Where to put physical flow variables?

SysMLPISF FTF
SimVars & Port Types

What block owns simulation variables (SimVariable) for physical interactions?

Options:

- Flow property types (kinds of things flowing).
- Port/block types (kinds of things / boundaries through which physical substances flow).
- SimBlocks (separate blocks from flow property types and port/block types).
1. SimVars on Flow Prop Types?

- Flow rate (and potential) are relative to ports or object boundaries the substances flow through.

- Flow rate and potential are not characteristics only of the kind of thing flowing.
SimVariables for all flow properties on the port/block type.

Reflects that potential and flow rate are relative to the port/block boundary.
3. SimVars on SimBlock?

SimVariables on separate block, not flow property type or port/block type.
### Pros/Cons

<table>
<thead>
<tr>
<th>Sim variables on ↓</th>
<th>Clutter on block / port type</th>
<th>Fidelity to the thing being modeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Flow property types (the kind of thing flowing)</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>2. Port types</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>3. SimBlocks (separate from flow property type &amp; port types)</td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>
Lack of fidelity to the thing being modeled can be resolved by

- This kind of simulation treats the things flowing as continuous (e.g., charge, not electrons).
  - Flow properties always have values (always giving potential and flow rate).
  - Flow properties can be typed by role types\(^1\) (e.g., FlowingCharge) specialized from substance types.
    - Role types only classify substances that are values of flow properties, adding sim variables that are relative to the port.
If fidelity issues are resolved, we can pick solution with least clutter (#1).

Physical interaction library would be:
Signal flow library would be:

- Real signal flows
  - RealSignalElement
    - rSig : Real
  - RealSignalInElement
    - in rSig : Real {redefines rSig}
  - RealSignalOutElement
    - out rSig : Real {redefines rSig}

- Boolean signal flows
  - BooleanSignalElement
    - bSig : Boolean
  - BooleanSignalInElement
    - in bSig : Boolean {redefines bSig}
  - BooleanSignalOutElement
    - out bSig : Boolean {redefines bSig}

- Integer signal flows
  - IntegerSignalElement
    - iSig : Integer
  - IntegerSignalInElement
    - in iSig : Integer {redefines iSig}
  - IntegerSignalOutElement
    - out iSig : Integer {redefines iSig}
The stereotypes would be:

§ «stereotype»
PhSConstant

UML::Property

§ «stereotype»
PhSVariable

isContinuous : Boolean = true
isConserved : Boolean = false
changeCycle : Integer = 0