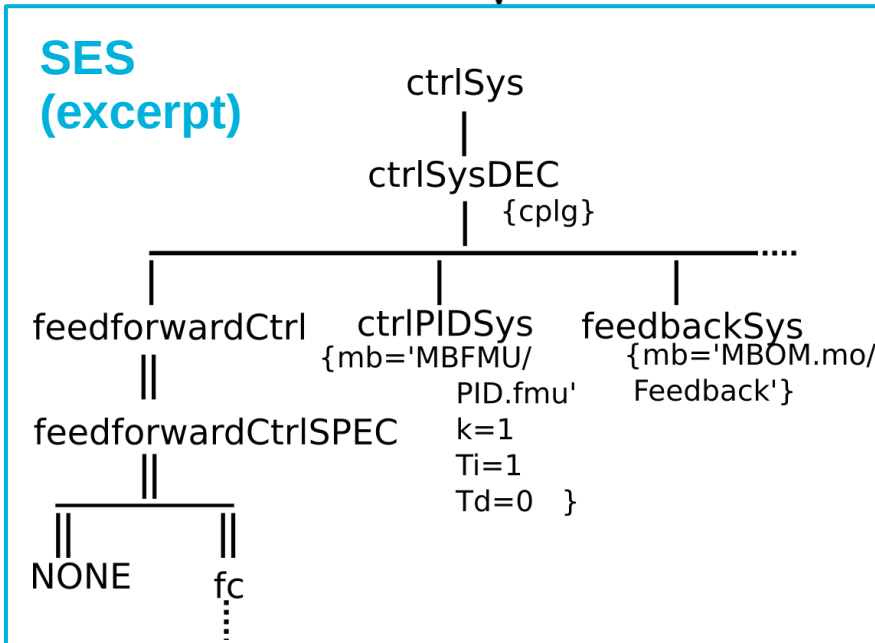
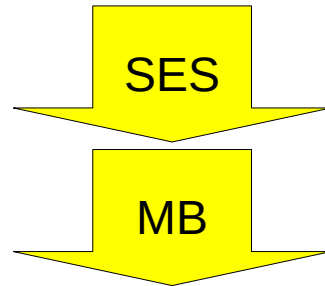
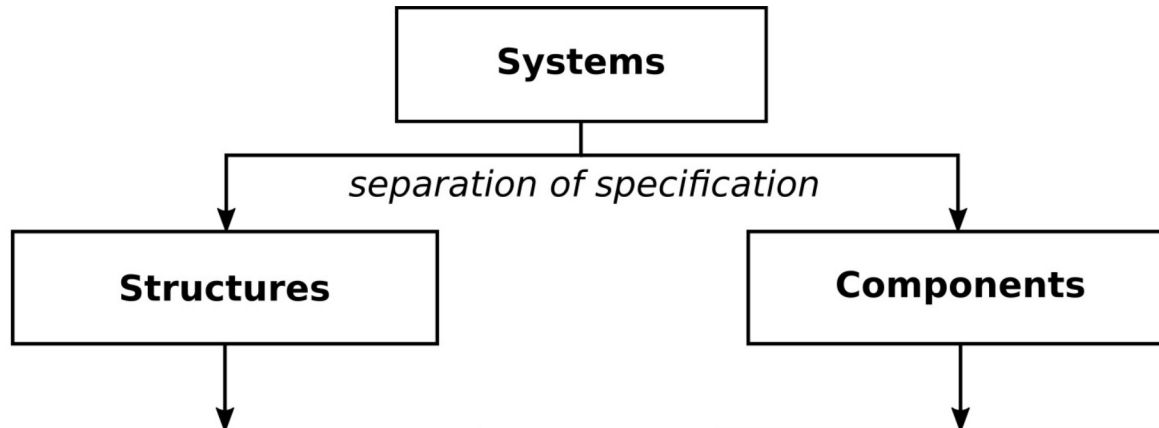
## Case Study with FMI



- 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

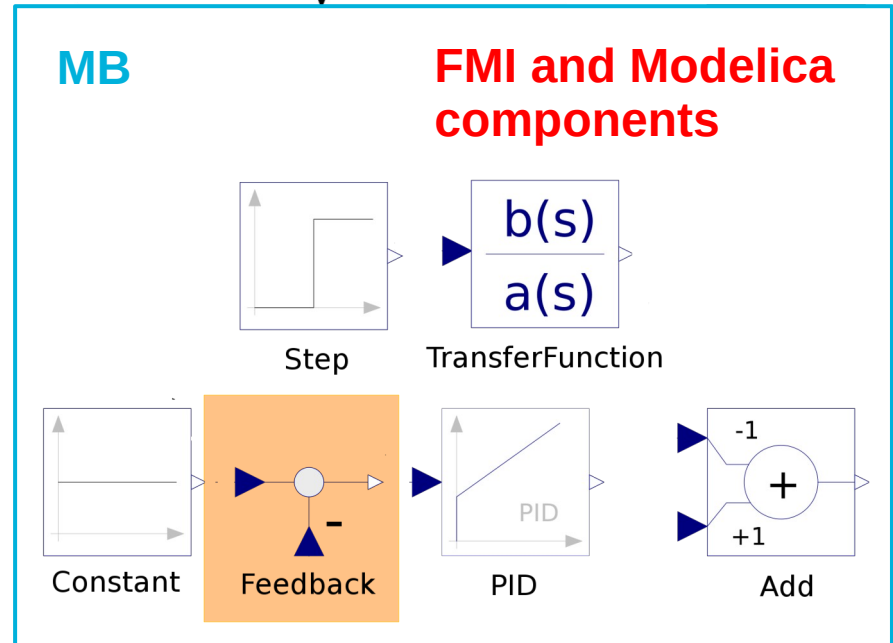
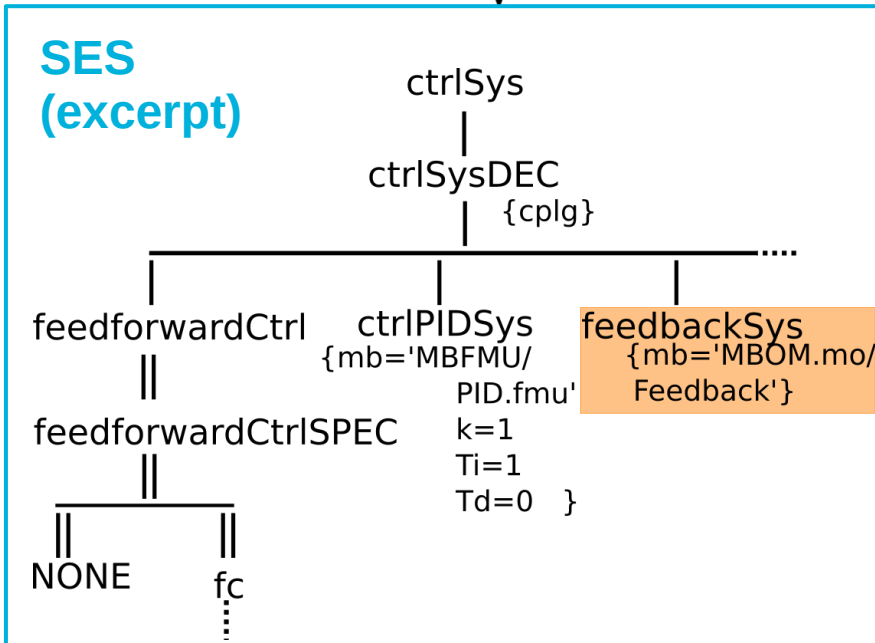
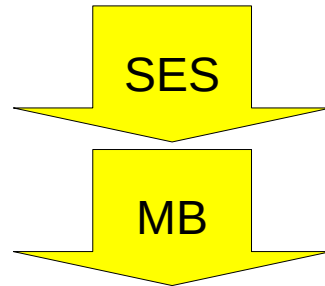
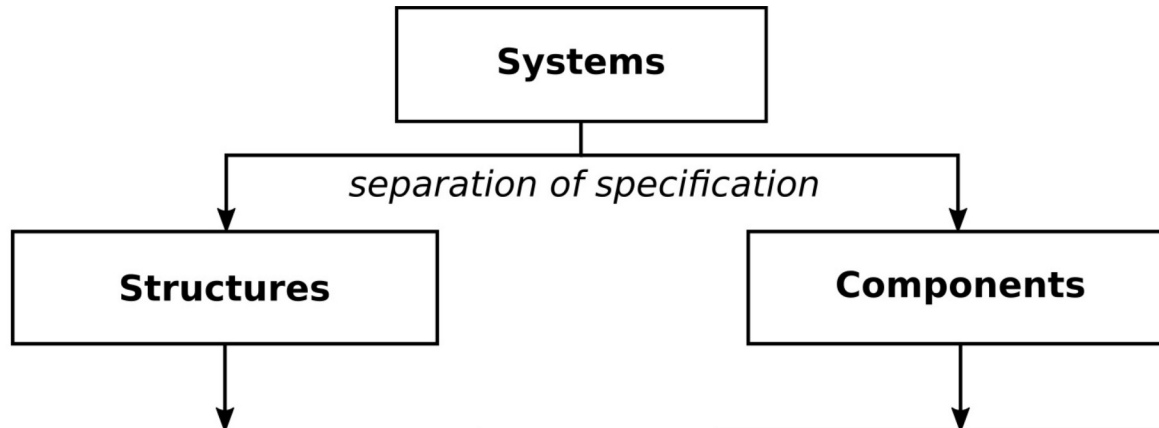


or



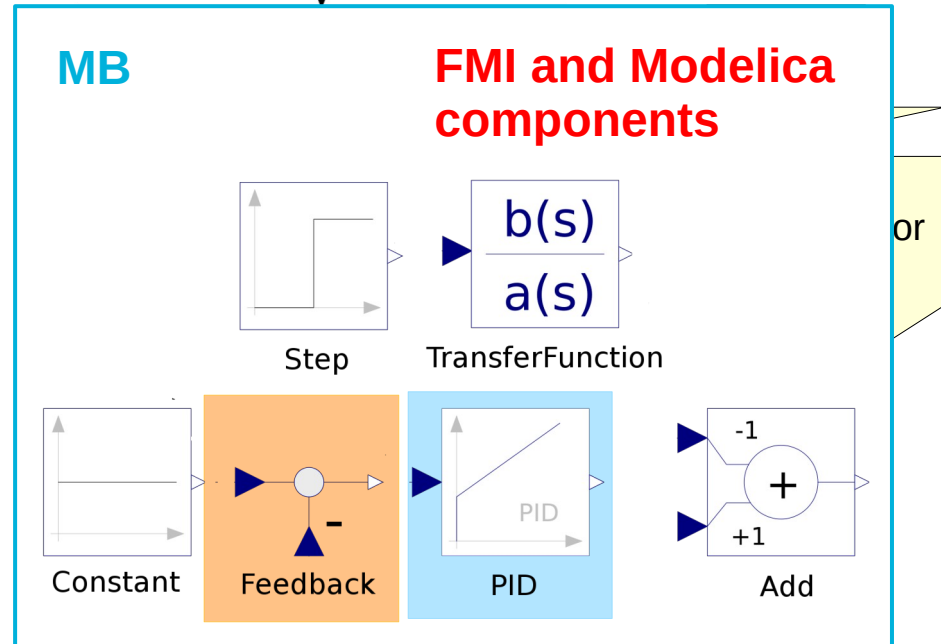
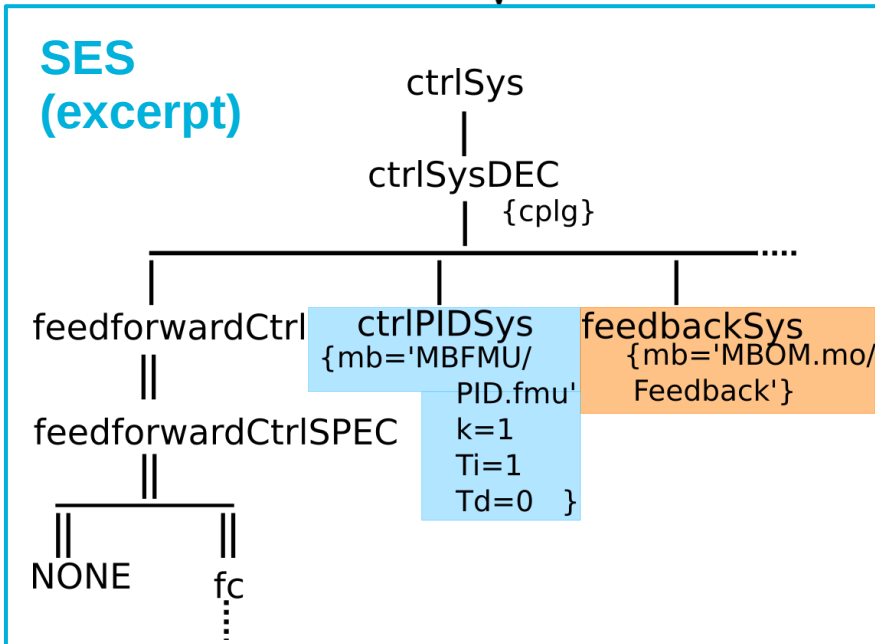
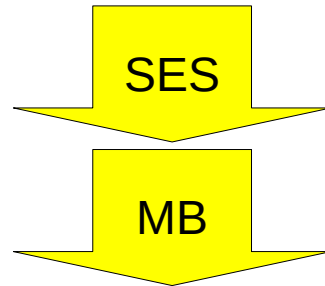
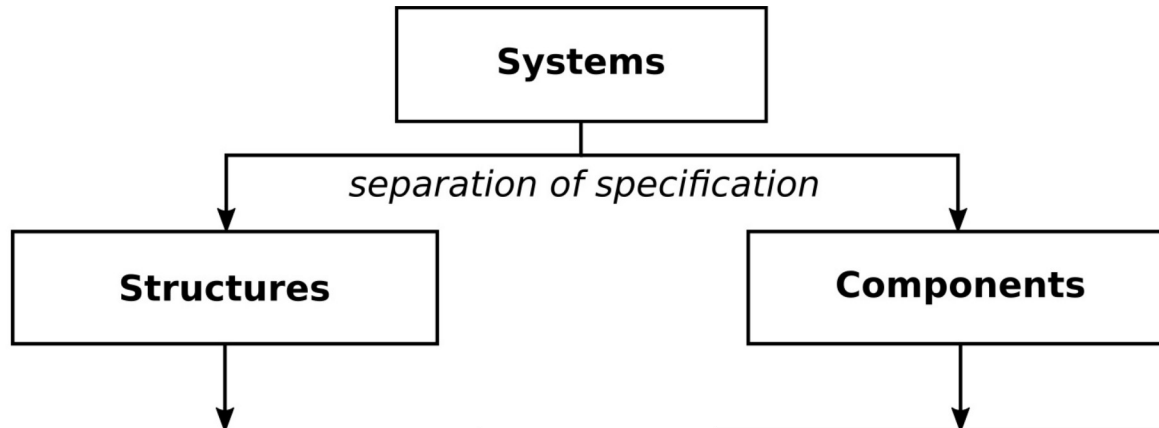


# SES/MB Approach with FMI Components



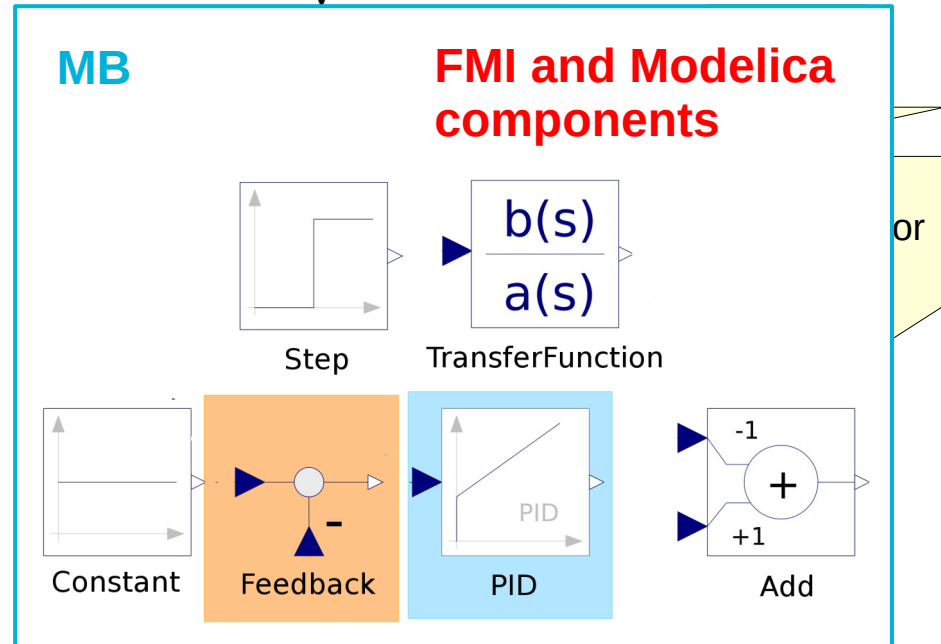
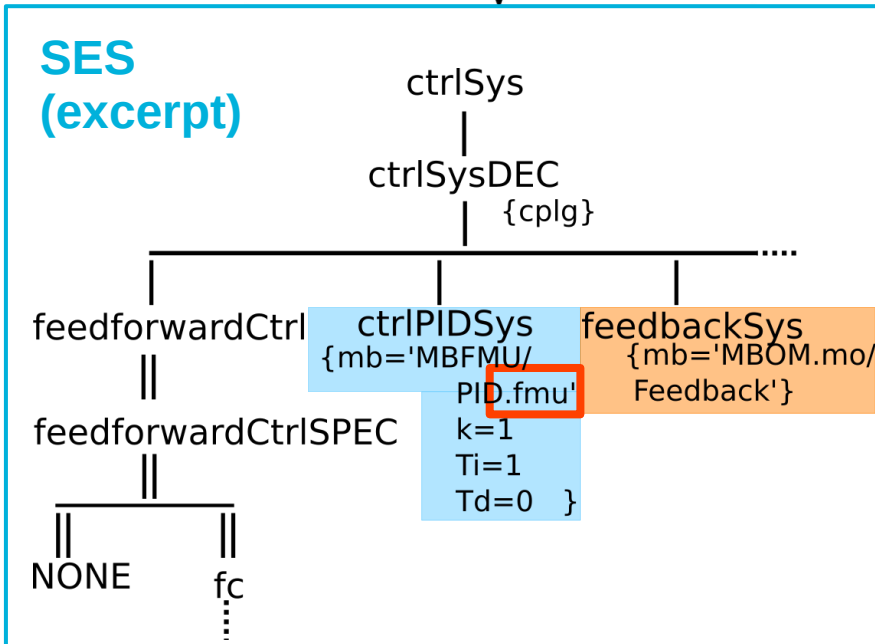
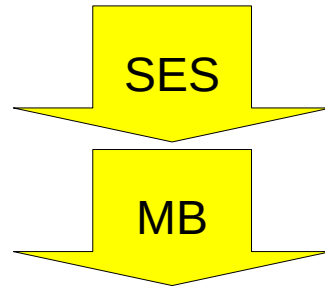
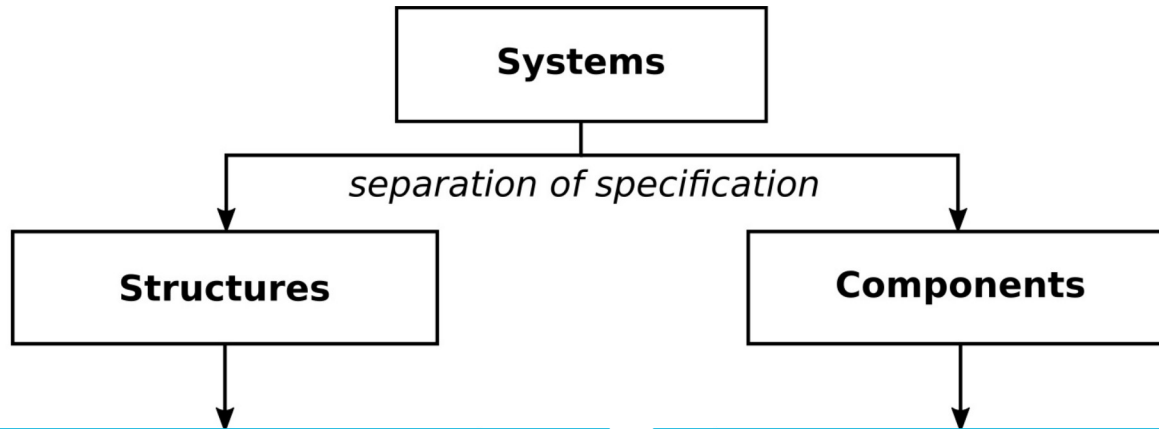


# SES/MB Approach with FMI Components





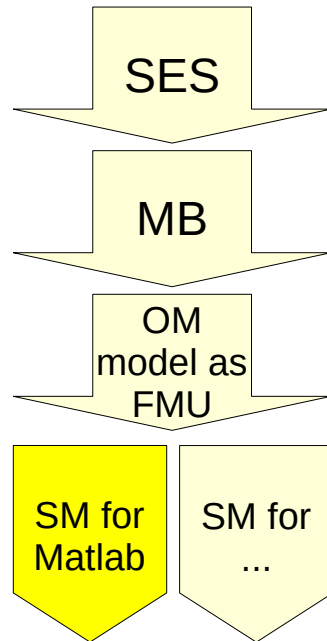
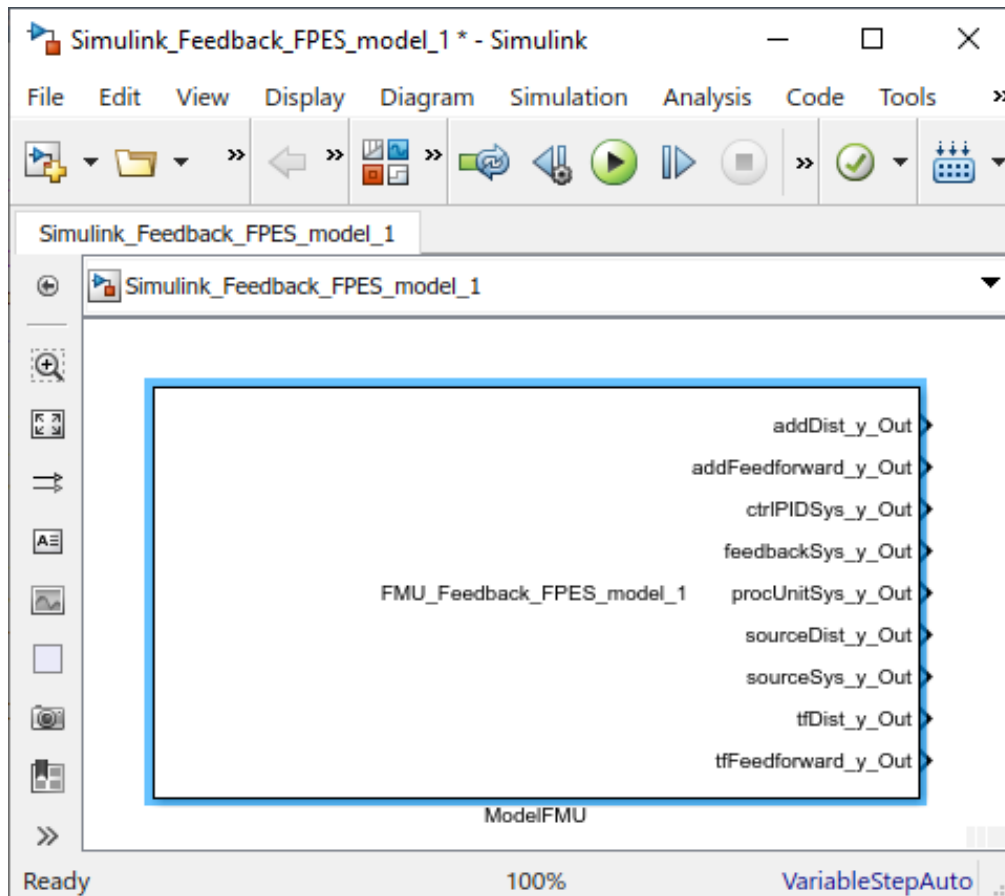
# SES/MB Approach with FMI Components







# Case Study: Model FMU Imported in MATLAB/Simulink



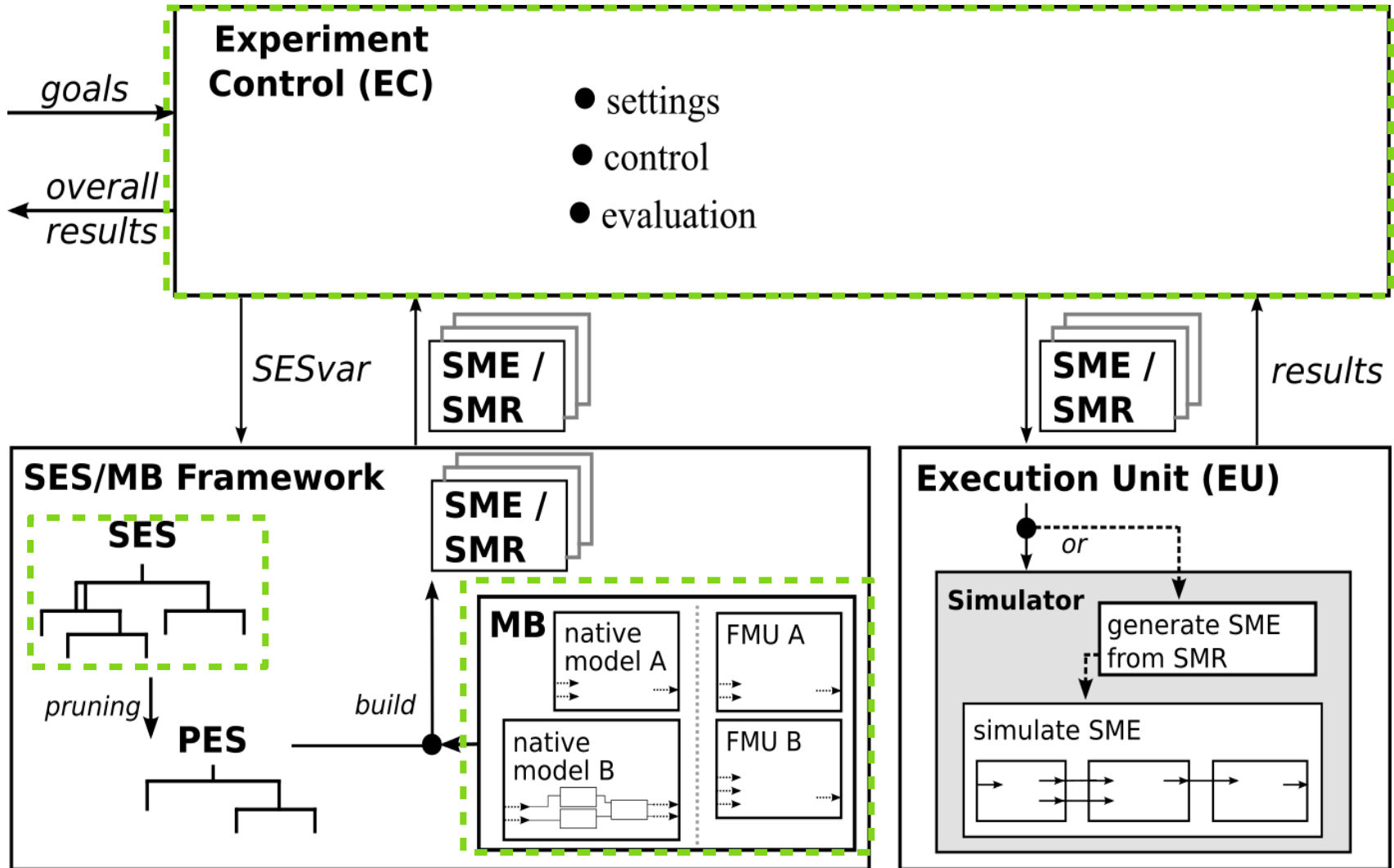


# Outline

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

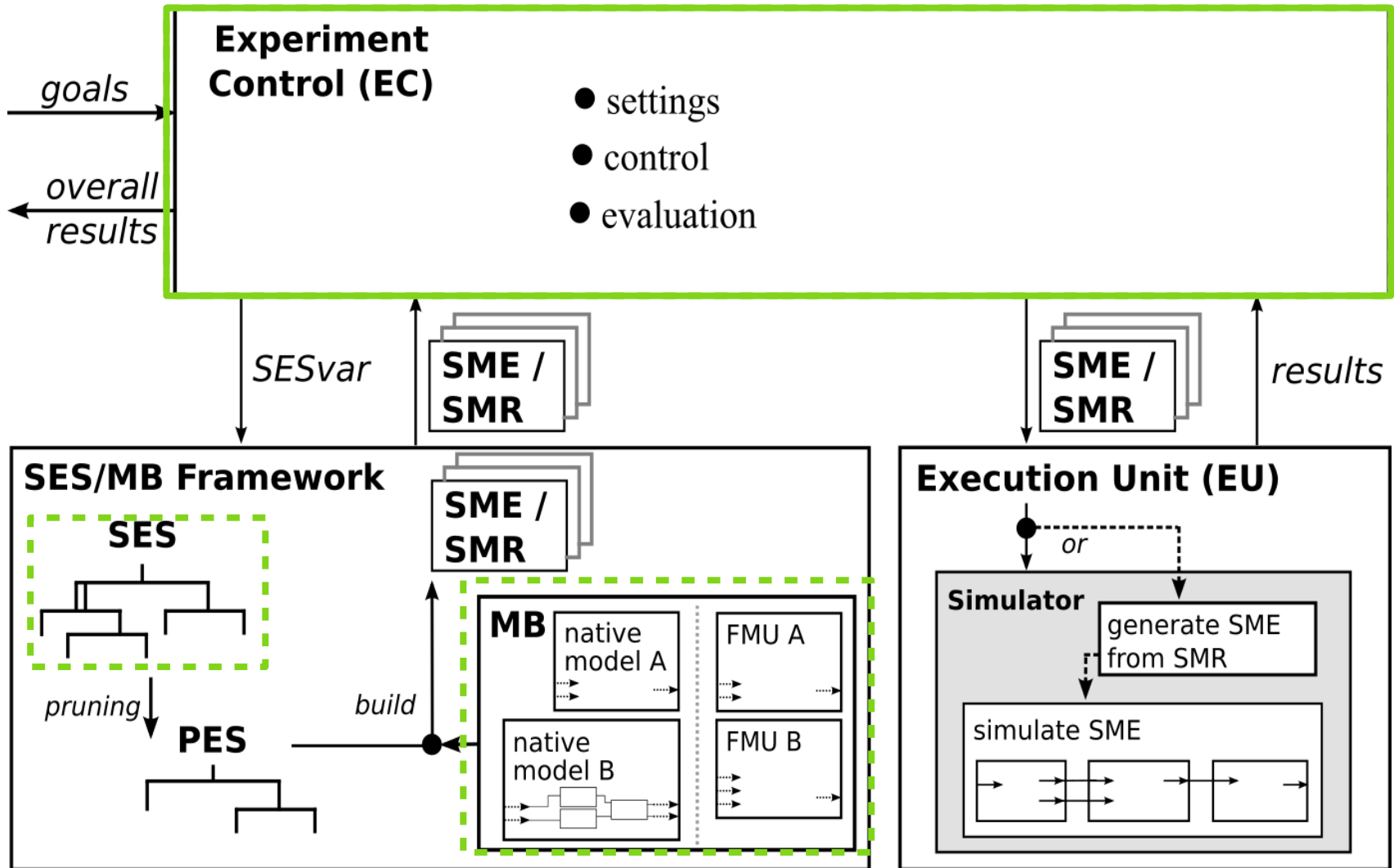


# Extended SES/MB Architecture (Native and FMI)





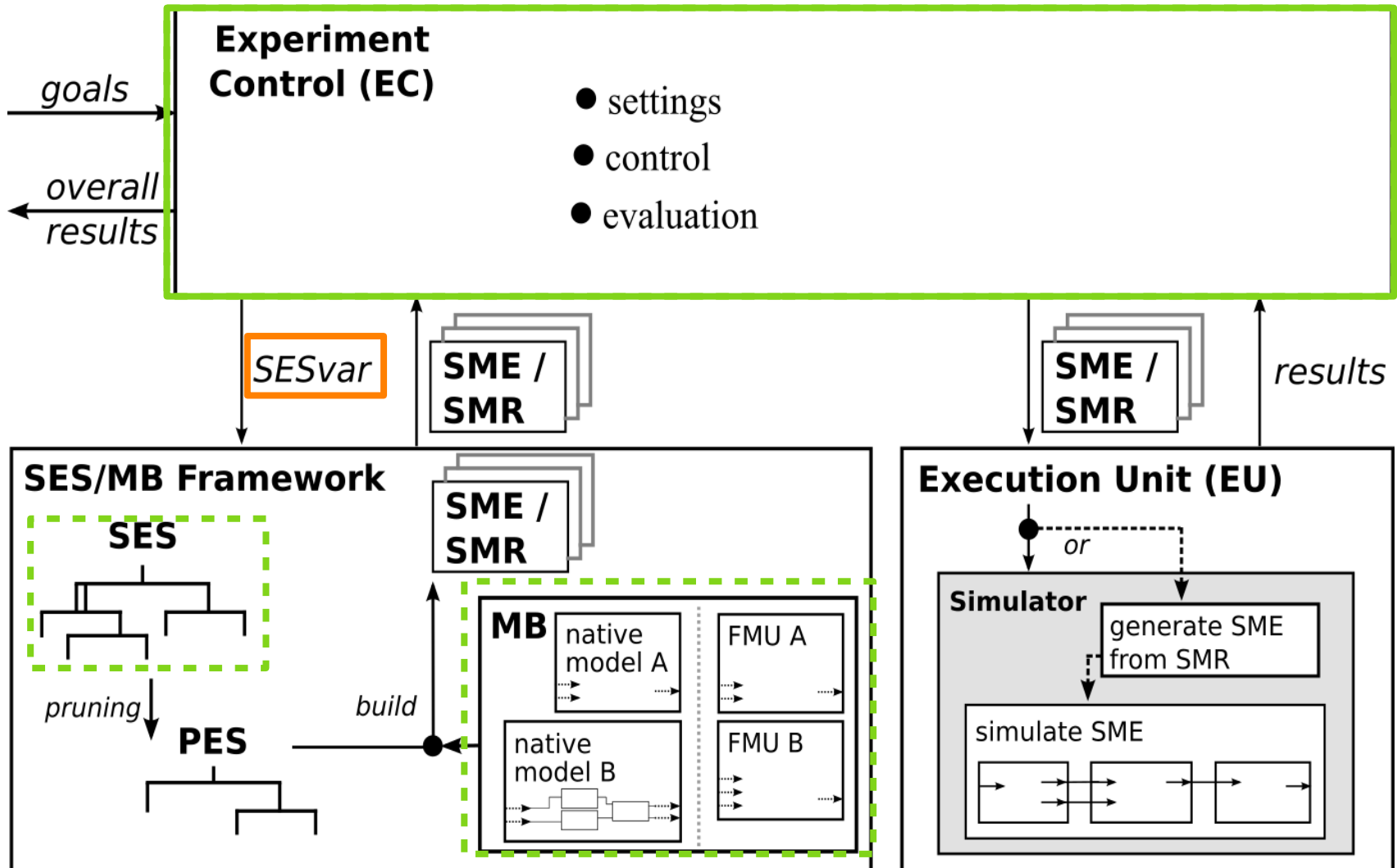
# Extended SES/MB Architecture (Native and FMI)





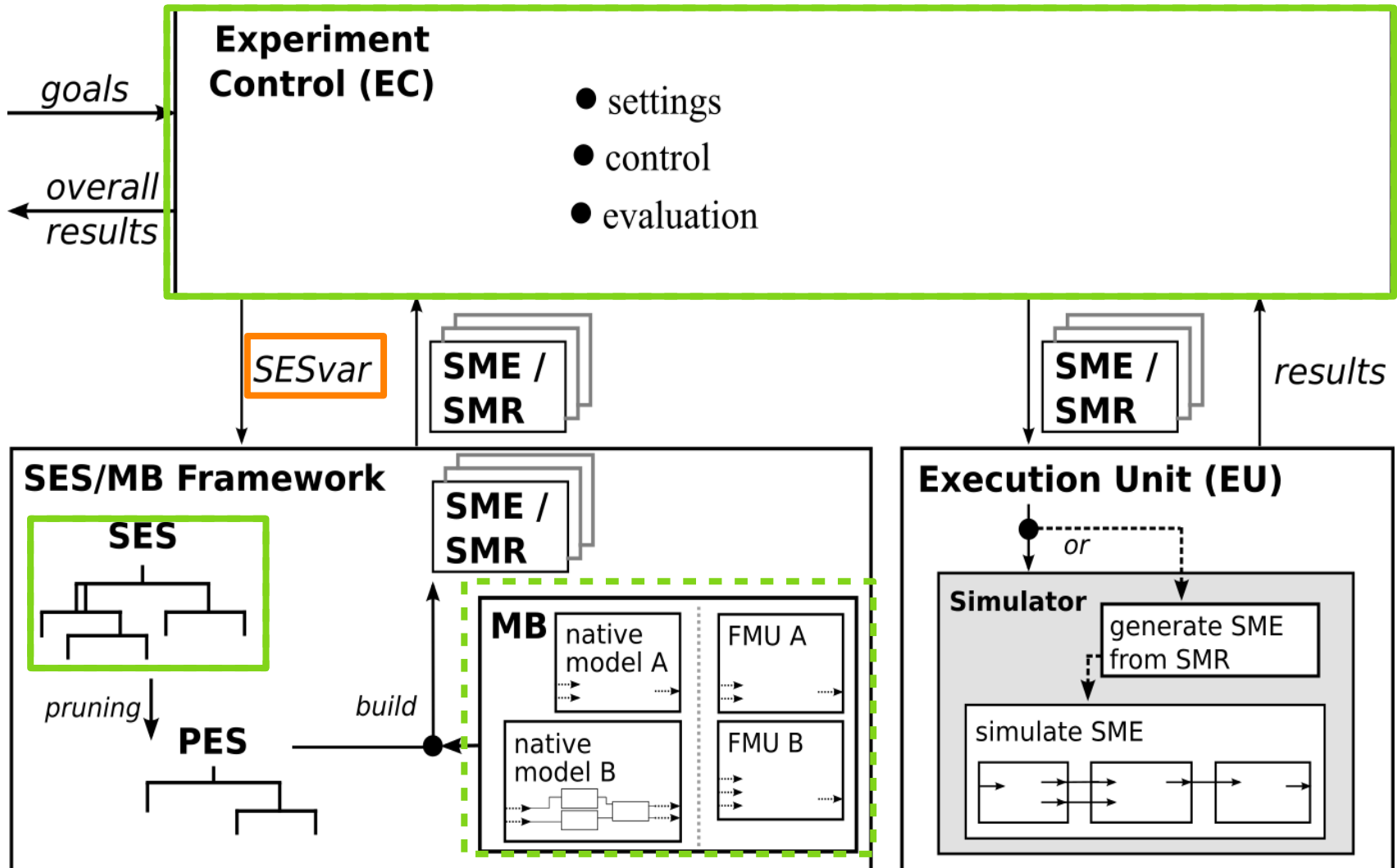


# Extended SES/MB Architecture (Native and FMI)



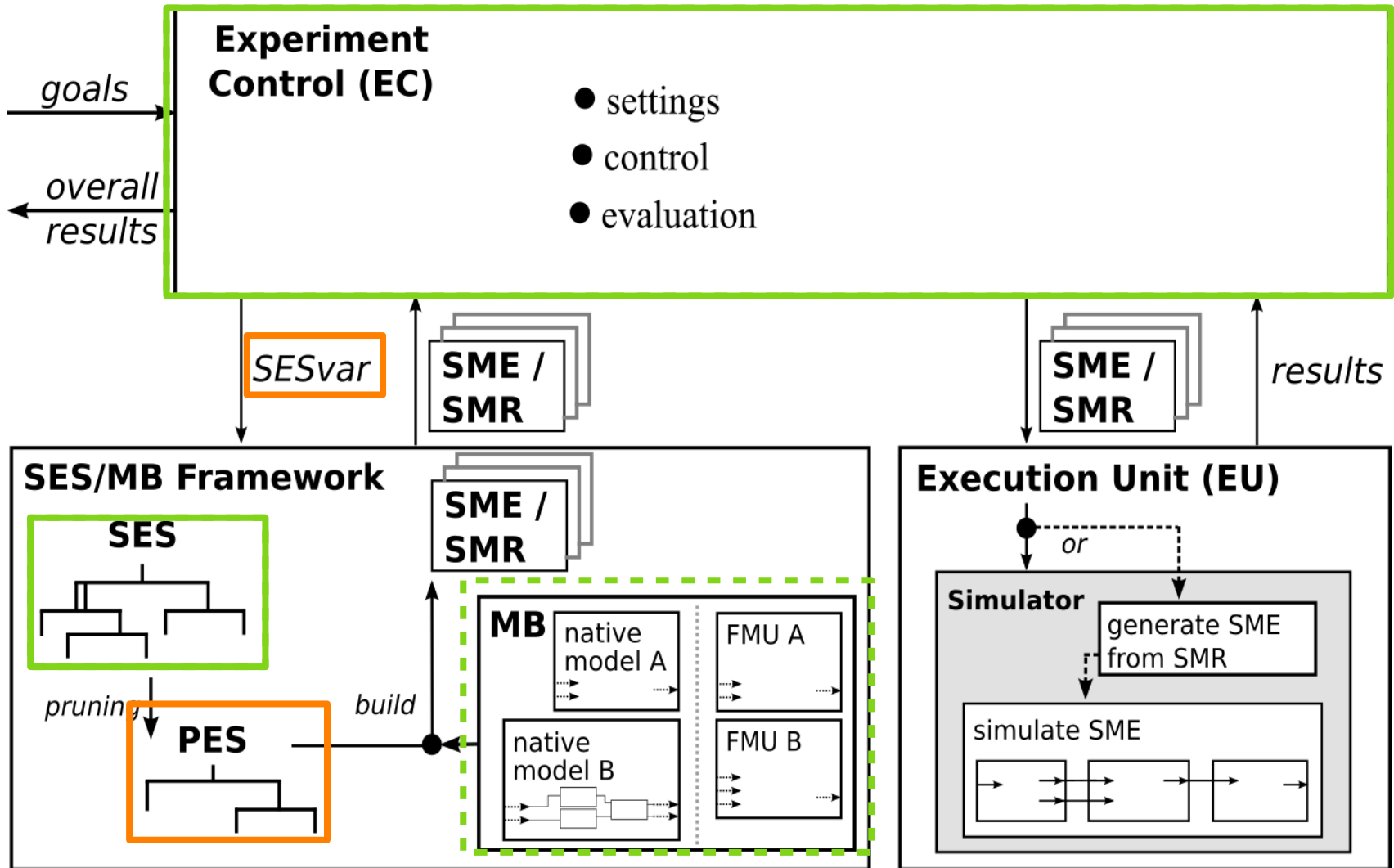


# Extended SES/MB Architecture (Native and FMI)



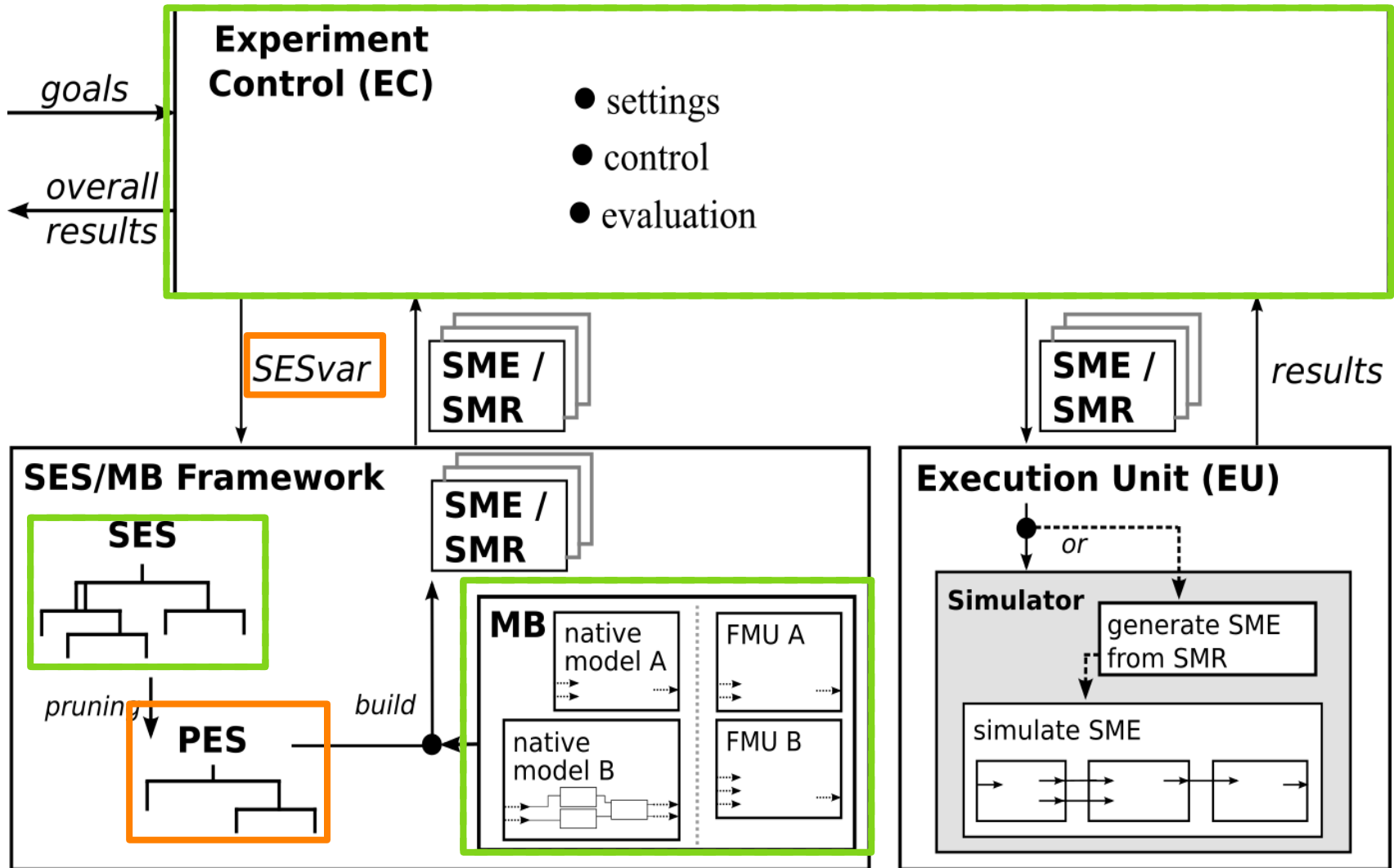


# Extended SES/MB Architecture (Native and FMI)



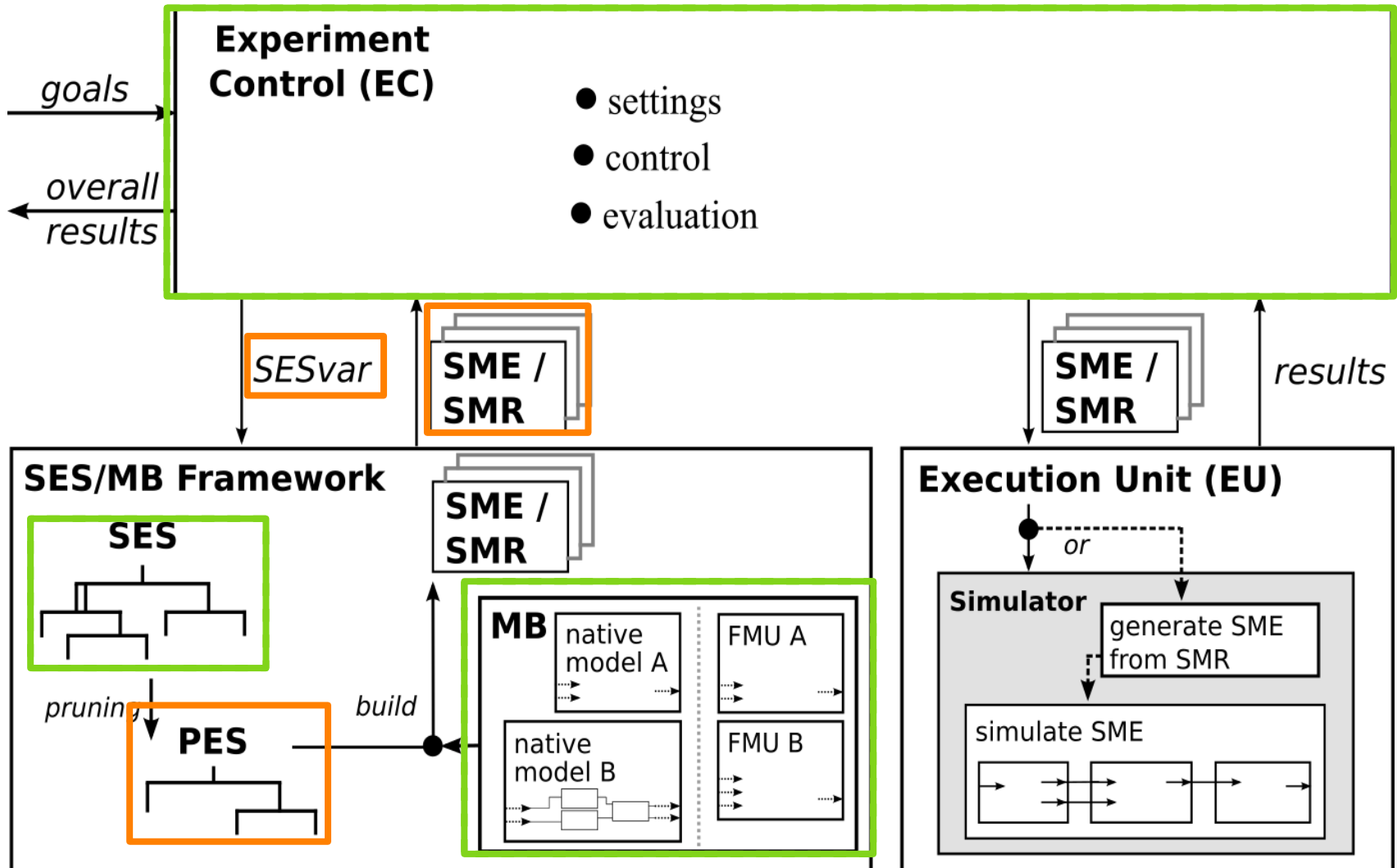


# Extended SES/MB Architecture (Native and FMI)



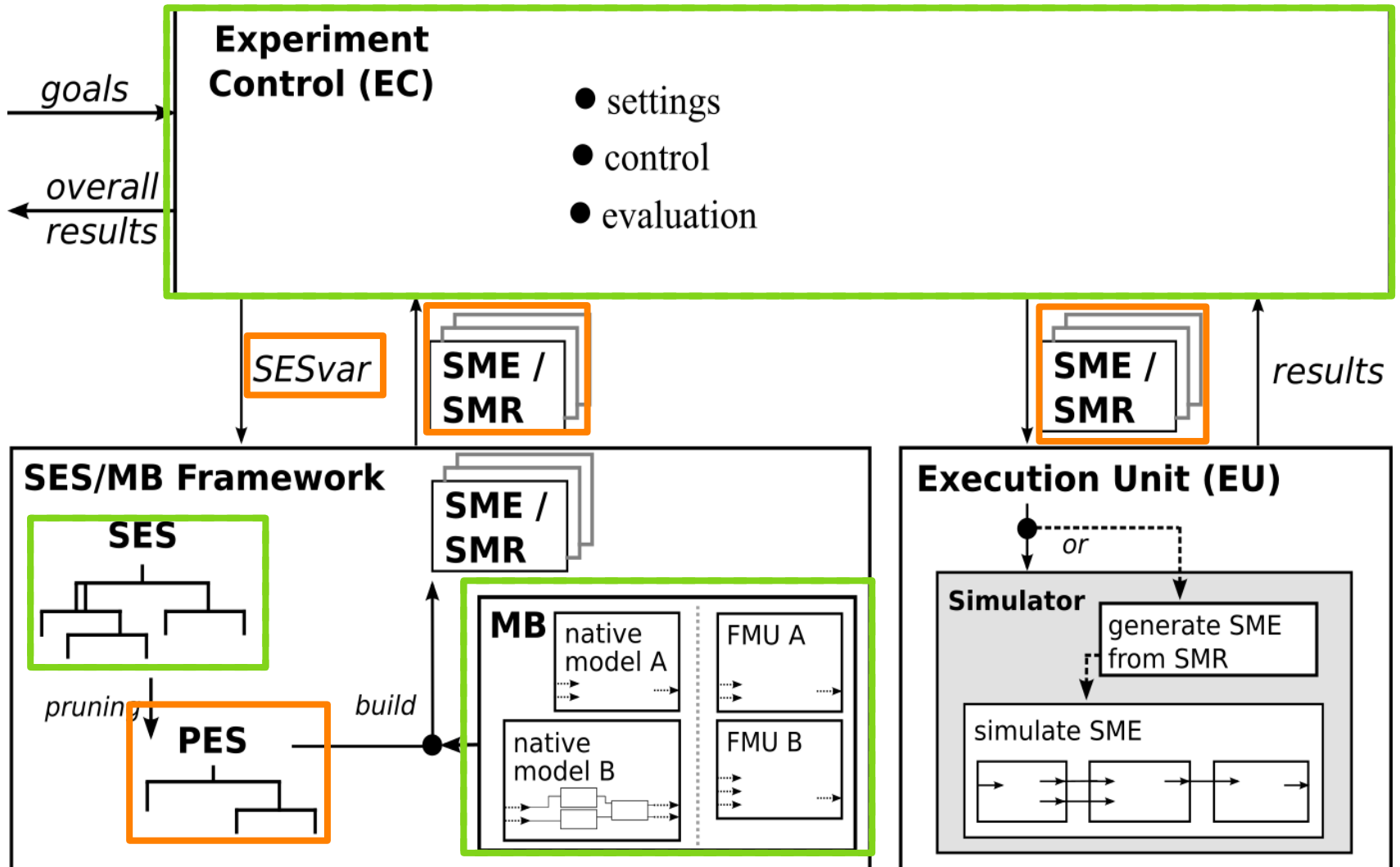


# Extended SES/MB Architecture (Native and FMI)



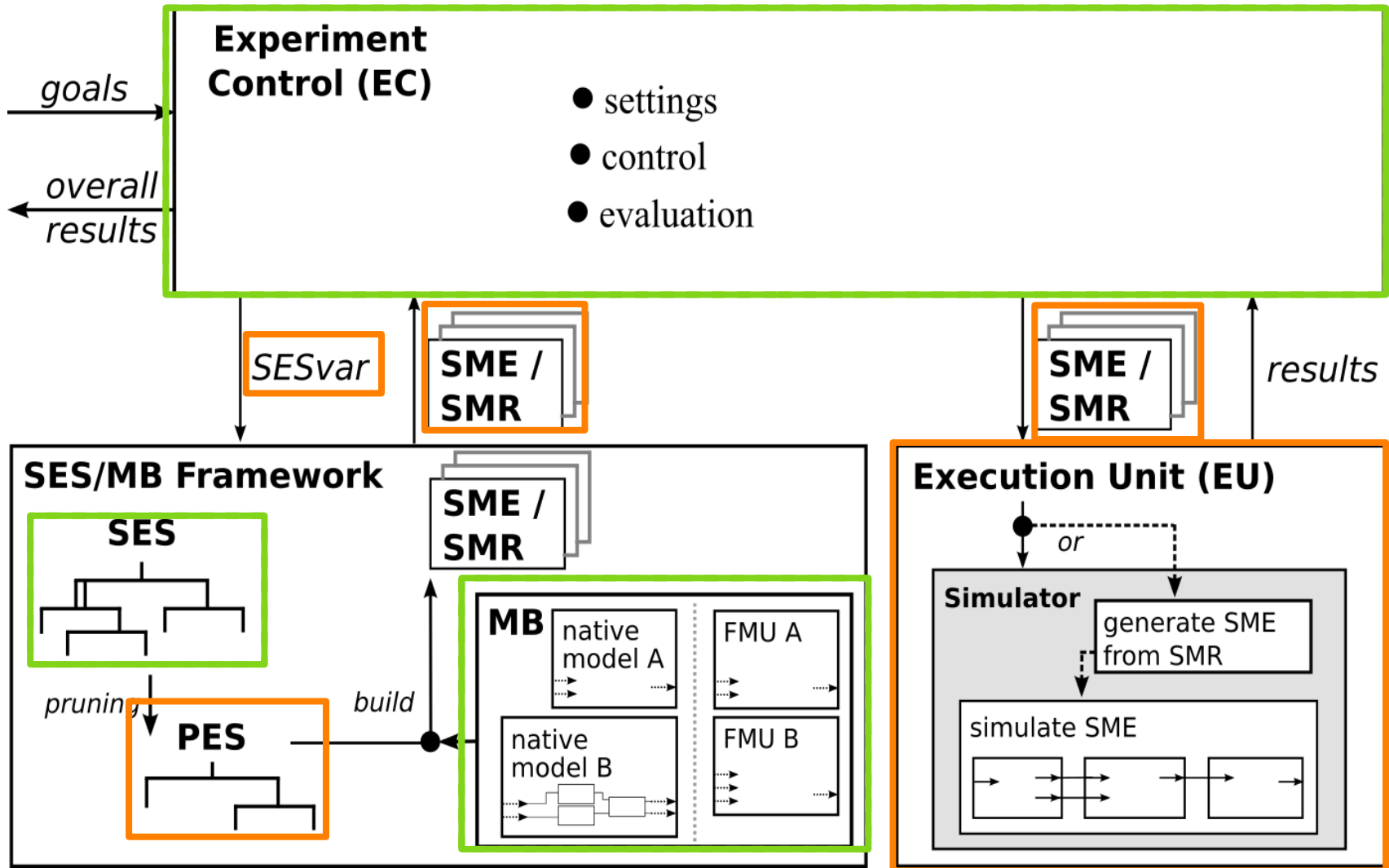


# Extended SES/MB Architecture (Native and FMI)



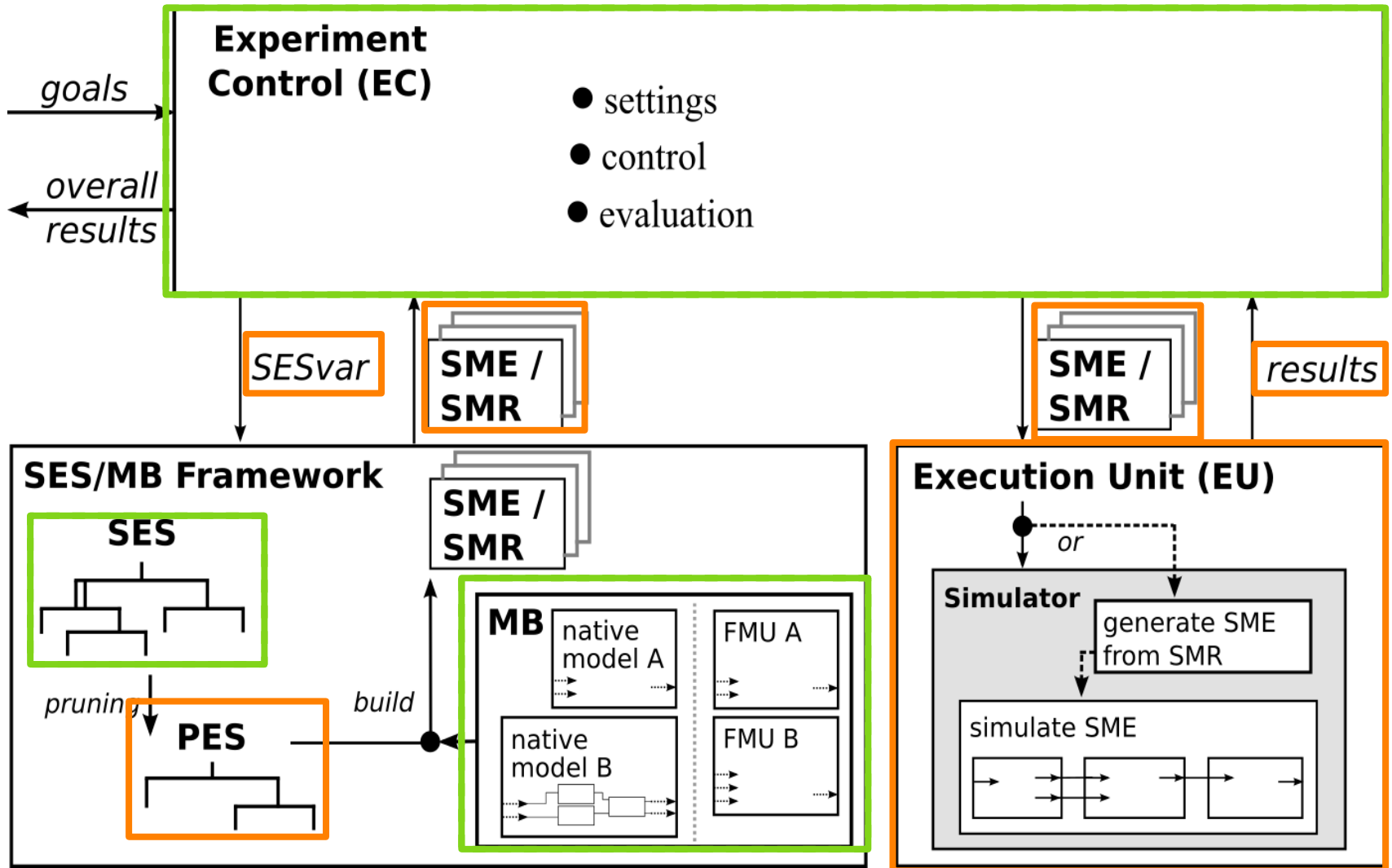


# Extended SES/MB Architecture (Native and FMI)





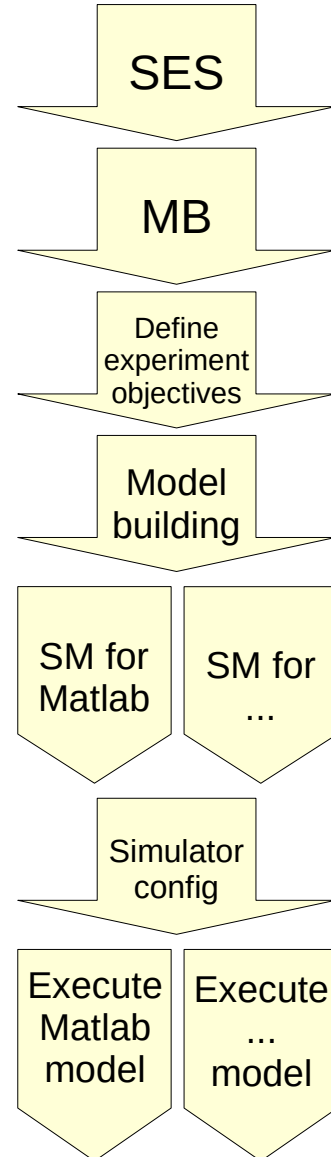
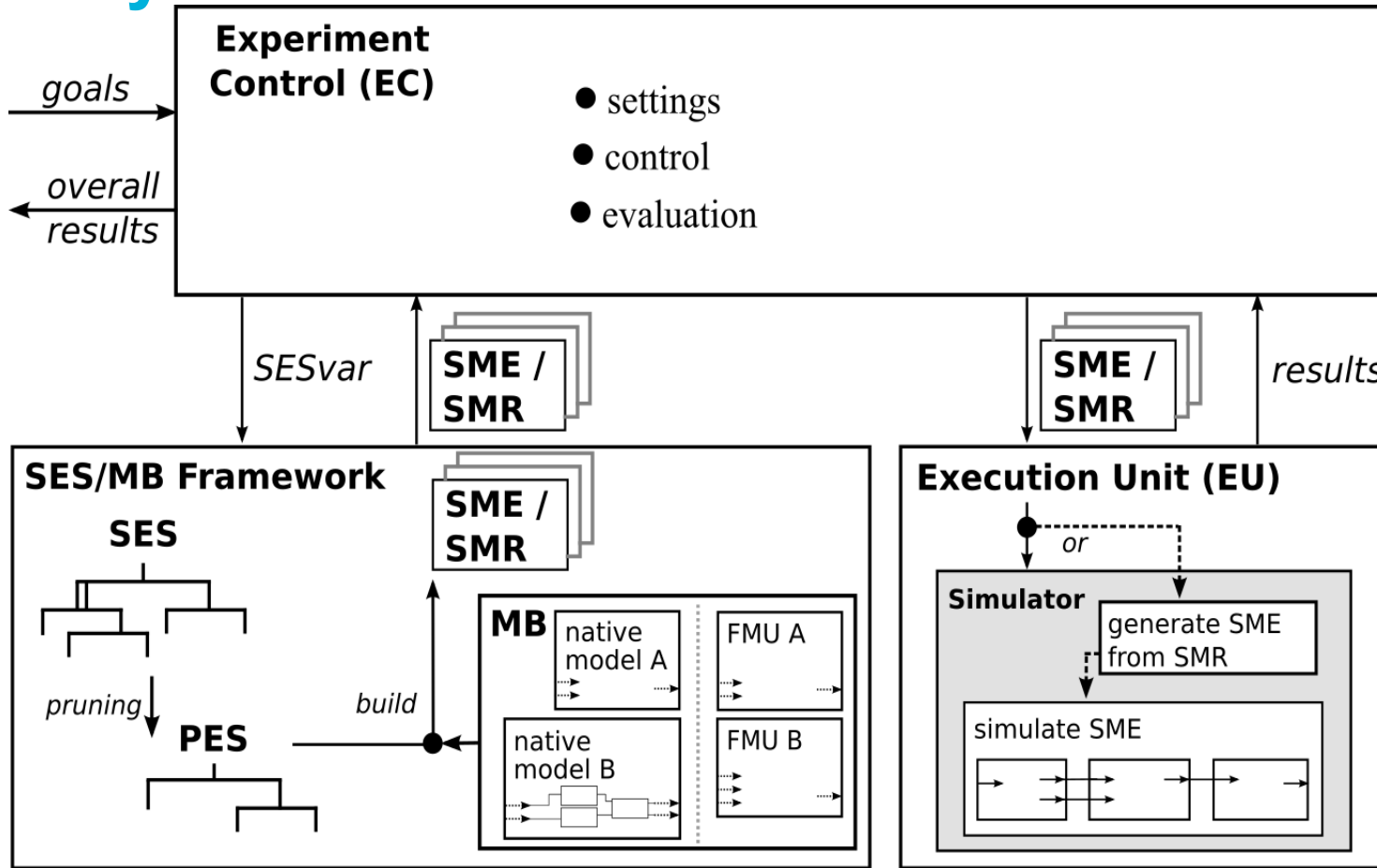
# Extended SES/MB Architecture (Native and FMI)





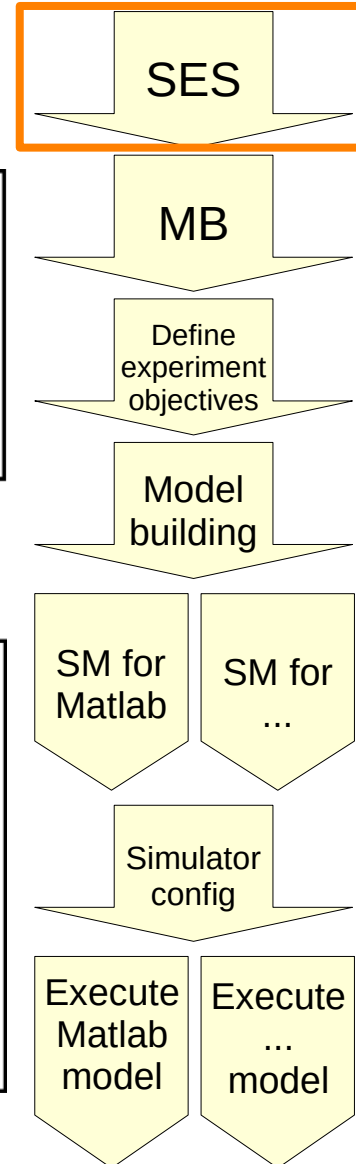
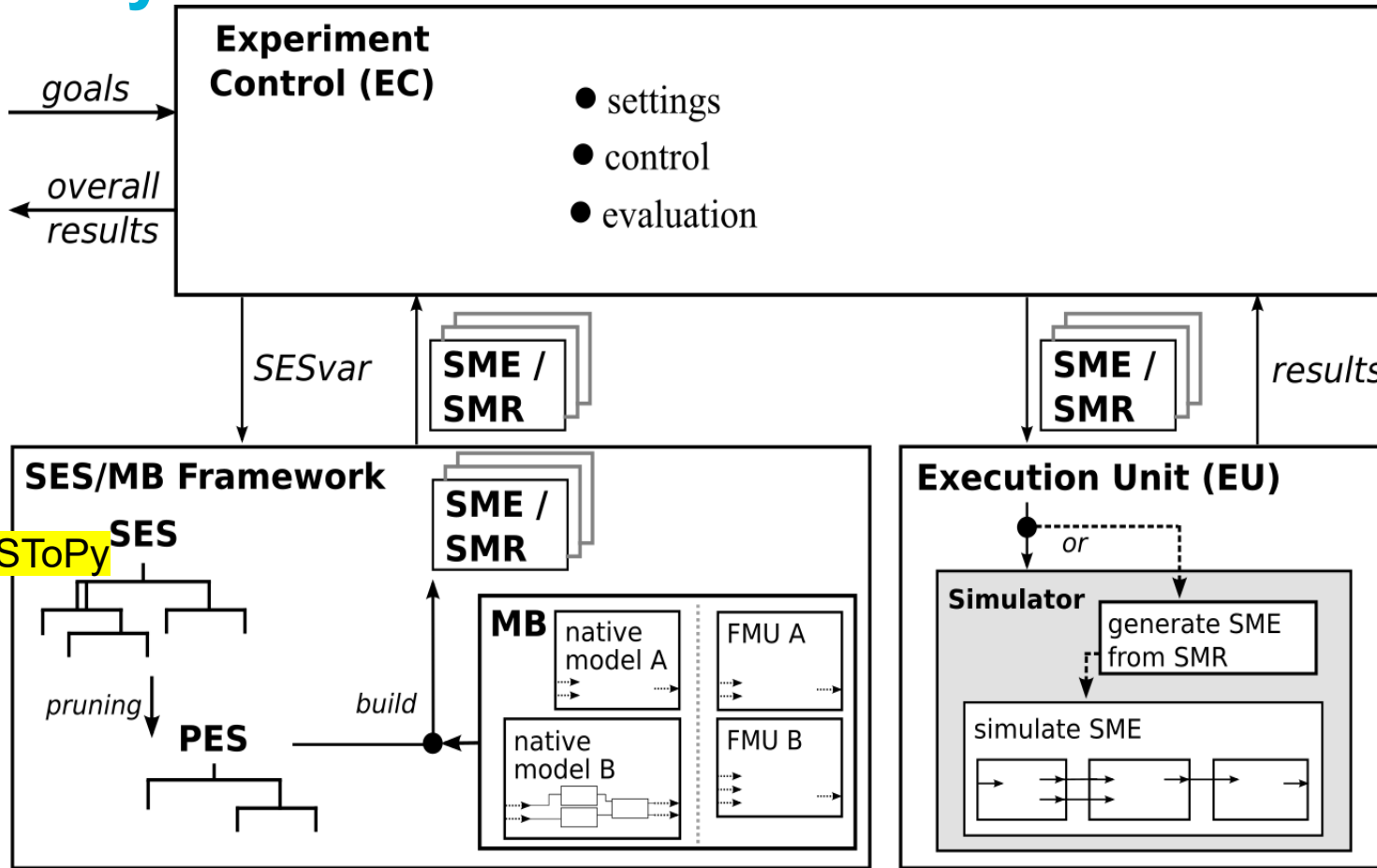


# Extended SES/MB Architecture Python Toolset





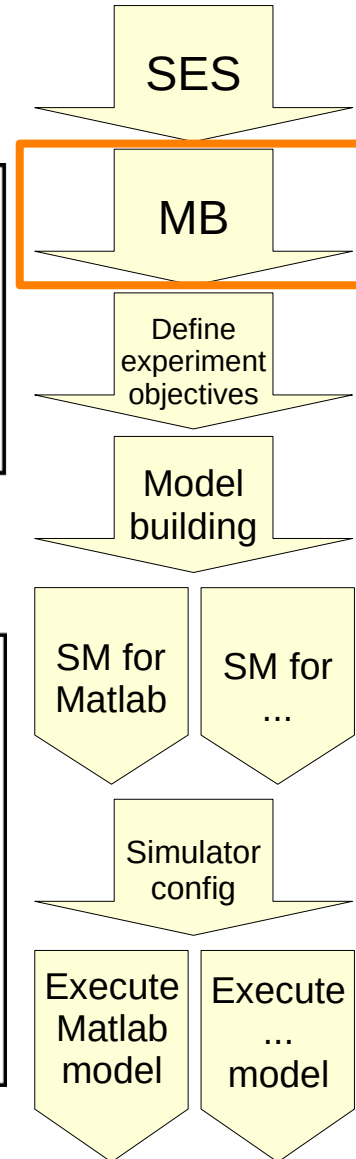
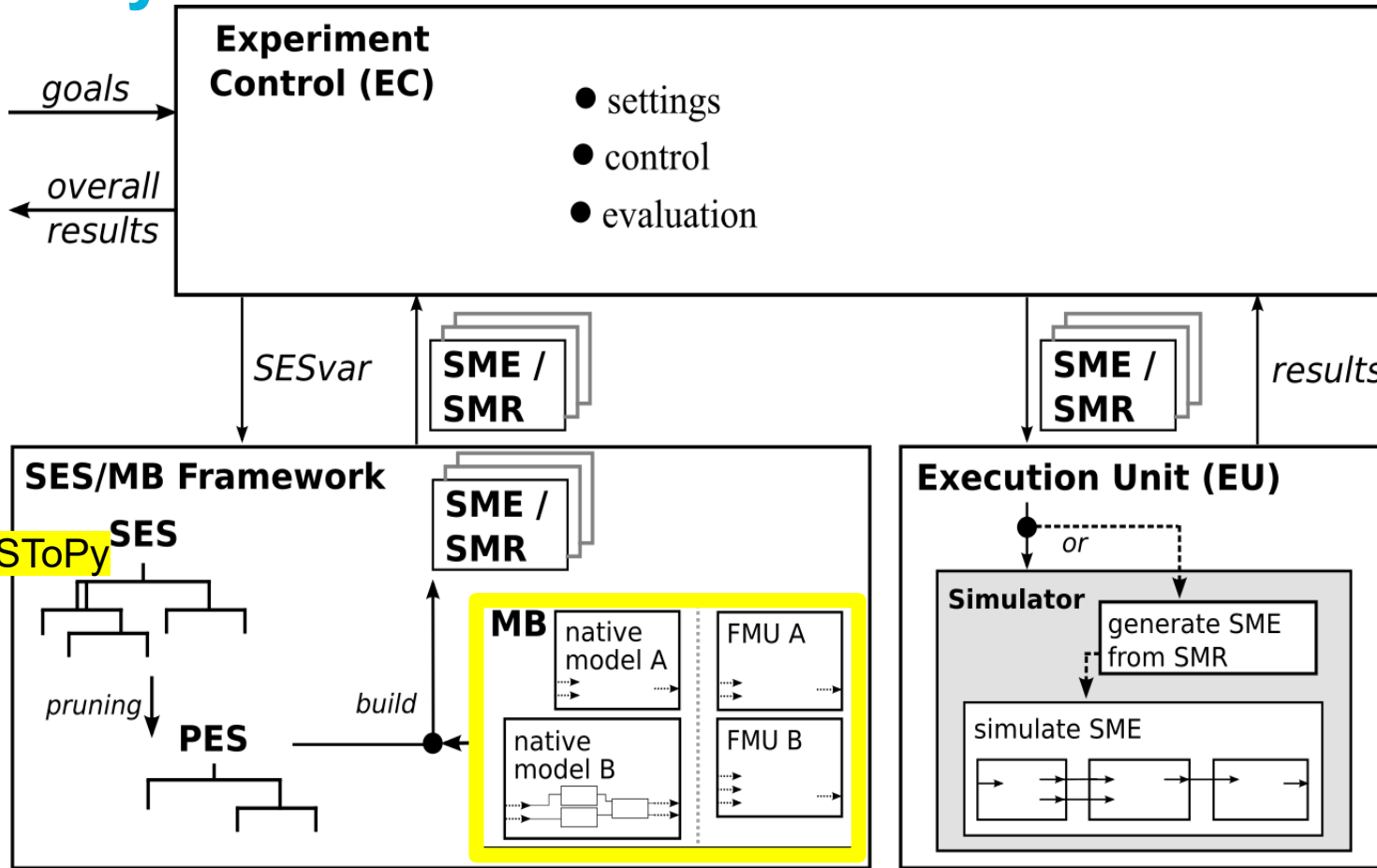
# Extended SES/MB Architecture Python Toolset



SESToPy



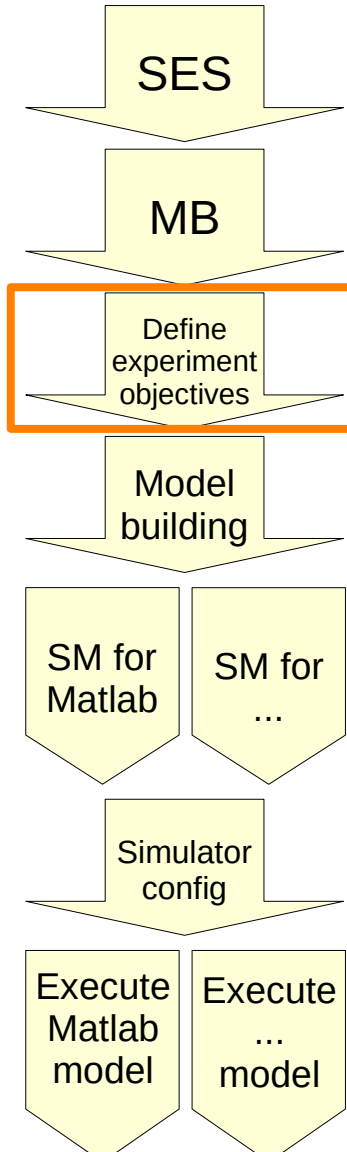
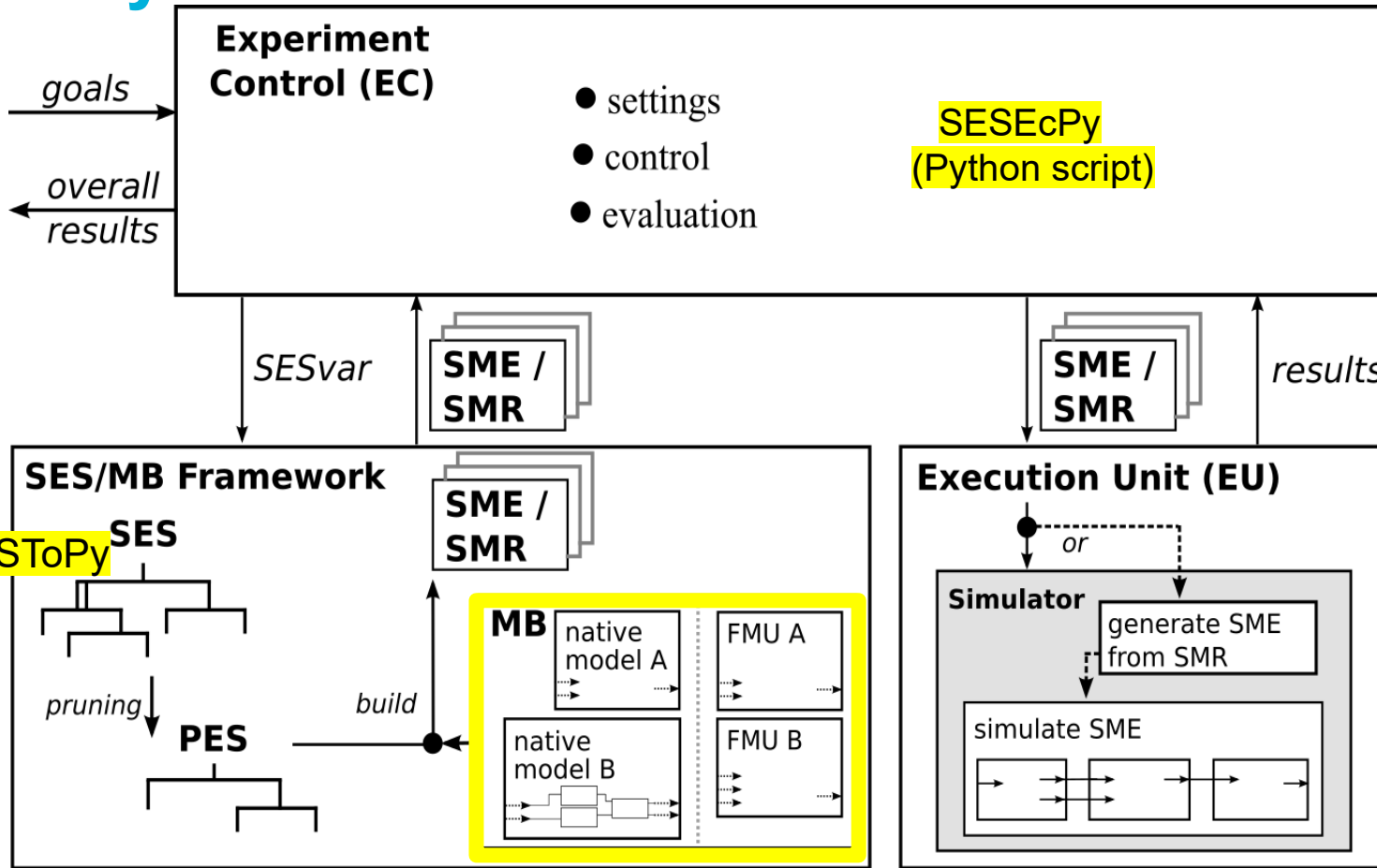
# Extended SES/MB Architecture Python Toolset



SESToPy

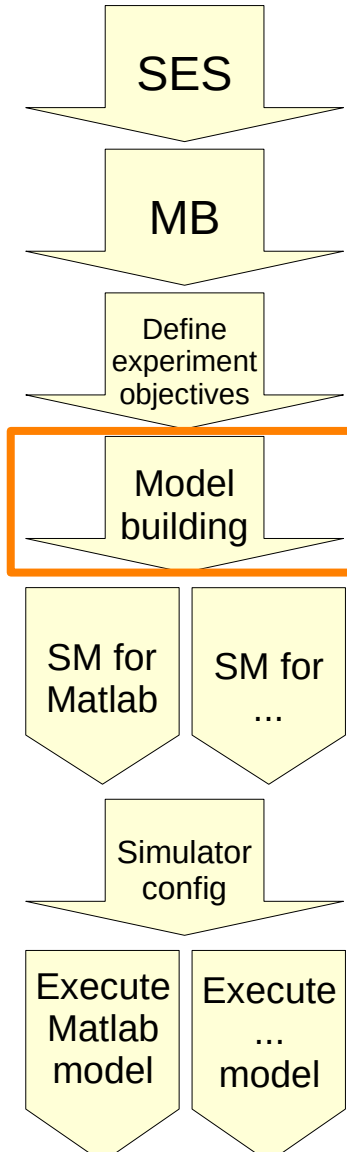
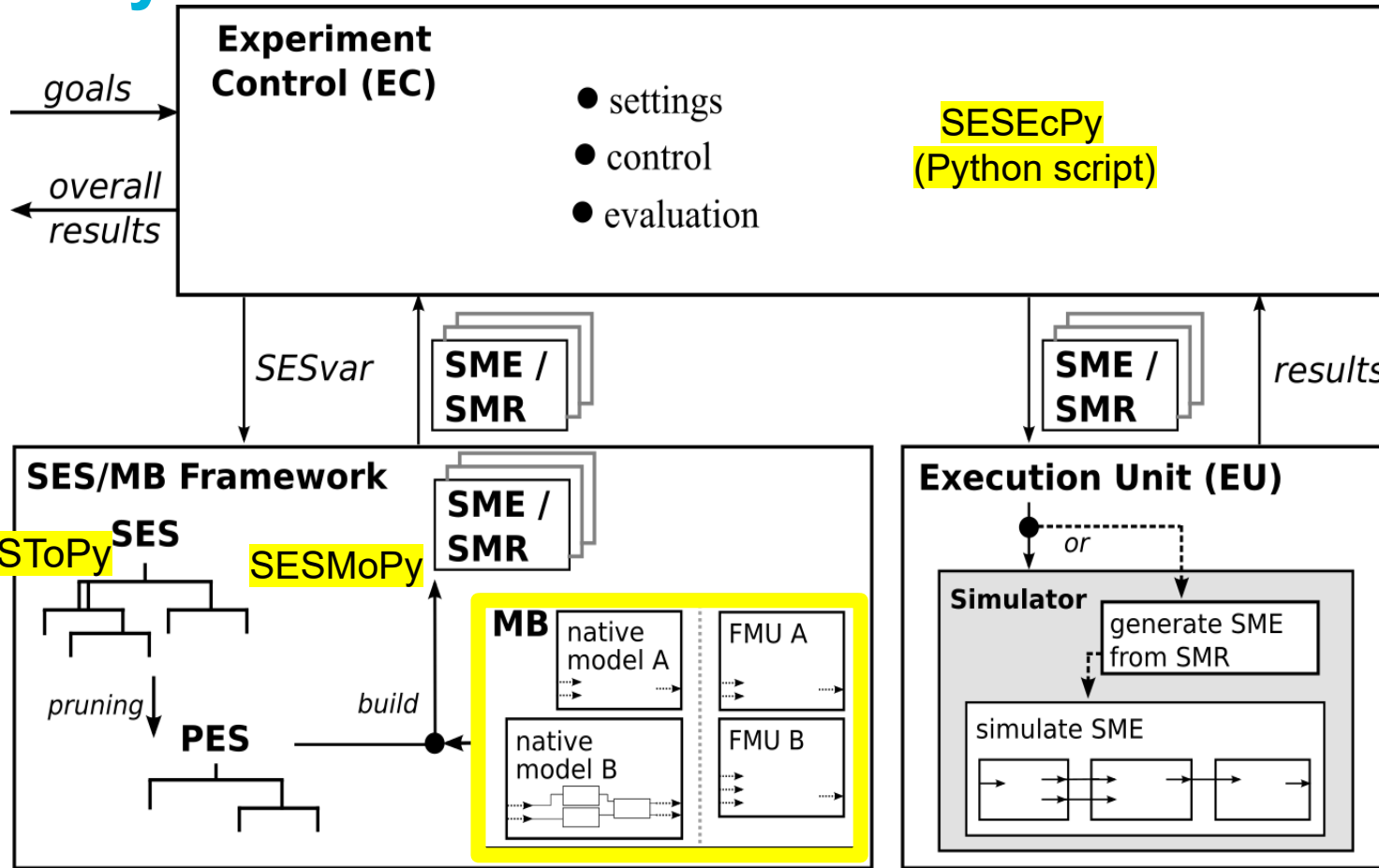


# Extended SES/MB Architecture Python Toolset



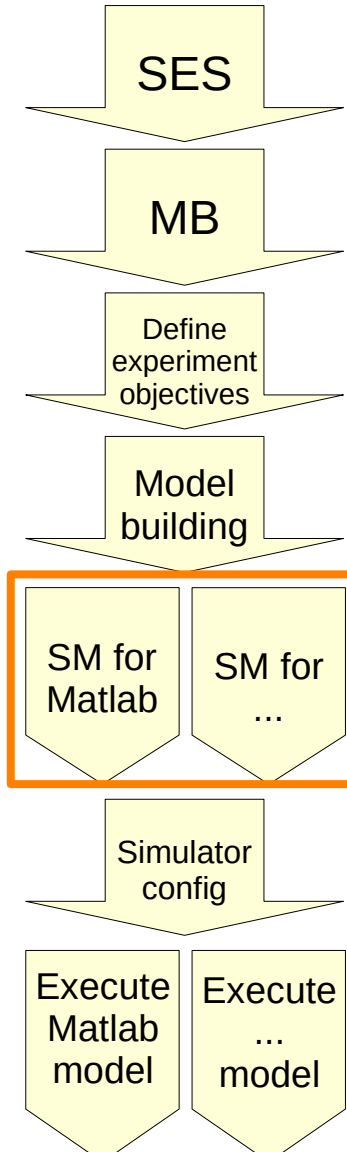
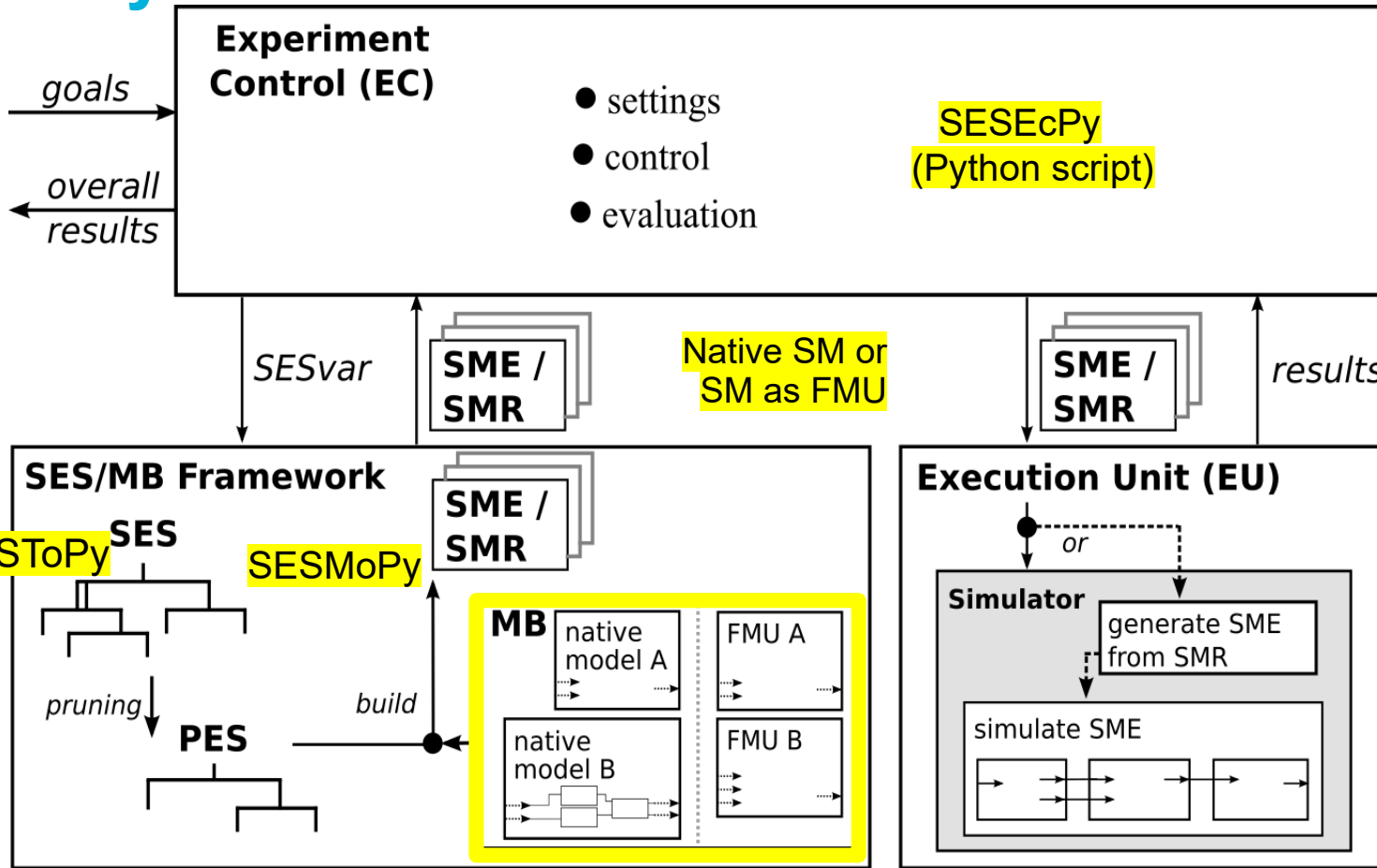


# Extended SES/MB Architecture Python Toolset



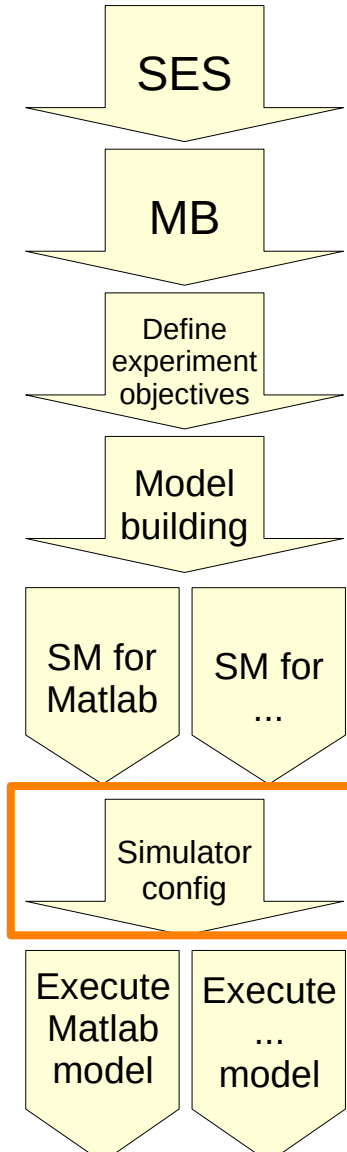
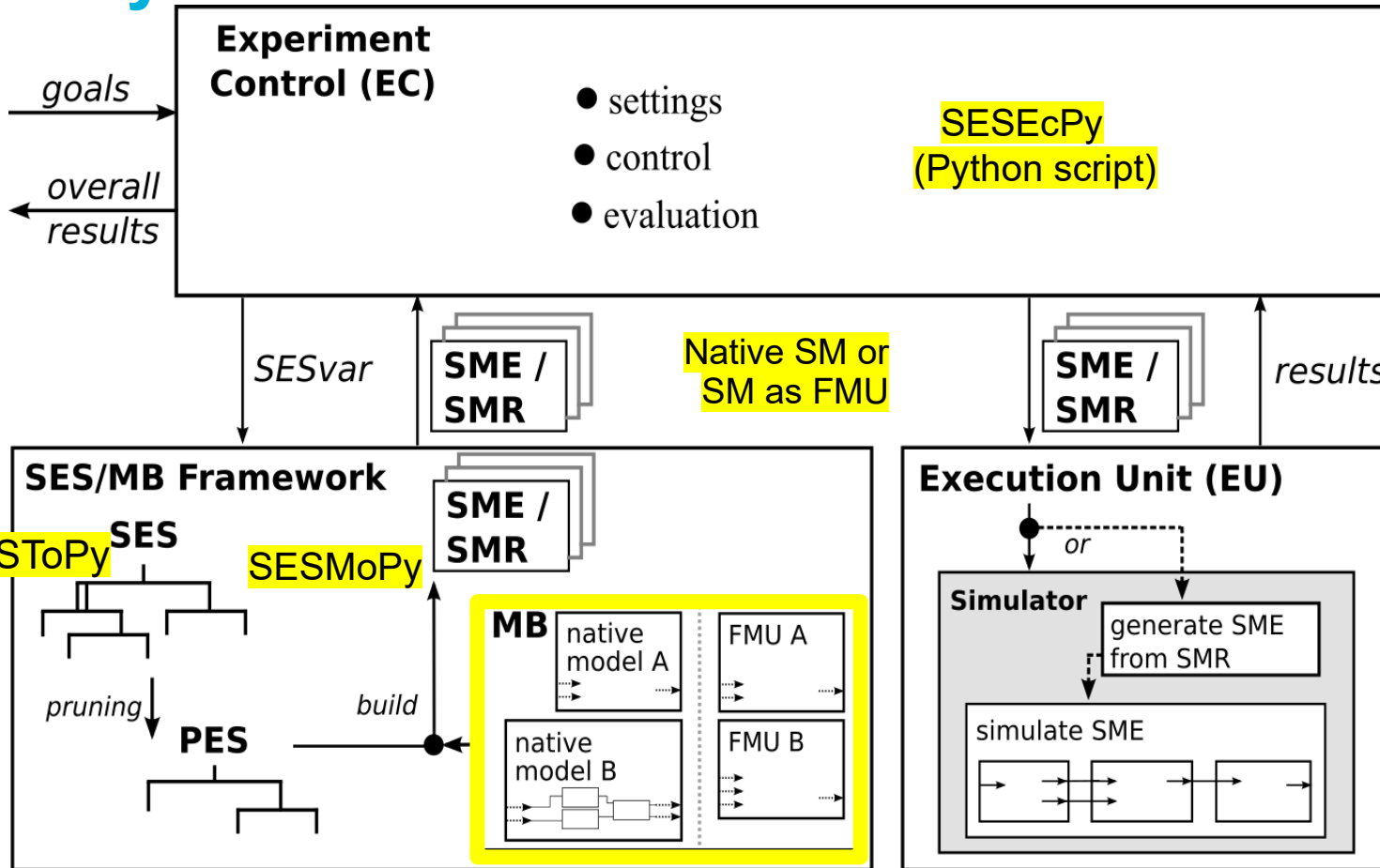


# Extended SES/MB Architecture Python Toolset



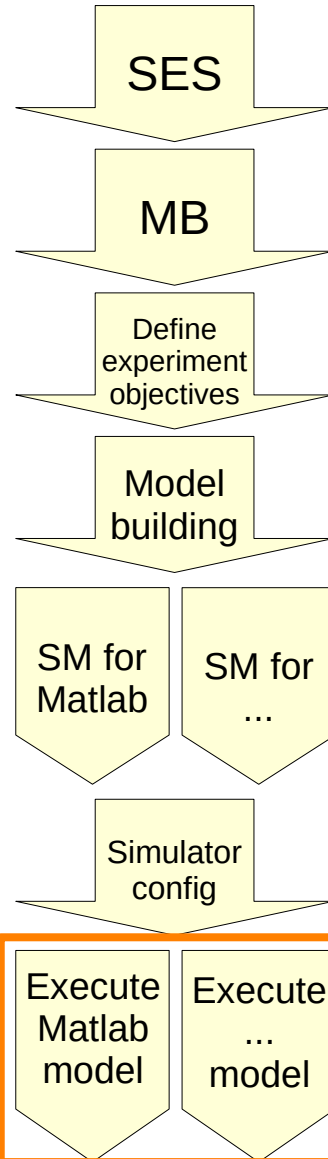
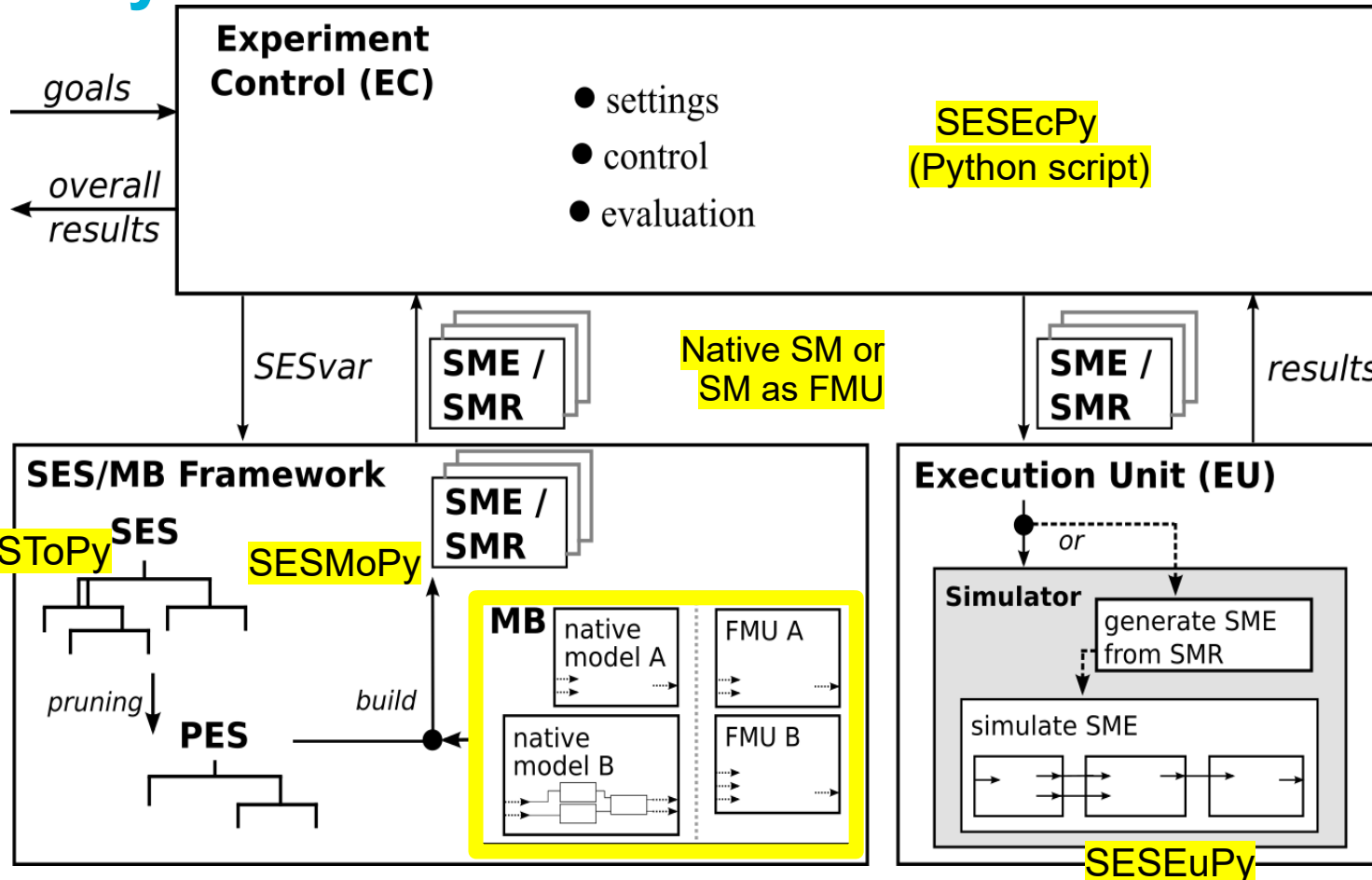


# Extended SES/MB Architecture Python Toolset





# Extended SES/MB Architecture Python Toolset

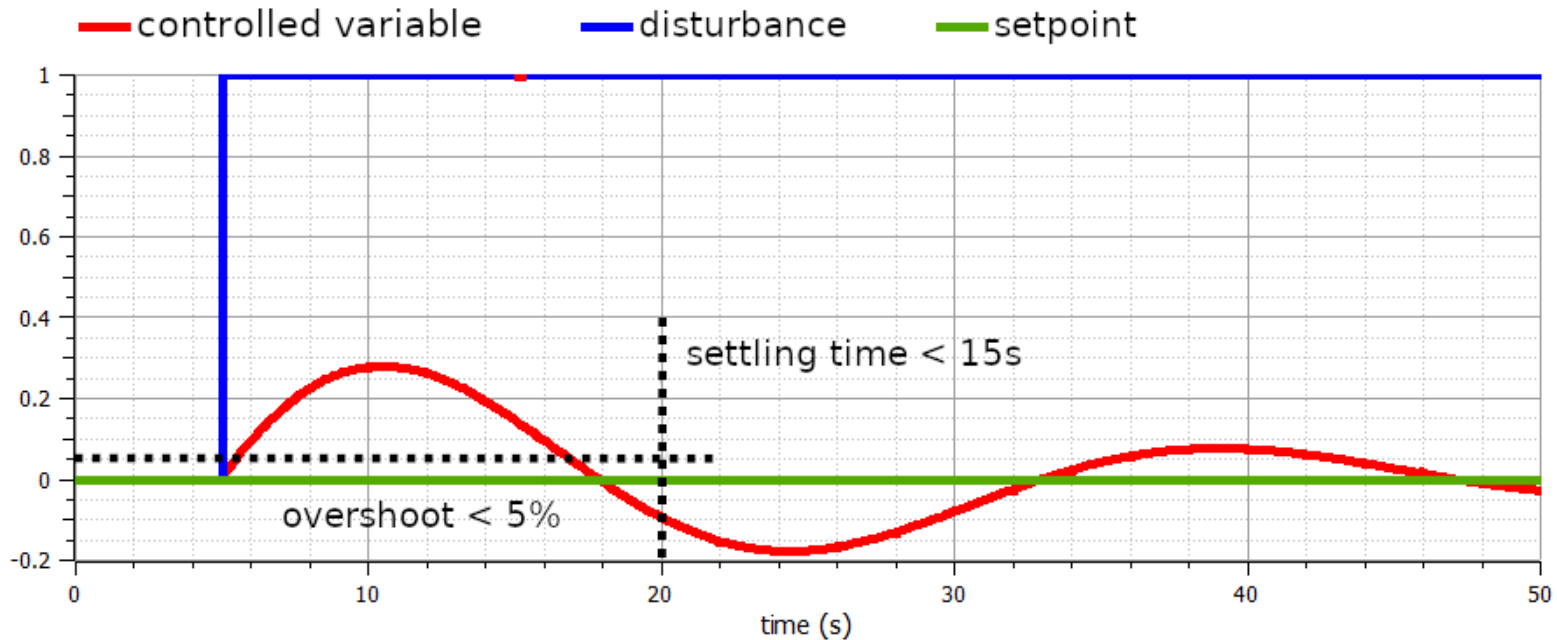
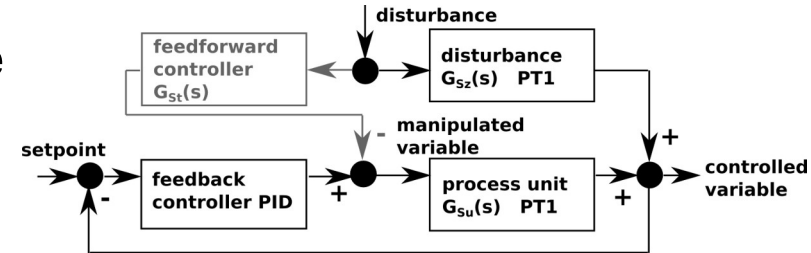






# Automation of Case Study: Control Goals

- 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:  $k=1$ ,  $T_i=1$ ,  $T_d=0$
    - `feedforward=0` simulate with PID:  $k=5$ ,  $T_i=0.5$ ,  $T_d=0$



## Case Study: Experimentation Steps

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

- **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:**

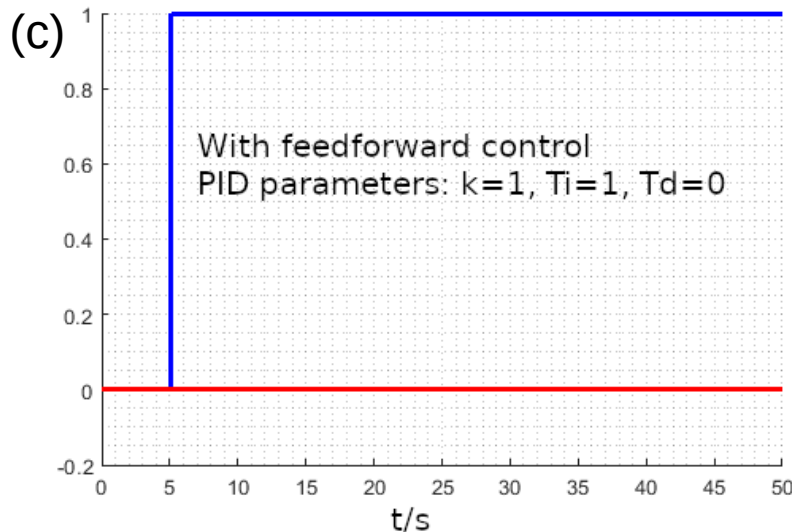
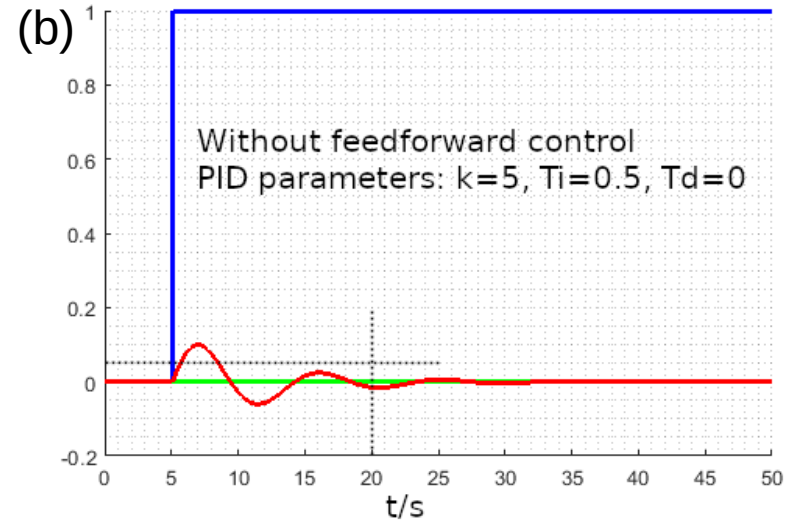
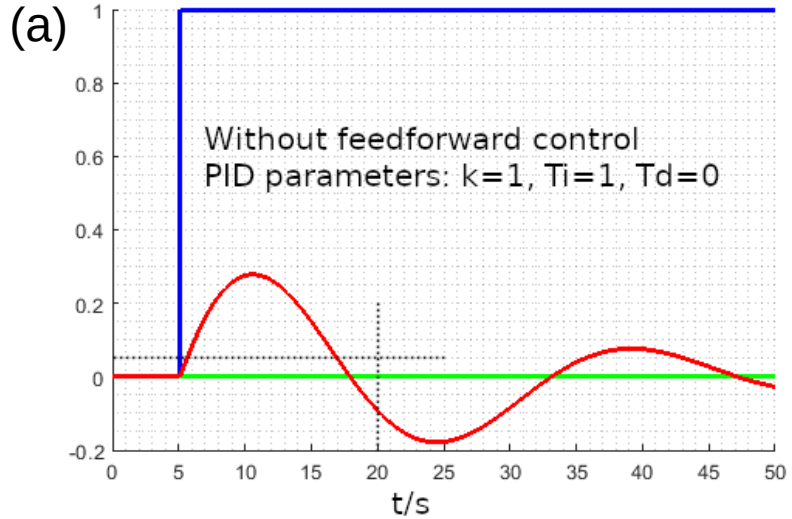
- Return goals cannot be reached with these configurations / parameters

---

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



# Case Study: Simulation Results

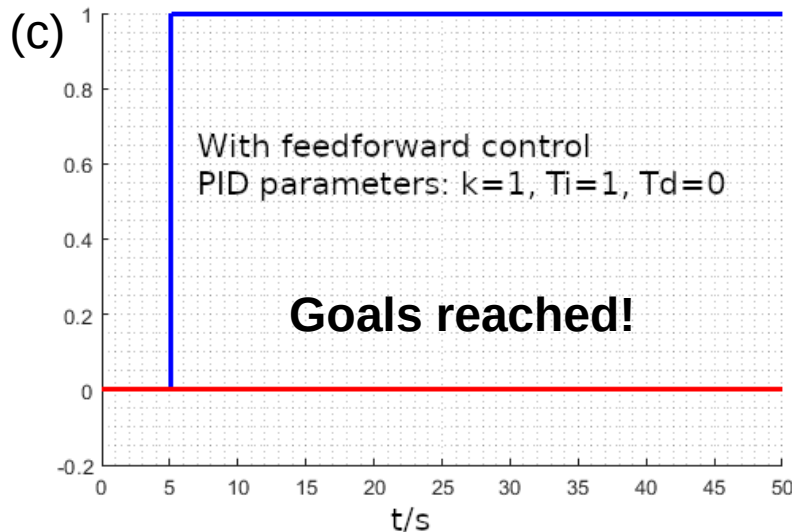
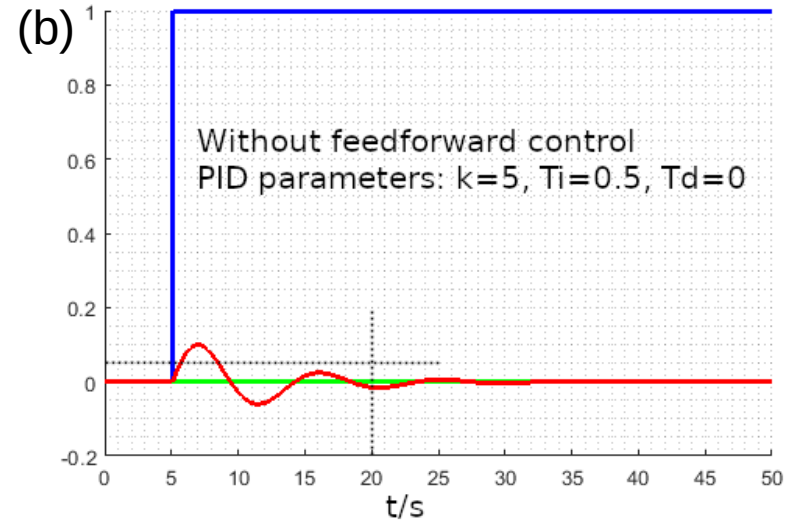
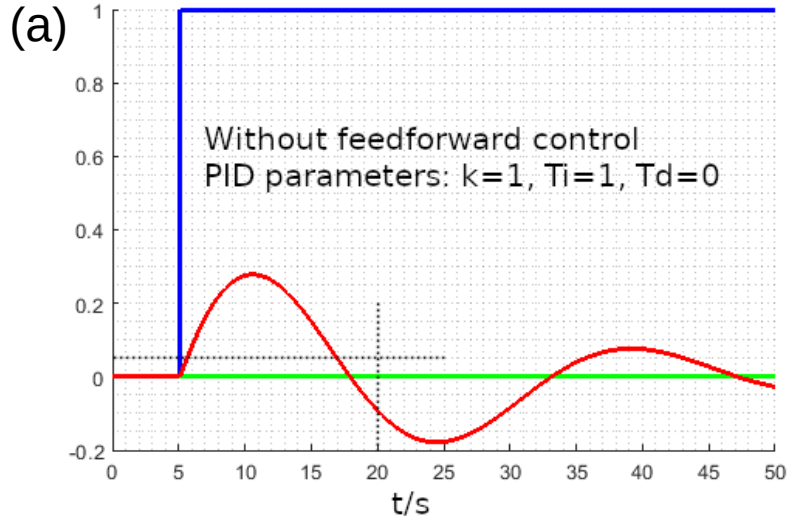


**Key:**  
— setpoint  
— disturbance  
— controlled variable

**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. 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)