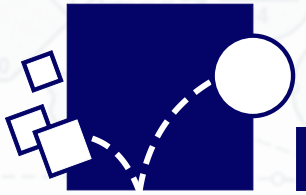
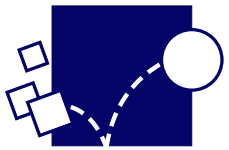


BSL MANAGEMENT SUPPORT

Business Simulation · Learning · Management Science

bsl-support.de



BSL MANAGEMENT SUPPORT

Introducing Object-Oriented Modeling Using Pre-Built Components

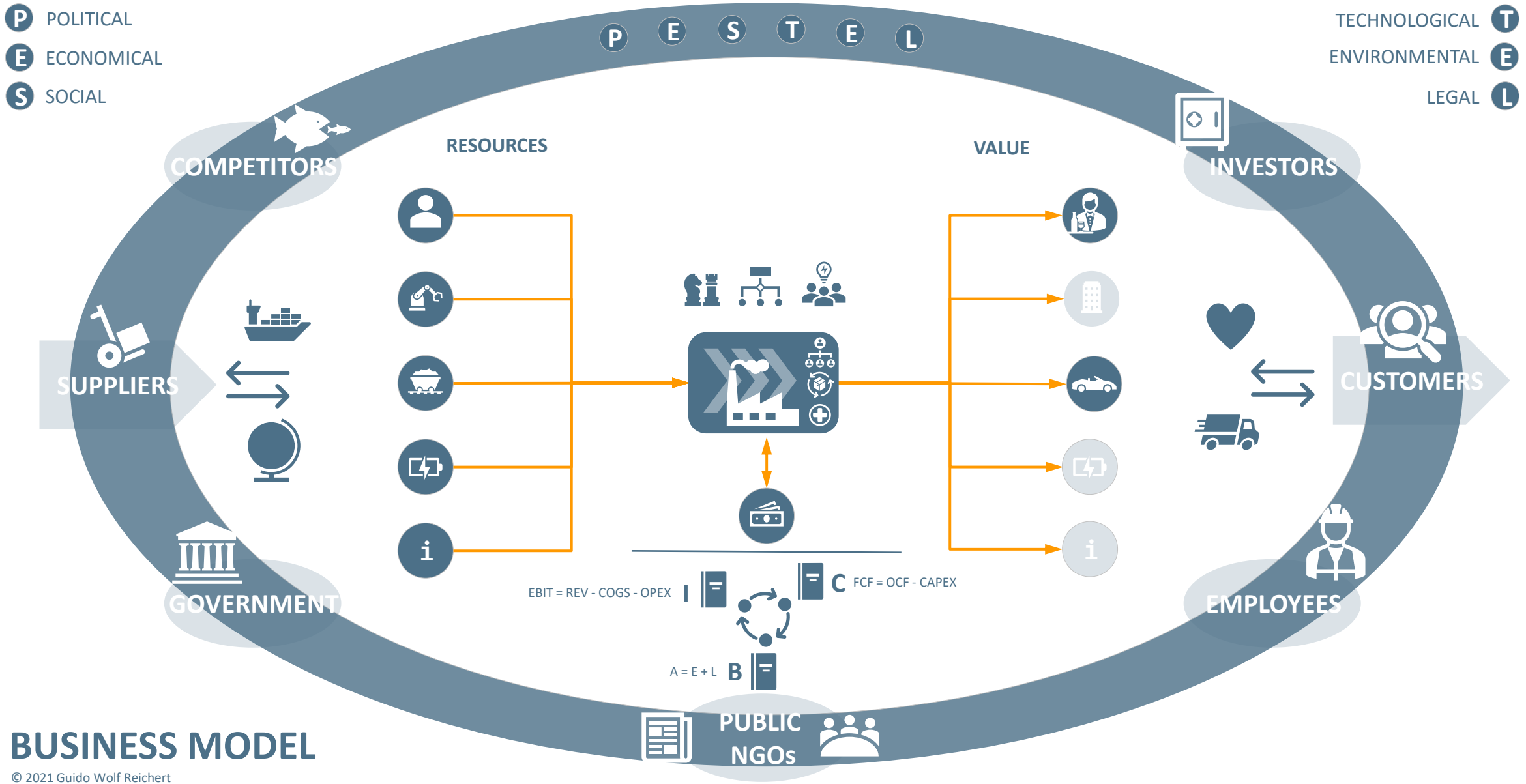
Workshop # 231 at the 40th International System Dynamics Conference

2022-07-22 Frankfurt, Germany

Guido Wolf Reichert (BSL MANAGEMENT SUPPORT) & Jan Brugård (Wolfram MathCore)

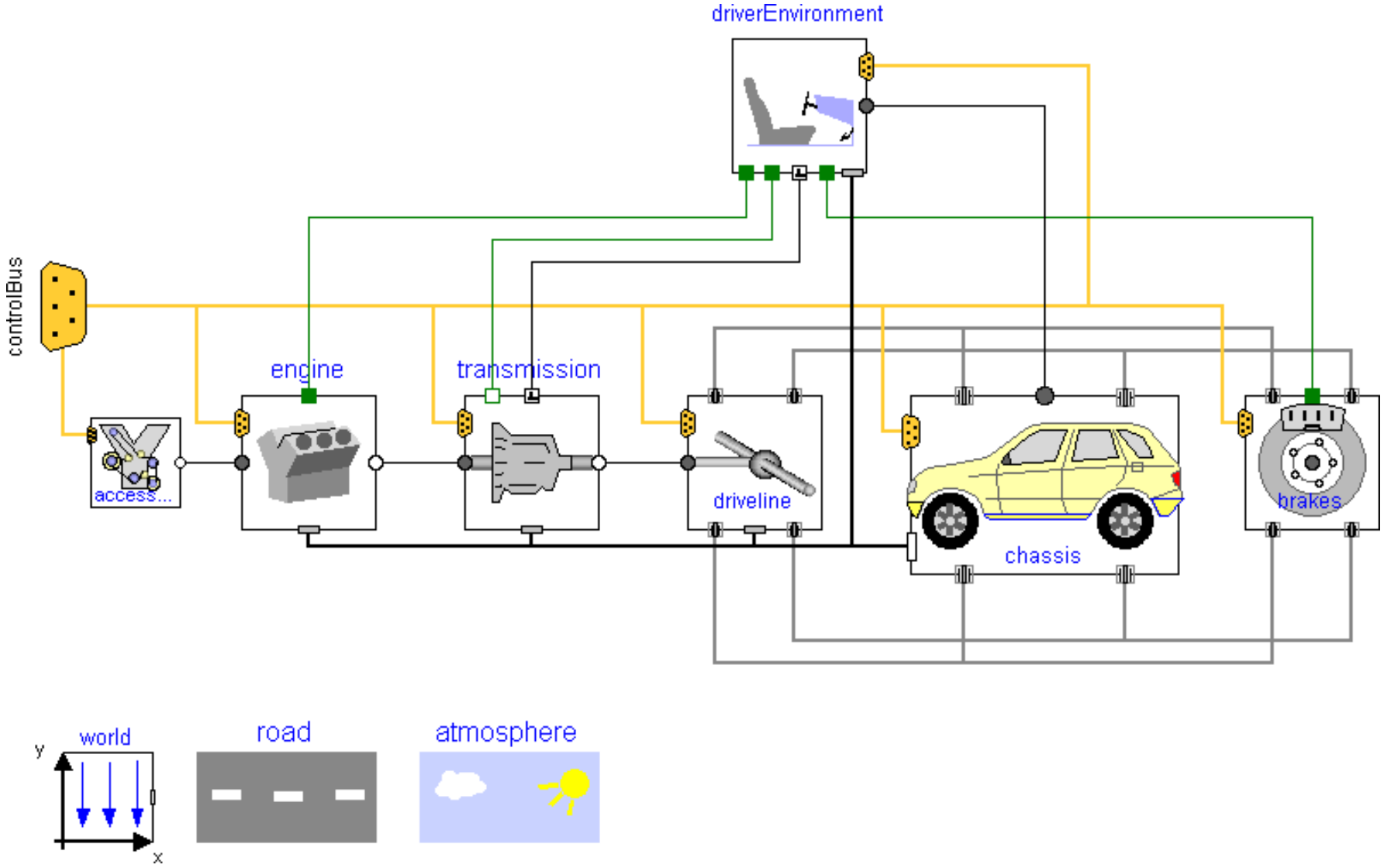
A Quick Introduction To Object-Oriented Modeling

... and some background



BUSINESS MODEL

BSL MANAGEMENT SUPPORT
Business Simulation · Learning · Management Science



Source: Documentation of the free VehicleInterfaces Library

What we want when we model large, complex systems ...

Wishlist

- model an enterprise or organization—or at least relevant parts—in a nested, **hierarchical** fashion (“system of systems” approach)
- (re-)use **pre-built components**, i.e., ideally simply connect and parameterize subsystems with expressive icons
- components should be **self-contained including documentation** and stored separately in readable textual form to **support collaboration** and modern version control, e.g., Git
- have our final model **fit on a single page** and look (a lot) like the Business Model chart, i.e., the real system’s structure should be immediately apparent at one glance
- make models easily deployable for **model exchange and co-simulation** of different models, e.g., models may come from different sources

The Business Simulation Library—A quick overview

Overview

Overview of the Main Packages

Icon	Name	D
	CausalLoop	A (C
	Stocks	C a
	Flows	P
	SourcesOrSinks	Fl
	Converters	In
	InformationSources	Ex
	MoleculesOfStructure	P *su*

Overview of the Main Packages

Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD⁺)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)

Source: https://github.com/bslMS/BusinessSimulation

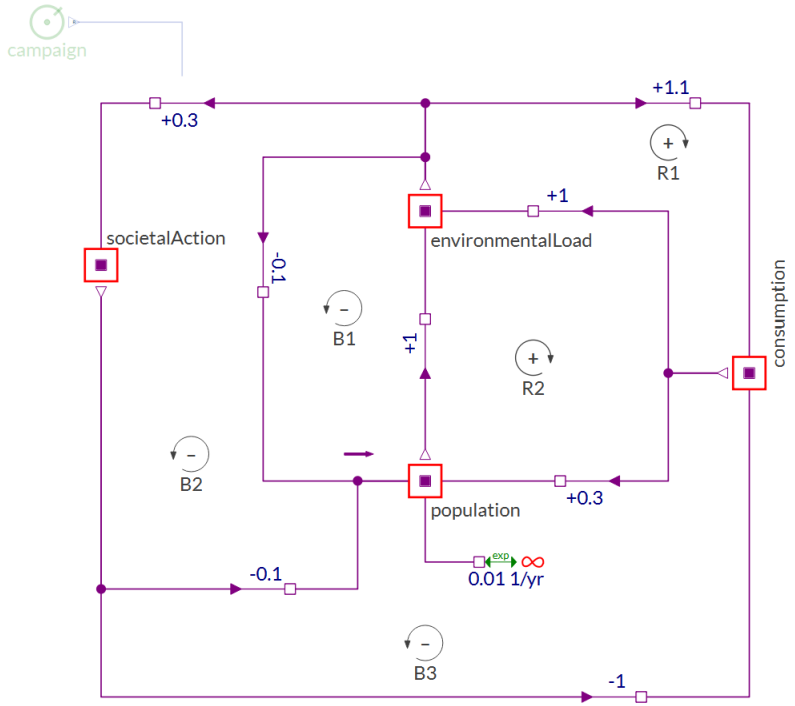
Source: https://github.com/bslMS/BusinessSimulation

Use coefficients of proportionality or elasticity and elementary processes to have causal loop diagrams spell out "*agile system dynamics*"

Overview

Overview of the Main Packages

Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD^+)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)



Source: https://github.com/bslMS/BusinessSimulation

BSL MANAGEMENT SUPPORT
Business Simulation · Learning · Management Science

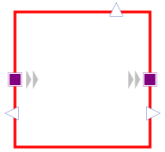
Next to conventional reservoirs the BSL introduces *dynamic stocks* to finally follow up on Jay Forrester's suggestions (Industrial Dynamics, 1961)

Overview

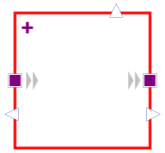
Overview of the Main Packages

Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD$^+$)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)

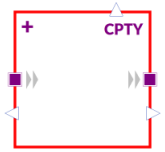
Source: https://github.com/bslMS/BusinessSimulation



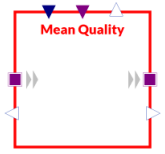
InformationLevel



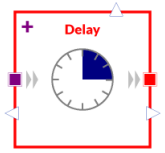
MaterialStock



CapacityRestrictedStock



HinesCoflow



DelayN

BSL MANAGEMENT SUPPORT
Business Simulation · Learning · Management Science

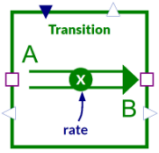
Next to unidirectional and bi-directional flows there are also interactions in the library, which let us fit predator-pray dynamics into a single component

Overview

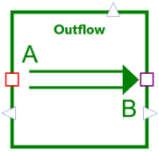
Overview of the Main Packages

Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD$^+$)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)

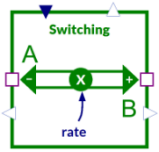
Source: https://github.com/bslMS/BusinessSimulation



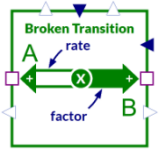
Transition



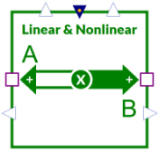
OutflowDynamicStock



Switching



BrokenTransition



ComplexInteraction

BSL MANAGEMENT SUPPORT
Business Simulation · Learning · Management Science

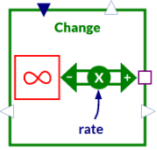
A cloud and a flow are succinctly merged into sources or sinks to model processes of growth and decline at a model's boundary

Overview

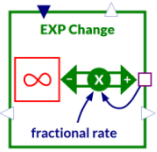
Overview of the Main Packages

Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD$^+$)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)

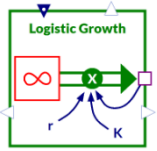
Cloud



ExogenousChange



ExponentialChange



LogisticGrowth

Source: https://github.com/bslMS/BusinessSimulation

Converters are the work horses of system dynamics models and you will find a whole variety in the library

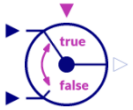
Overview

Overview of the Main Packages

Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD$^+$)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)

Source: https://github.com/bslMS/BusinessSimulation

Discrete Delays ...



Logical Converters ...

Lookup Converters ...

Vector Converters ...

Regular Converters ...

While we aim at maximal endogeneity, information inputs are needed—at least for testing

Overview

Overview of the Main Packages

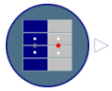
Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD$^+$)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)

Source: https://github.com/bslMS/BusinessSimulation

PulseInput

RampInput

TimeInput



LinearTimeTable

ExogenousData

BSL MANAGEMENT SUPPORT
Business Simulation · Learning · Management Science

Widely known “Molecules of Structure” are now to be found nicely structured according to their *interfaces*, i.e., connectors

Overview

Overview of the Main Packages

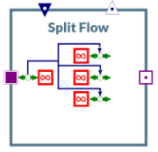
Icon	Name	Description
	CausalLoop	Agile system dynamics modeling with quantitative causal loop diagrams (CLD$^+$)
	Stocks	Containers ("reservoirs") used to represent entities that have been stored in a specific state
	Flows	Processes that move entities from one stock to another at a specific rate
	SourcesOrSinks	Flows into or out of a stock with infinite capacity at a system's boundary
	Converters	Information processing (blocks)
	InformationSources	External information input
	MoleculesOfStructure	Pre-built components to model *information processing*, *decision making* and *subsystems* in general (blocks, incubators, transceivers, and actuators)

Source: https://github.com/bslMS/BusinessSimulation

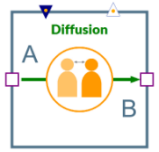
Information Processing ...

Policy ...

Incubators ...

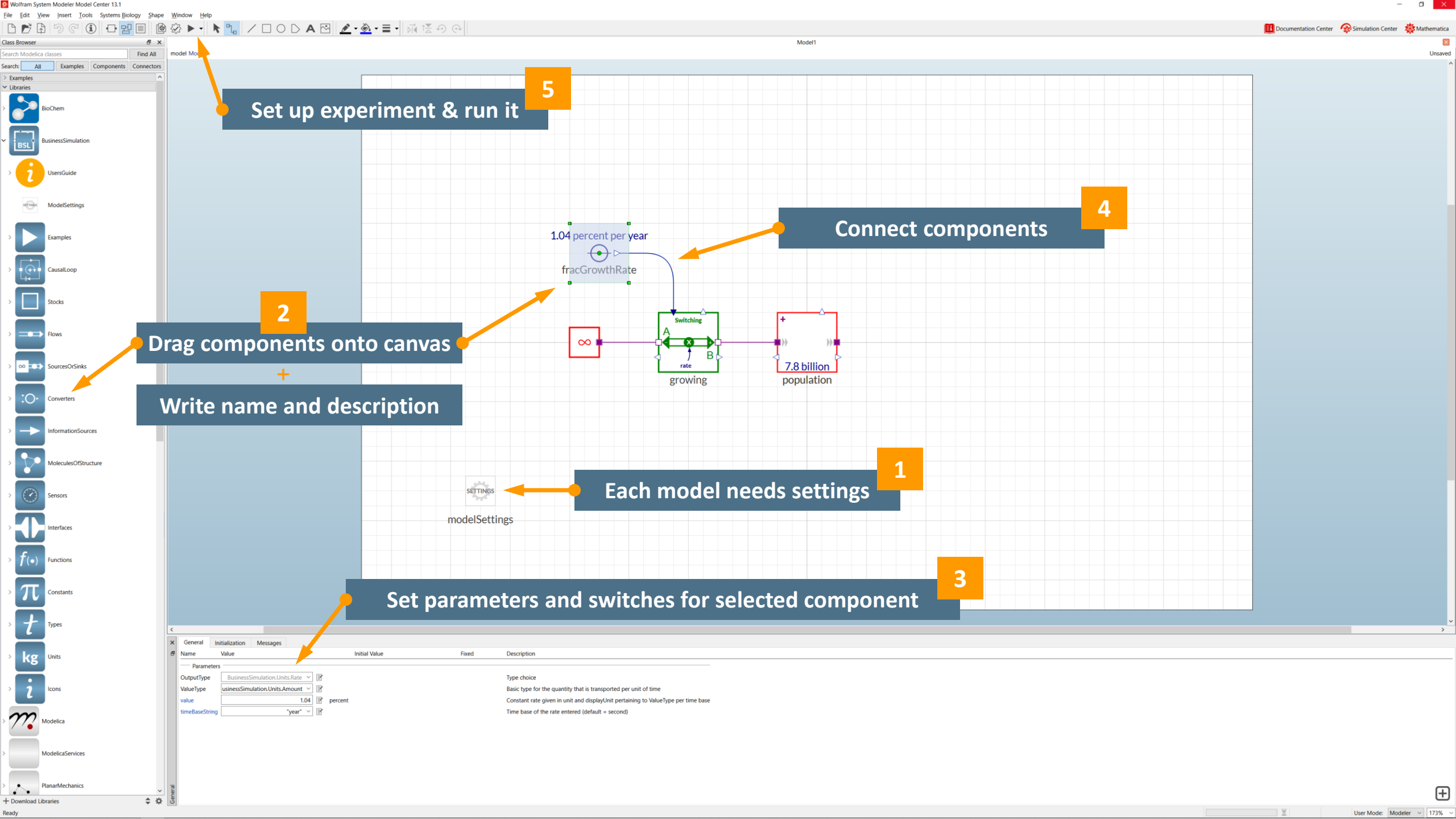


Transceivers ...

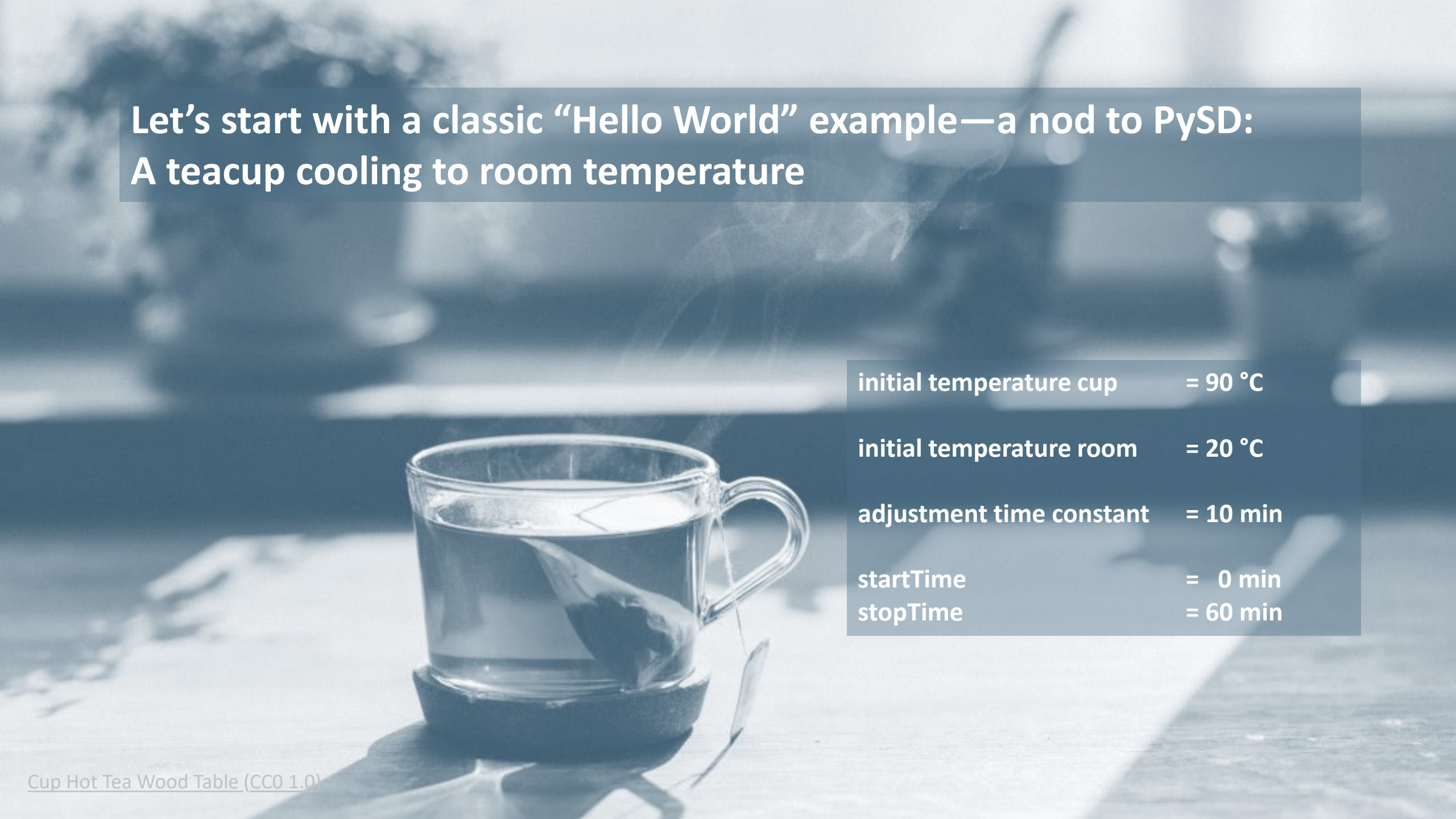


Actuators ...

BSL MANAGEMENT SUPPORT
Business Simulation · Learning · Management Science


Wolfram System Modeler Model Center 13.1
File Edit View Insert Tools Systems Biology Shape Window Help
Documentation Center
Simulation Center
Mathematica
Class Browser
Search Modelica classes
Find All
Search: All Examples Components Connectors
Examples
Libraries
BioChem
BusinessSimulation
UsersGuide
ModelSettings
Examples
CausalLoop
Stocks
Flows
SourcesOrSinks
Converters
InformationSources
MoleculesOfStructure
Sensors
Interfaces
Functions
Constants
Types
Units
Icons
Modelica
ModelicaServices
PlanarMechanics
Download Libraries
Model1
Unsaved
5
Set up experiment & run it
4
Connect components
1.04 percent per year
fracGrowthRate
2
Drag components onto canvas
+
Write name and description
Switching
A
B
rate
growing
7.8 billion
population
SETTINGS
modelSettings
1
Each model needs settings
3
Set parameters and switches for selected component
General
Initialization
Messages
Name
Value
Initial Value
Fixed
Description
Parameters
OutputType
BusinessSimulation.Units.Rate
Type choice
ValueType
usinessSimulation.Units.Amount
Basic type for the quantity that is transported per unit of time
value
1.04
percent
Constant rate given in unit and displayUnit pertaining to ValueType per time base
timeBaseString
"year"
Time base of the rate entered (default = second)
Ready
User Mode:
Modeler
173%

Let’s start with a classic “Hello World” example—a nod to PySD: A teacup cooling to room temperature

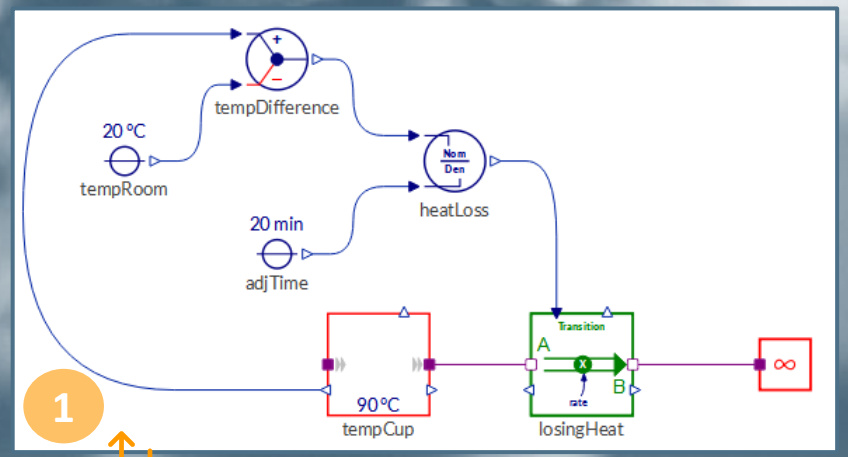


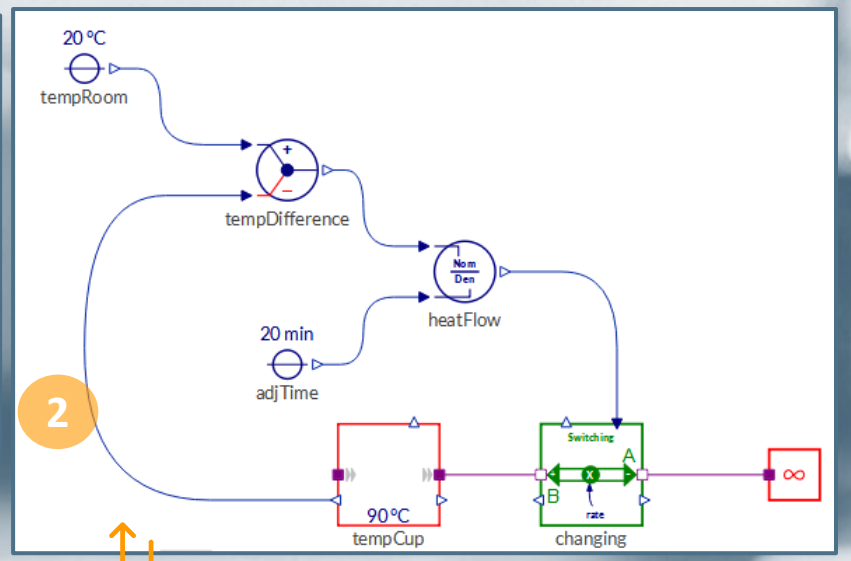
initial temperature cup	= 90 °C
initial temperature room	= 20 °C
adjustment time constant	= 10 min
startTime	= 0 min
stopTime	= 60 min

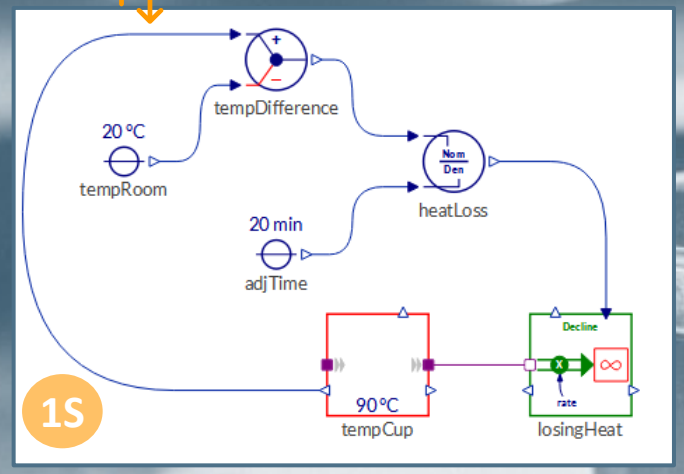
Cup Hot Tea Wood Table (CC0 1.0)

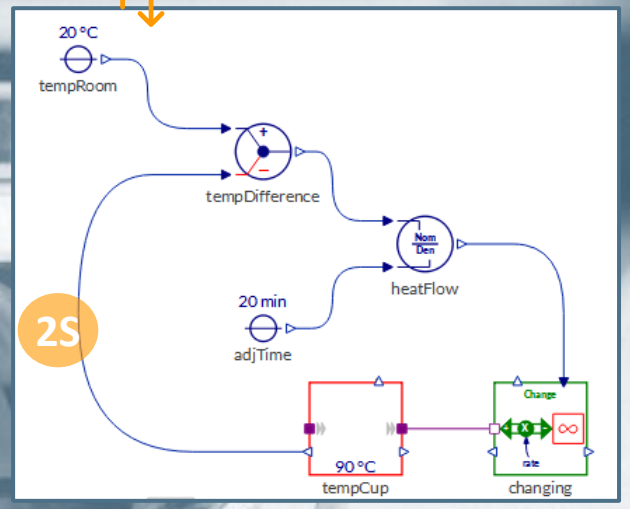
Let's move from a straightforward textual model to a component-based version ...

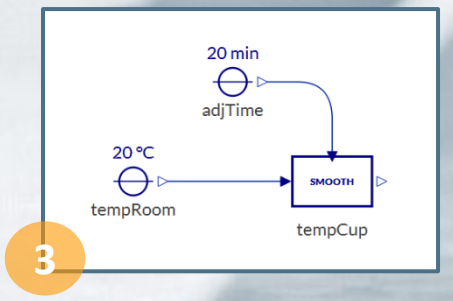
```
model TeaCupTextual "Textual model for cooling cup"
 import BusinessSimulation.Units.{Time, Rate};
 extends BusinessSimulation.Icons.Example;
  // parameters
 parameter Real initTempCup (unit = "degC") = 90 "Initial temperature in the cup";
 parameter Real tempRoom (unit = "degC") = 20 "Temperature in the room";
 parameter Time adjTime (displayUnit = "min") = 600 "Time constant for adjustment process";
  // stock & flow variables
 Real tempCup (start = initTempCup, unit = "degC") "Temperature in the cup is modeled as a stock";
 Rate heatFlow (displayUnit = "1/min") "Outflow of heat from the cup";
equation
 heatFlow = (tempRoom - tempCup) / adjTime;
 der (tempCup) = heatFlow;
 »;
end TeaCupTextual;
```











higher

Components

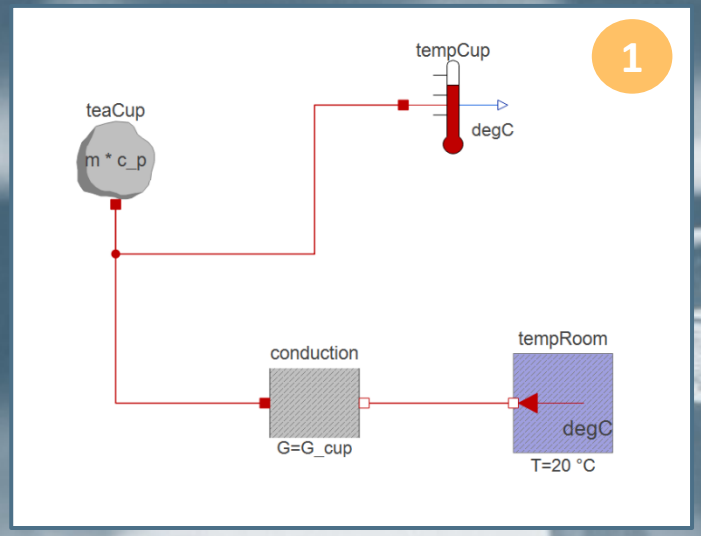
lower

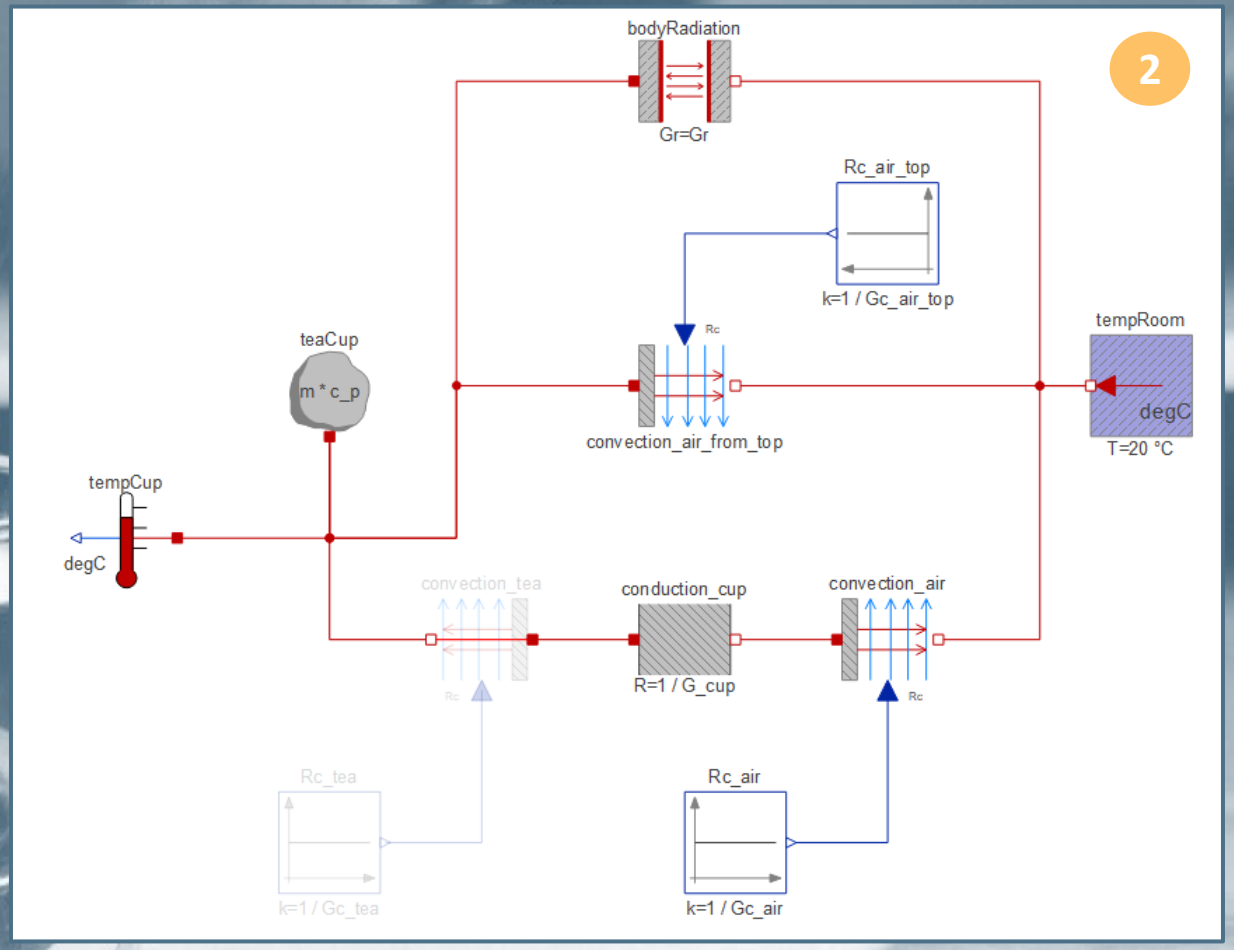
lower

Reusability

higher

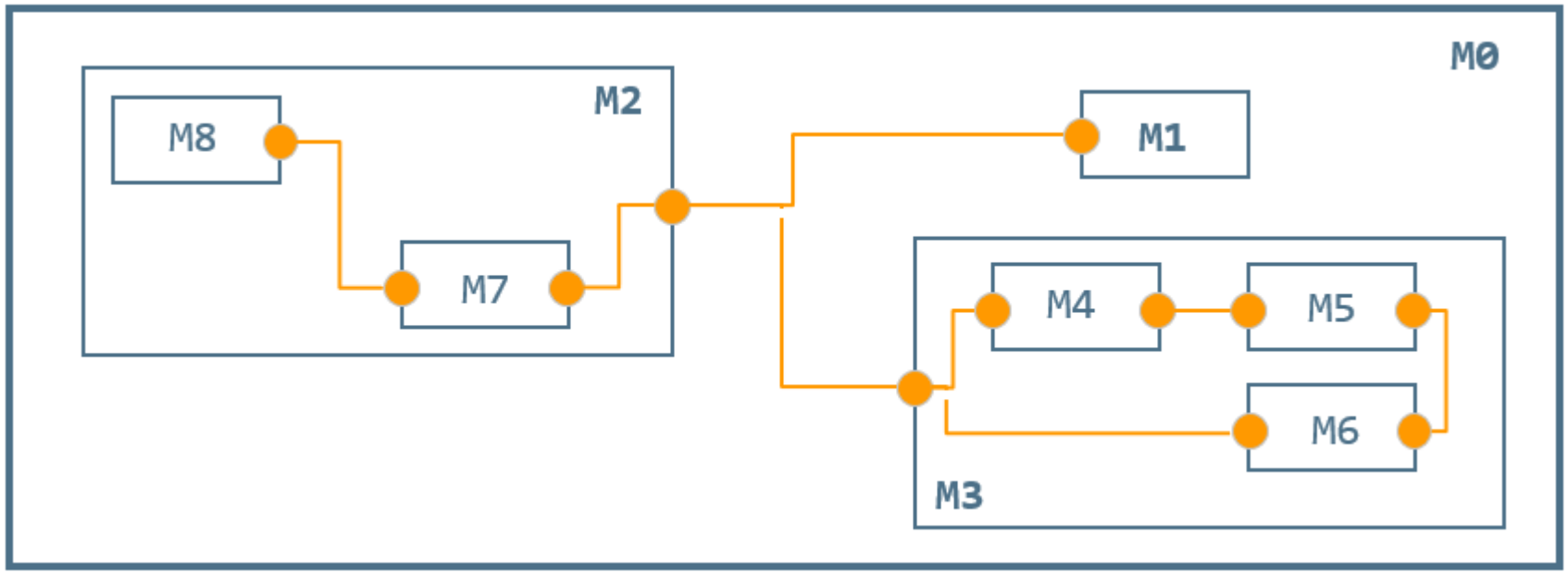
Why not consider *physical components* for physical processes?





In Modelica we use pre-built models (components), which exchange information or matter via interfaces (connectors) in a nested fashion

Hierarchical Modeling

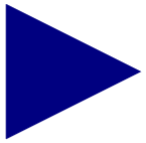


SCHEMATIC

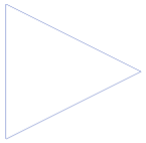
In the Business Simulation Library there are four basic connectors: Causal input/output connectors, and acausal stock and flow ports

Connectors

Causal Connectors for Information Exchange



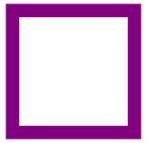
`RealInput`



`RealOutput`

Acausal Connectors for Physical Exchange

`StockPort`



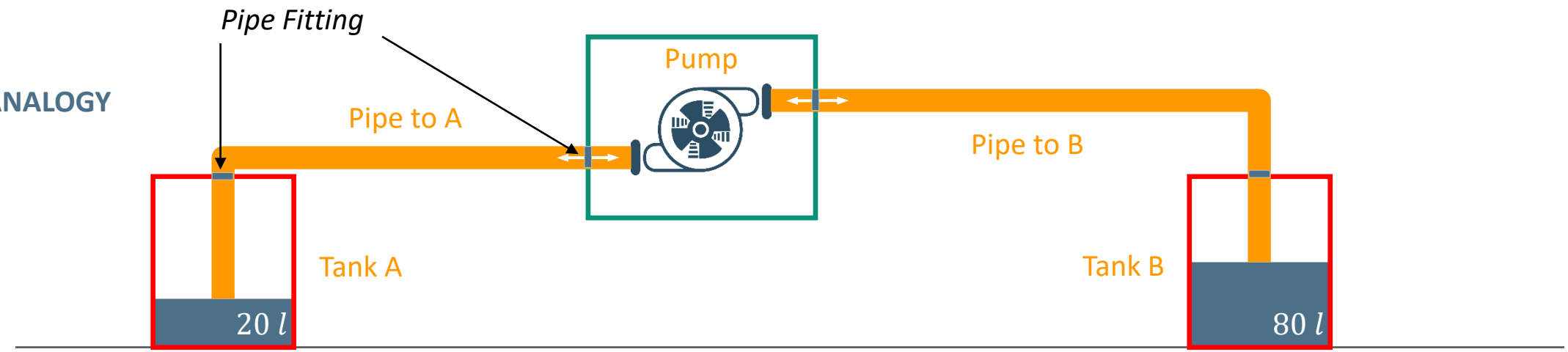
`FlowPort`

A dot inside these icons will indicate a multi-connector (e.g., a vector version)

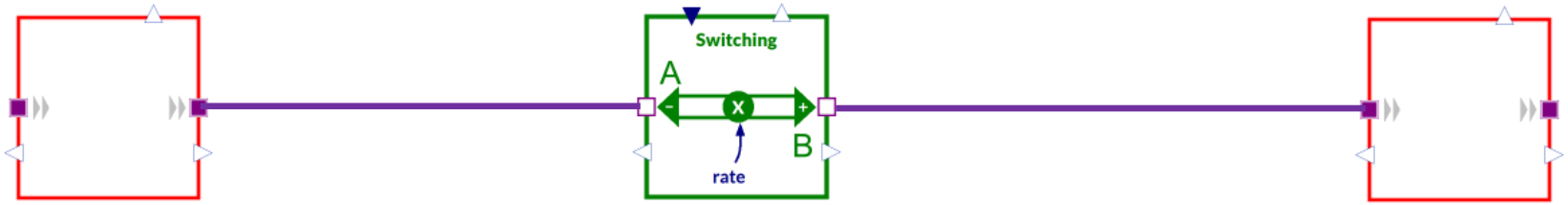
In SD we love to use physical analogies like reservoirs, pumps, and pipes. But why should a pipe or its *fittings* be "causal"?

Acausal Connectors (1)

PHYSICAL ANALOGY



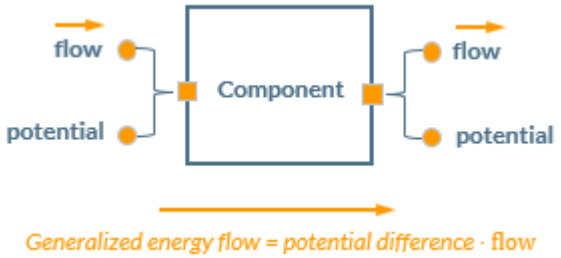
COMPUTER MODEL



Acausality promotes reusability of connectors and components in physical modeling.

Each acausal connector has two variables: a *flow* and a *potential*—as we usually don't track energy flows in SD we replace *potential* with *stock*

Acausal Connectors (2)



Listing 3. Stock port connector

```
connector StockPort "Connector for stock components"
  import BusinessSimulation.Units.Rate;
  Real stock "Current amount of 'mass' in the stock";
  flow Rate rate "Flow that affects the stock";
  output Boolean stopInflow "= true, if nothing can flow into the stock";
  output Boolean stopOutflow " = true, if nothing can flow out of the stock";
end StockPort;
```

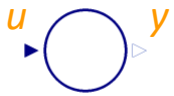
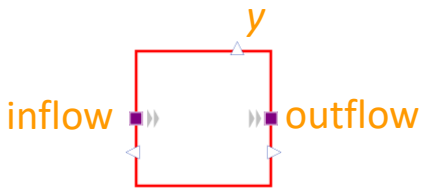
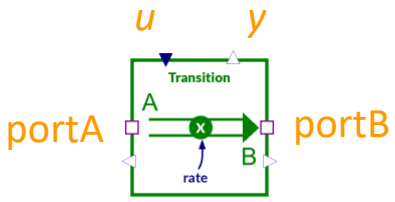
Listing 4. Flow port connector

```
connector FlowPort "Used to represent stock and flow connections"
  import BusinessSimulation.Units.Rate;
  Real stock "The current amount of 'mass' in a connected stock";
  flow Rate rate "Flow that affects the stock";
  input Boolean stopInflow "= true, if nothing can flow into a connected stock"
    ;
  input Boolean stopOutflow "= true, if nothing can flow out of a connected
    stock";
end FlowPort;
```

Flows will observe Boolean flags for restricted stocks, e.g., material stocks cannot be drained below zero.

In Modelica we are basically writing a system of differential-algebraic *equations* in peacemeal fashion, so that it can be solved

Equation Puzzle

	CONNECTORS	EQUATIONS
CONVERTERS **...** *transform information input*		$y = f(u)$
STOCKS *... solve ODE for internal state*		$\mathrm{der}(x) = \mathrm{inflow.rate} + \mathrm{outflow.rate}$ $\mathrm{inflow.stock} = x$ $\mathrm{outflow.stock} = x$ $y = \mathrm{inflow.stock}$
FLOWS *... set rates of transition*		$\mathrm{portA.rate} = u$ $\mathrm{portB.rate} = -\mathrm{portA.rate}$ $y = \mathrm{portA.rate}$

Note: A *positive* rate per definition flows *into* the component in Modelica.

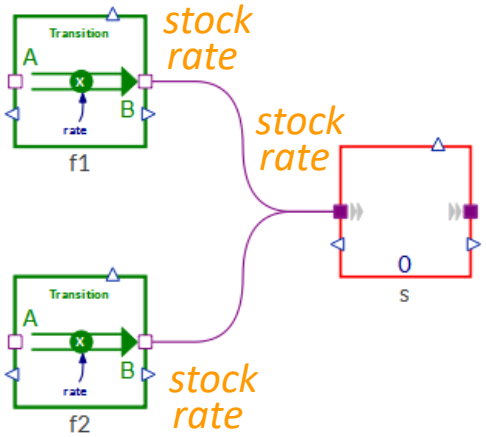
Connections introduce equations behind the scenes—the equations for acausal connections arise from Kirchhoff's Laws

Connect-Equations

Connect-Equation (Causal)

DIAGRAM

Connect-Equations (Acausal)



TEXT

```
connect(converter.y, func.u)
```

```
connect(f1.portB, s.inflow)
connect(f2.portB, s.inflow)
```

EQUATIONS

$$\mathbf{converter.y} = \text{func.u}$$

$$\mathbf{f1.portB.rate} + \mathbf{f2.portB.rate} + \text{s.rate} = 0$$

$$\text{f1.portB.stock} = \mathbf{s.stock}$$

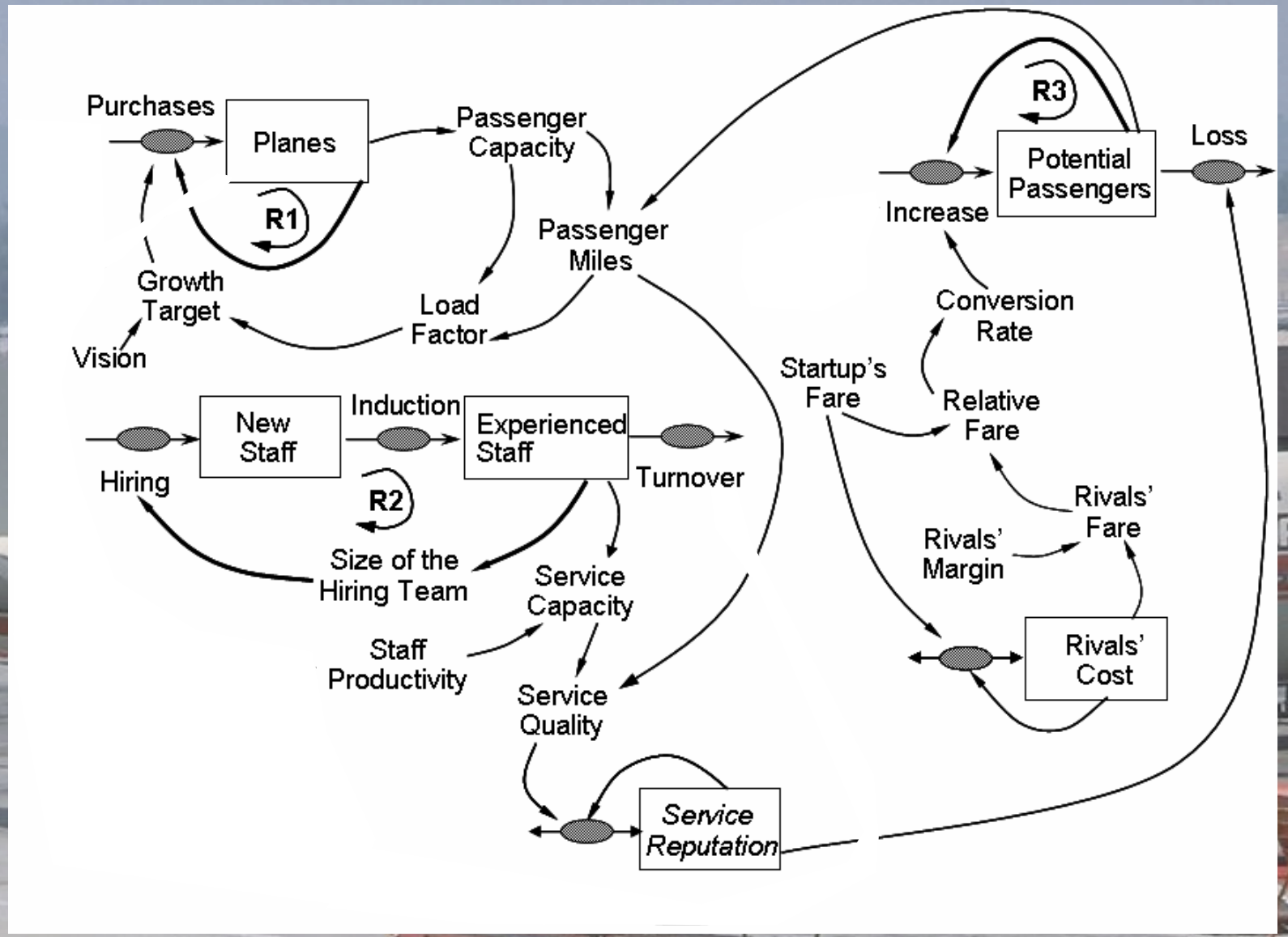
$$\text{f2.portB.stock} = \mathbf{s.stock}$$

Known values in equations printed in bold face, i.e., flows set rates and stocks determine their internal state from solving **der**(x) = *sum of flows*

Building More Complex Models

... in a way that supports collaboration and avoids repetition

J.D.W. Morecroft, "System Dynamics, RBV and Behavioural Theories of Firm Performance: Lessons from People Express," *Proceedings of the 26th International Conference of the System Dynamics Society*, 2008.



J.D.W. Morecroft, "System Dynamics, RBV and Behavioural Theories of Firm Performance: Lessons from People Express," *Proceedings of the 26th International Conference of the System Dynamics Society*, 2008.

Eduard Marmet, https://wolfr.am/15blpZqTI, CC BY-SA 3.0
A Boeing 747 of People Express at London Gatwick Airport (June 1983).

Collaboration & Deployment

Using Git and GitHub

A Stack Overflow survey shows that Git really is the main tool used by software developers for version control and collaboration

Version Control Choices

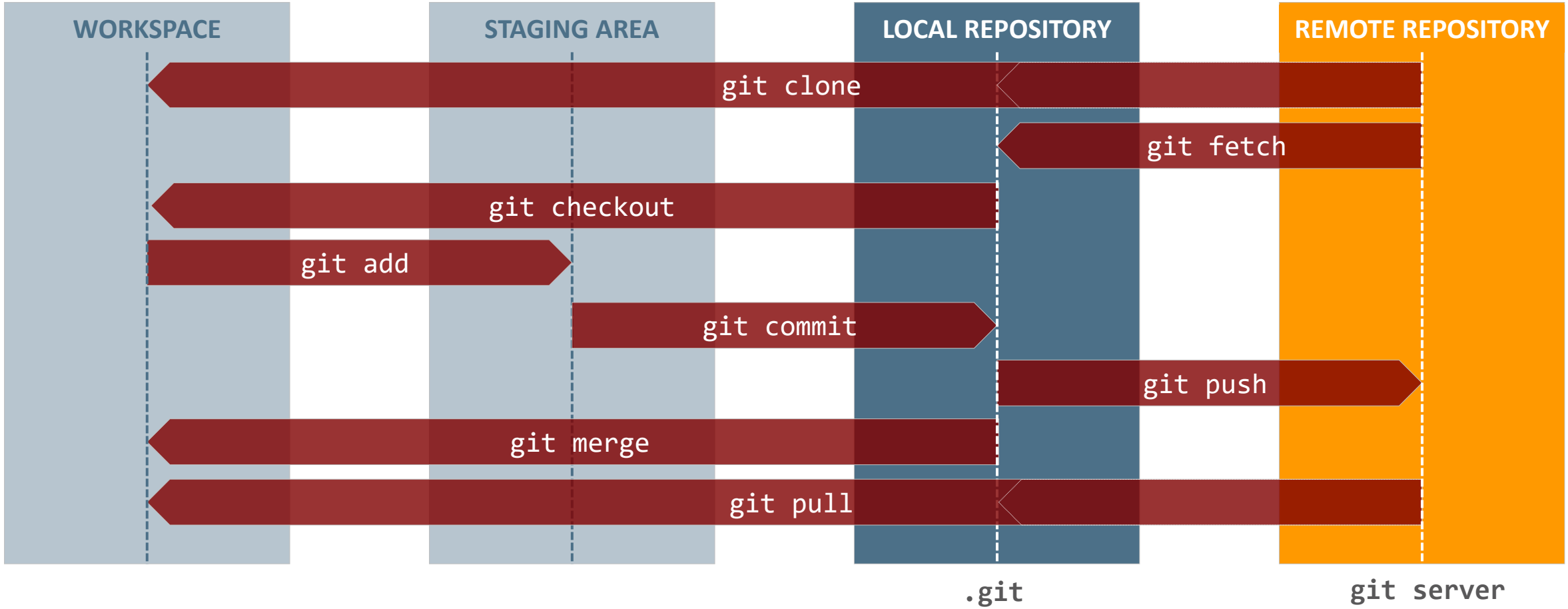
Version control systems used by responding developers:

Name	2015	2017	2018	2022
Git	69.3%	69.2%	87.2%	93.9%
Subversion	36.9%	9.1%	16.1%	5.2%
TFVC	12.2%	7.3%	10.9%	[ii]
Mercurial	7.9%	1.9%	3.6%	1.13%
CVS	4.2%	[ii]	[ii]	[ii]
Perforce	3.3%	[ii]	[ii]	[ii]
VSS	[ii]	0.6%	[ii]	[ii]
ClearCase	[ii]	0.4%	[ii]	[ii]
Zip file backups	[ii]	2.0%	7.9%	[ii]
Raw network sharing	[ii]	1.7%	7.9%	[ii]
Other	5.8%	3.0%	[ii]	[ii]
None	9.3%	4.8%	4.8%	4.3%

Source: Wikipedia contributors. "Git.“ Wikipedia, The Free Encyclopedia, 28 Jun. 2022.

Local changes are *staged*, then *committed*, and finally *pushed* into the remote repository; branches typically get *merged* upon a *pull request*

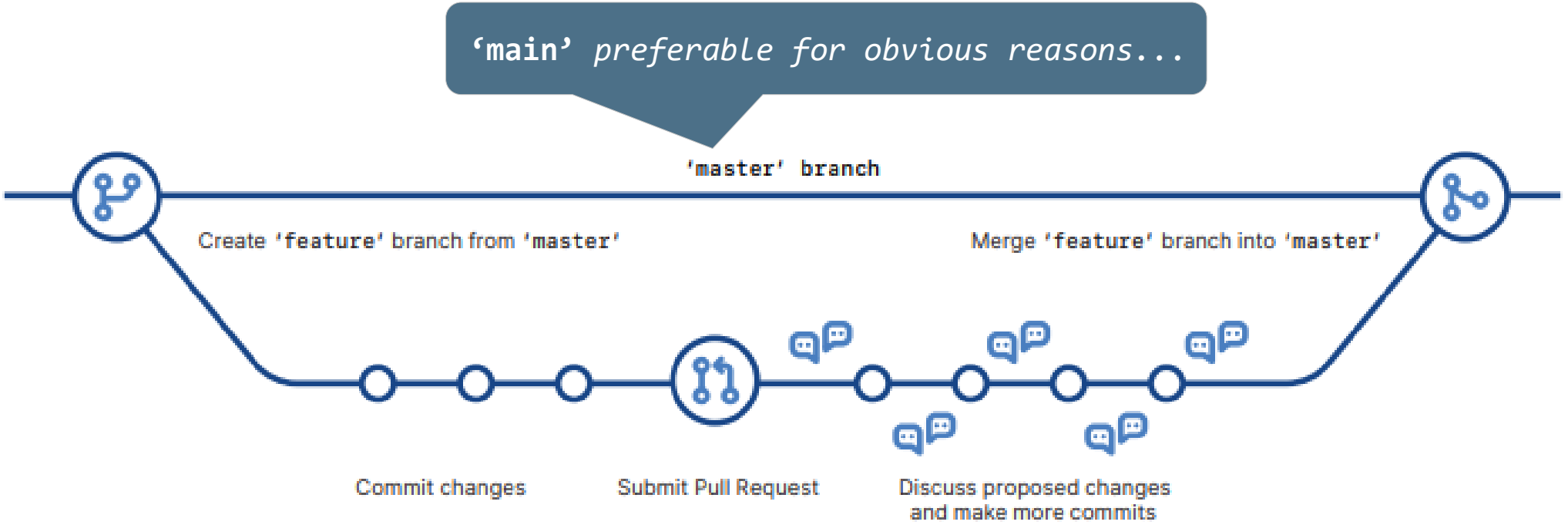
Git Workflows



Source: https://ndpsoftware.com/git-cheatsheet.html#loc=index

Branching—next to speed and its distributed organization—is what makes Git stand out among version control solutions

Branches



Source: GitHub. “Git Cheat Sheet.” Link: https://training.github.com/downloads/github-git-cheat-sheet.pdf

After setting up a local repository we can push this to a central repository on a git server

Set up Git

STEP 1: Set up a local repository for a project

1. Prevent empty directories from being ignored by adding an empty **`.gitignore`** file.
2. Ignore certain files (e.g., binaries) and directories by explicitly listing these in the **`.gitignore`** file in the root directory.
3. Configure Git to handle line endings in a compatible fashion between operating systems:

```
git config --global core.autocrlf true   (Windows)
git config --global core.autocrlf input  (Unix)
```

4. Setup a repository in the root directory and import files:

```
git init
git add .
git commit –m “initial commit”
```

STEP 2: Prepare remote bare repository for distribution

1. Create a new public or private repository on a git server (e.g., GitHub, GitLab, Bitbucket) that should be completely empty (no README).
2. Push local repository to remote (shortname: **`origin`**) and make the remote main branch the tracking reference:

```
git remote add origin <url>
git push –set-upstream origin main
```

3. All members of the development team can now clone the remote repository:

```
git clone <url>
```

Turn to https://git-scm.com/ for documentation and reference.

BSL MANAGEMENT SUPPORT

Business Simulation · Learning · Management Science

Dipl.-Kfm.
Guido Wolf Reichert

Schauenburgerstrasse 116 · 24118 Kiel (Germany)

+49 431 90 89 89 02
+49 431 90 89 89 03

management@bsl-support.de
www.bsl-support.de