The Semantically Reflected Digital Twin

CAMPaM 2023 Tutorial

Einar Broch Johnsen
Eduard Kamburjan

University of Oslo

Today

- **Part I Digital Twins Introduction: Concepts and Engineering Perspectives**
- **Part II** Modelling Knowledge using Semantic Technologies
- **Part III** Semantically Reflected Digital Twins

Digital Twins — The Hype

Digital twins are an emerging, enabling technology for industry to transition to the next level of digitisation

Digital Twins — The Hype

Digital twins are an emerging, enabling technology for industry to transition to the next level of digitisation

Increasing traction

1. Digital twins: a means to **understand** and **control** assets in nature, in industry, and in society at large
2. Companies increasingly create digital twins of their physical assets

Digital Twins — The Hype

Digital twins are an emerging, enabling technology for industry to transition to the next level of digitisation

Increasing traction

1. Digital twins: a means to **understand** and **control** assets in nature, in industry, and in society at large
2. Companies increasingly create digital twins of their physical assets

Success stories

1. GE experienced 5–7% increase of energy production from digitizing wind farms
2. Johns Hopkins Hospital's centre for clinical logistics reported 80% reduction of operating theatre holds due to delays
3. For the Johan Sverdrup oil field, digital twin innovations have boosted earnings by $216 million in one year

Digital Twins: Emerging Engineering Discipline

- DTs originally conceived at NASA for the space program.
- They have emerged as an engineering discipline, based on **best practices**

NASA's definition of a DT

*"an **integrated multi-physics, multi-scale**, probabilistic simulation of a vehicle or system that uses the **best available** physical models, sensor updates, fleet history, etc., to **mirror the life of its flying twin**. It is **ultra-realistic** and may consider one or more important and interdependent vehicle systems"*

NASA Modeling, Simulation, Information Tech. & Processing Roadmap, 2010

DTs & Models

Is a digital twin just another word for “model”?

DTs & Models

Is a digital twin just another word for “model”?

Is a digital twin just another word for “control system”?

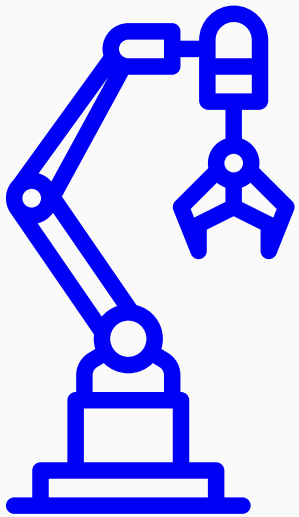
DTs & Models

Is a digital twin just another word for “model”?

Is a digital twin just another word for “control system”?

A digital twin integrates aspects of models and control systems

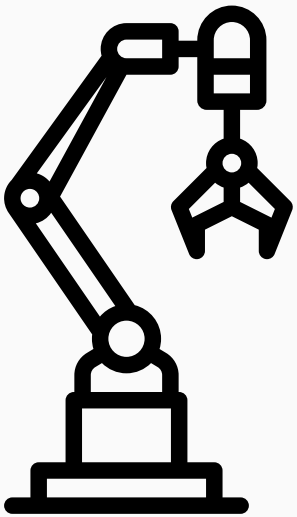
What is a Digital Twin?

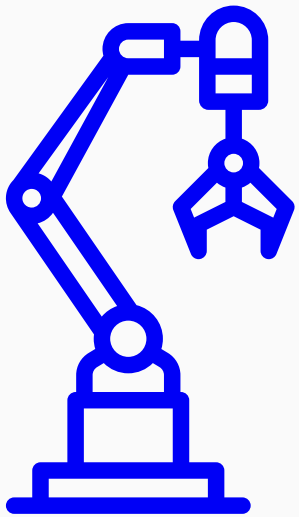


Model

What is a Digital Twin?

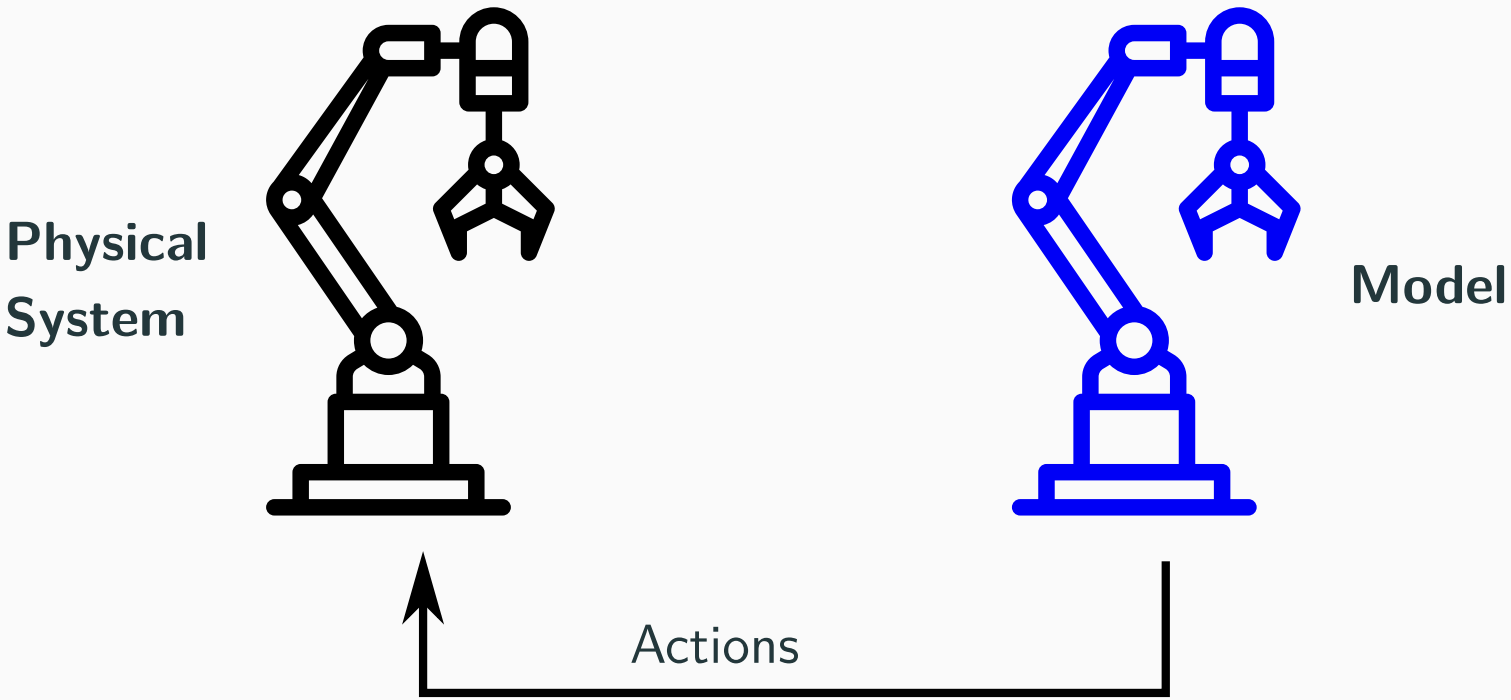
Physical System



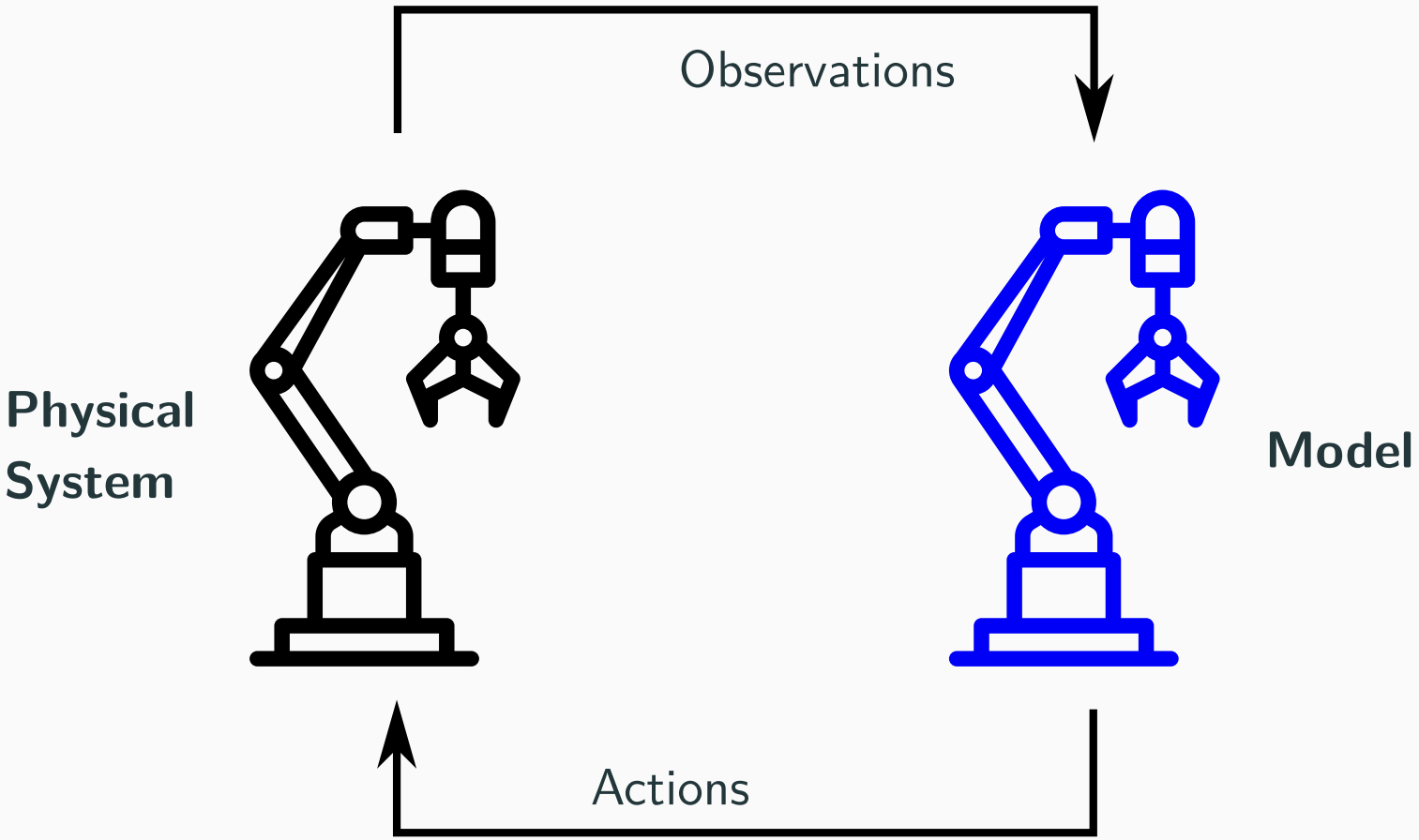


Model

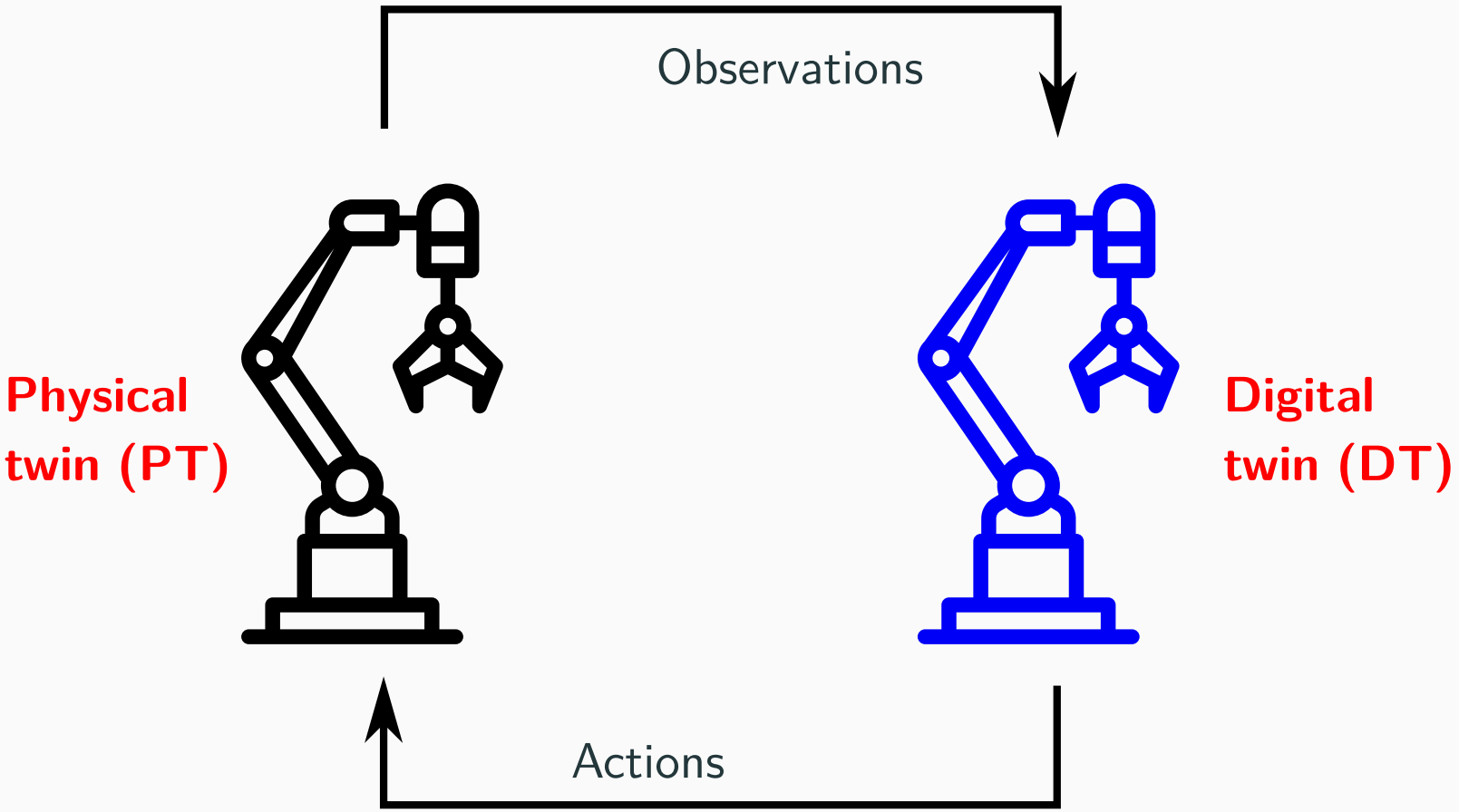
What is a Digital Twin?



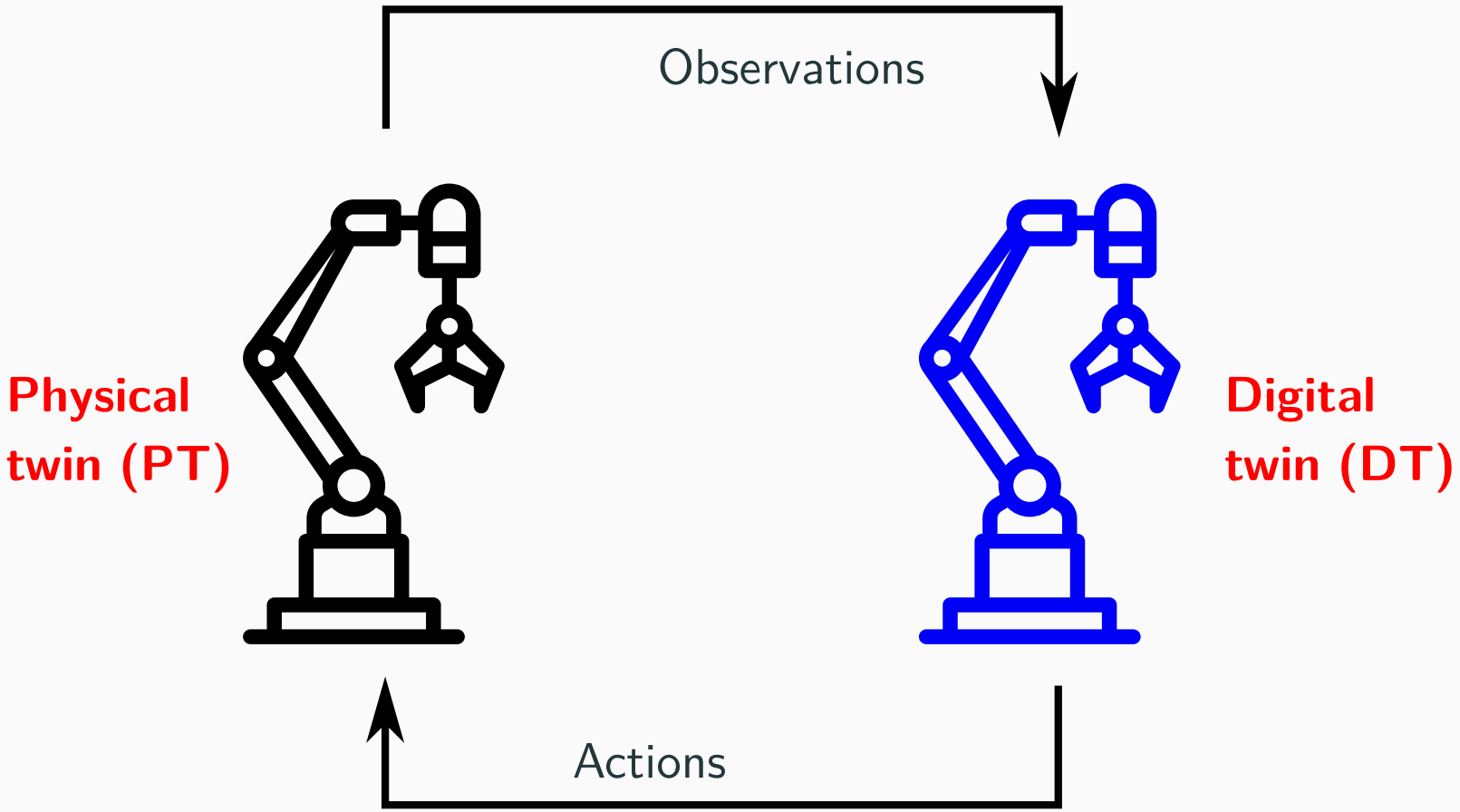
What is a Digital Twin?



What is a Digital Twin?



What is a Digital Twin?



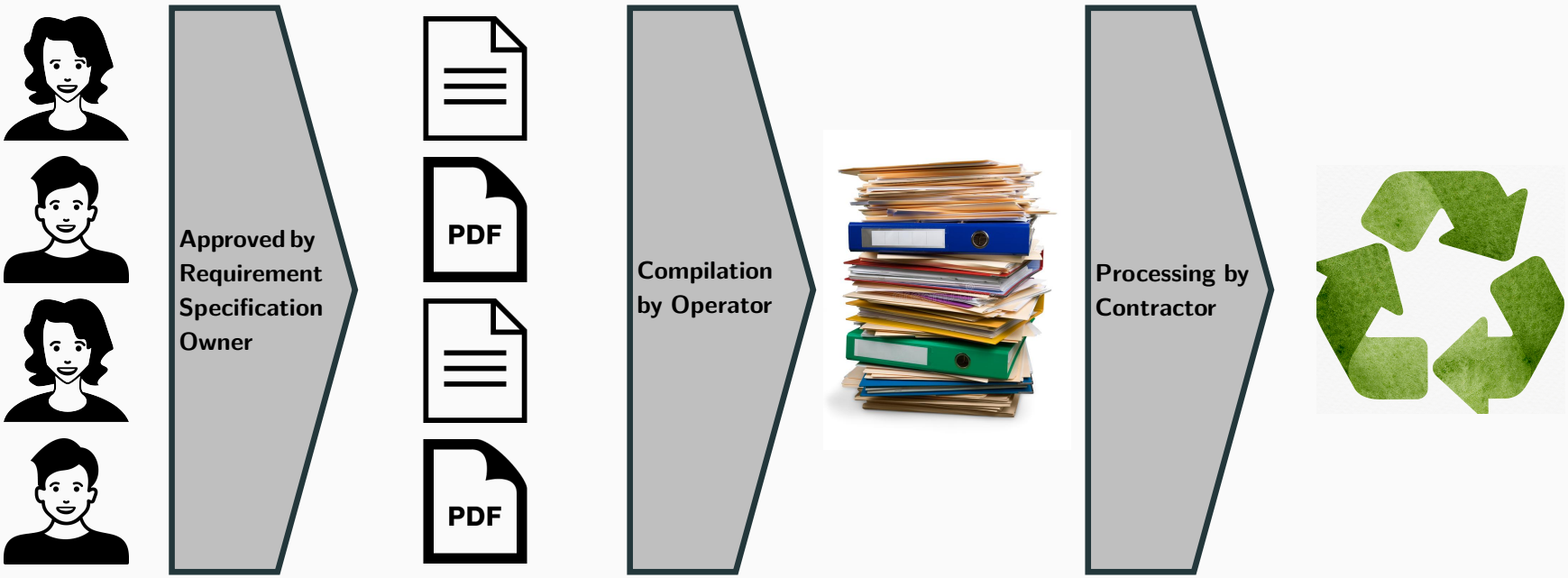
Digital Twin

1. DT is a live replica of a physical system (PT)
2. DT is connected to PT in near real-time via data streams

Lifecycle Management

Digital Thread: The Digital Twin Evolves in Tandem with the Asset

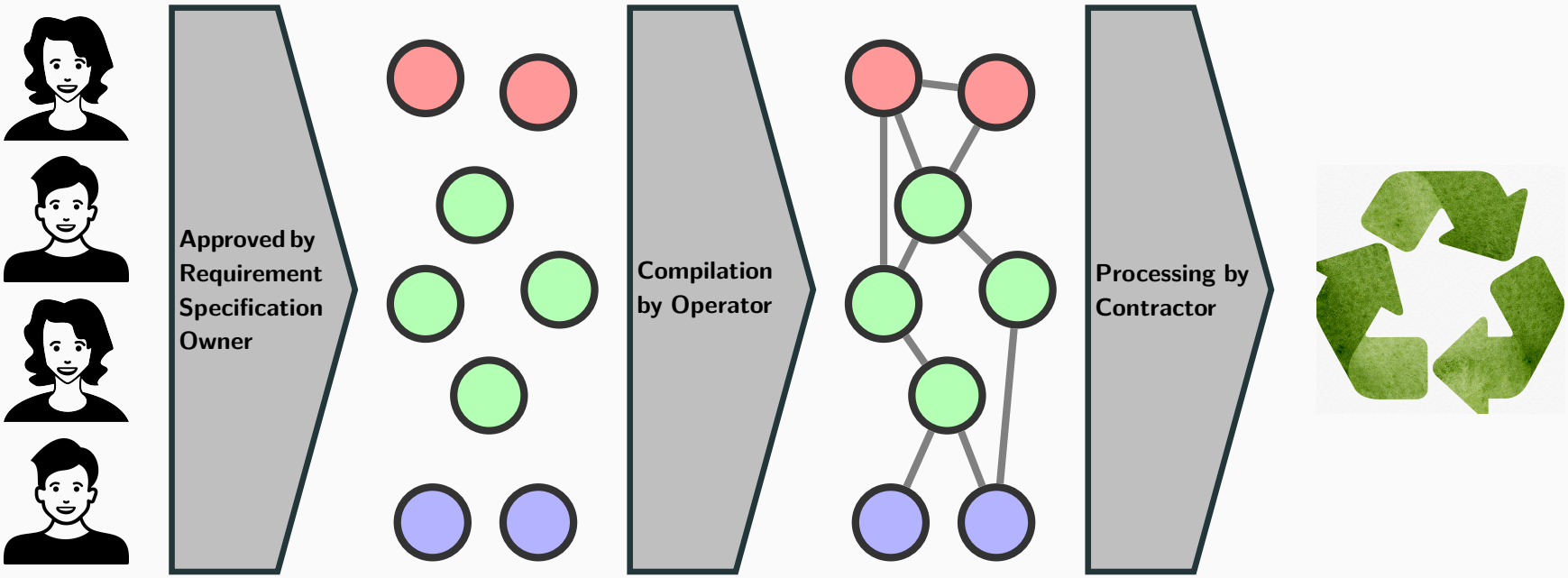
1. Connects the designs, requirements and software that go into the system represented by the DT
2. Connects the different phases of the system to the DT: design, development, operation, decommissioning, ...



Lifecycle Management

Digital Thread: The Digital Twin Evolves in Tandem with the Asset

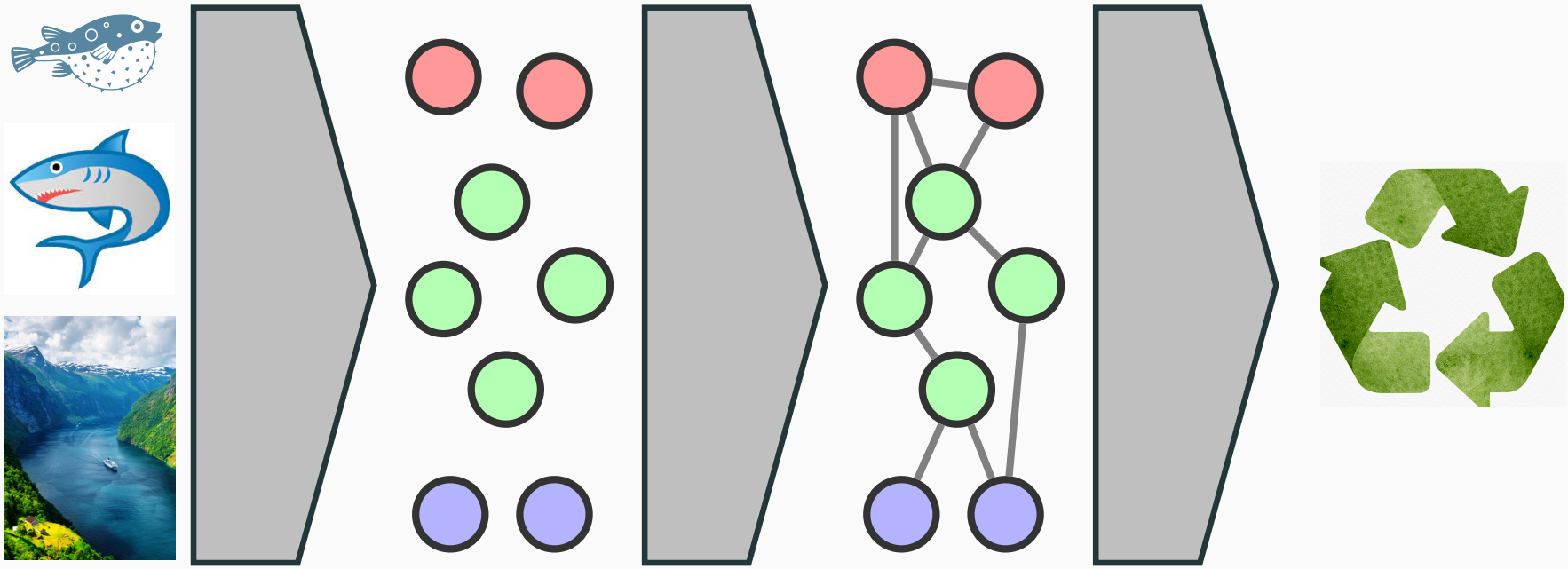
1. Connects the designs, requirements and software that go into the system represented by the DT
2. Connects the different phases of the system to the DT: design, development, operation, decommissioning, ...



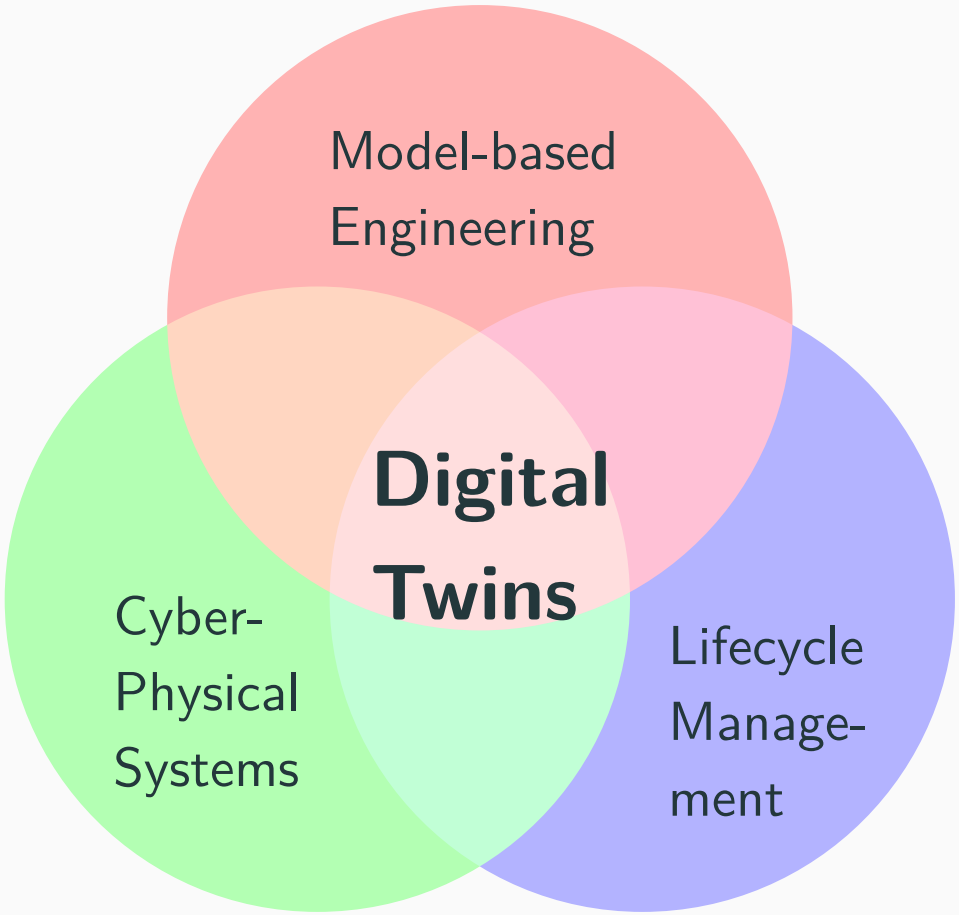
Lifecycle Management

Digital Thread: The Digital Twin Evolves in Tandem with the Asset

1. Connects the designs, requirements and software that go into the system represented by the DT
2. **What are the lifecycle events for operational systems?**



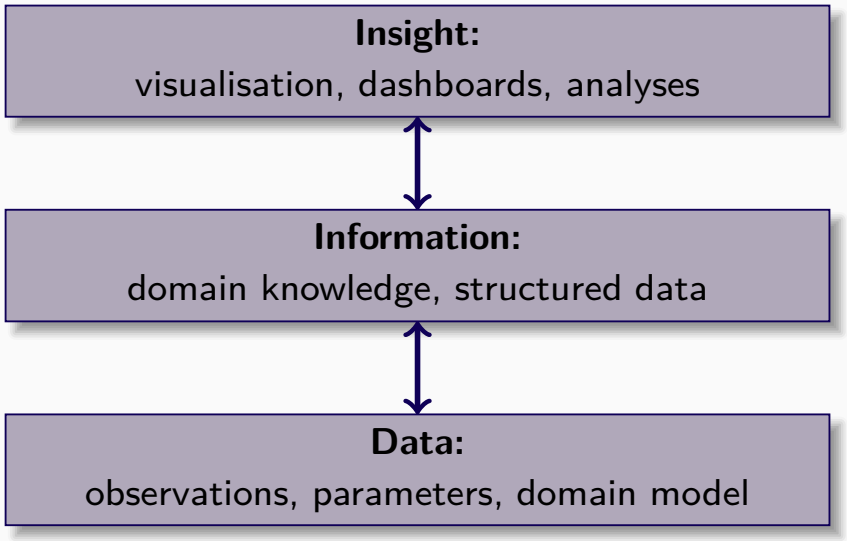
Digital Twins: A New Paradigm in SE?



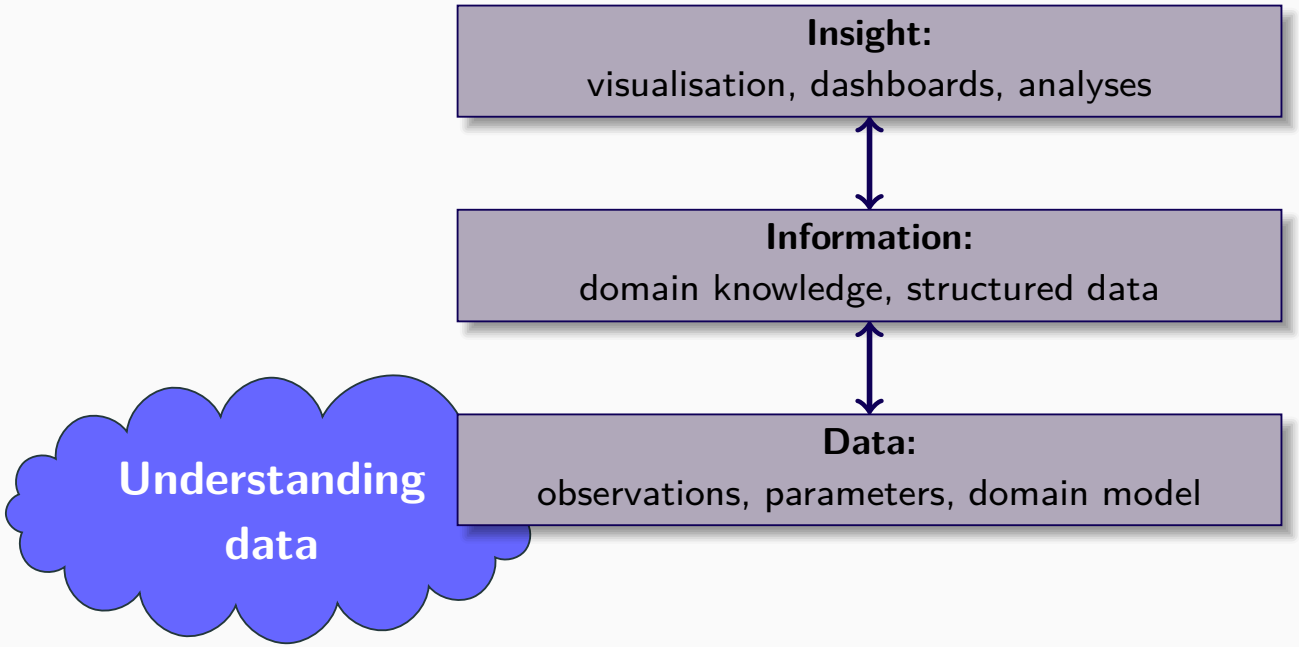
DTs: a new paradigm in SE

- Models go beyond the system design phase
- **Model-centric systems** the purpose is not models to build software, but software to maintain models
- **Model evolution:** reflect changes to the asset (automatically) throughout its lifetime
- **CPS in-the-large:** distributed, heterogeneous

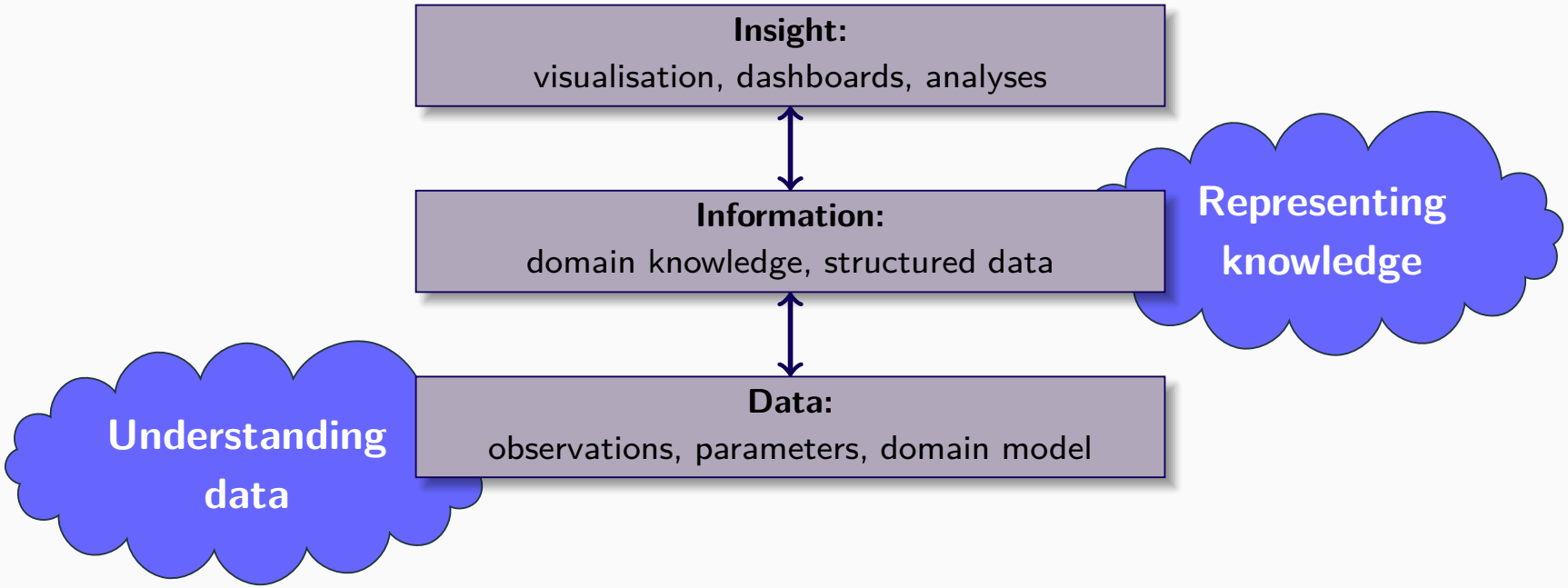
The Conceptual Layers of a Digital Twin



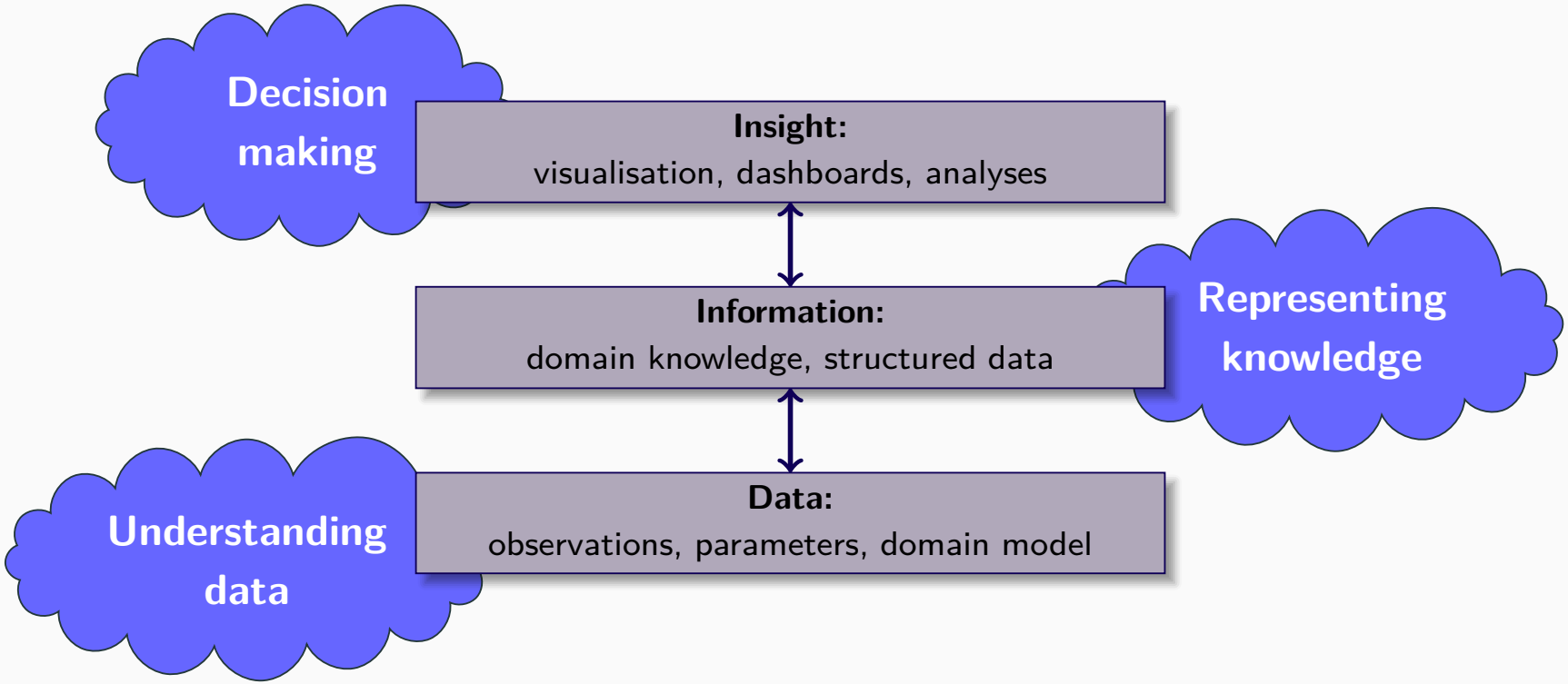
The Conceptual Layers of a Digital Twin



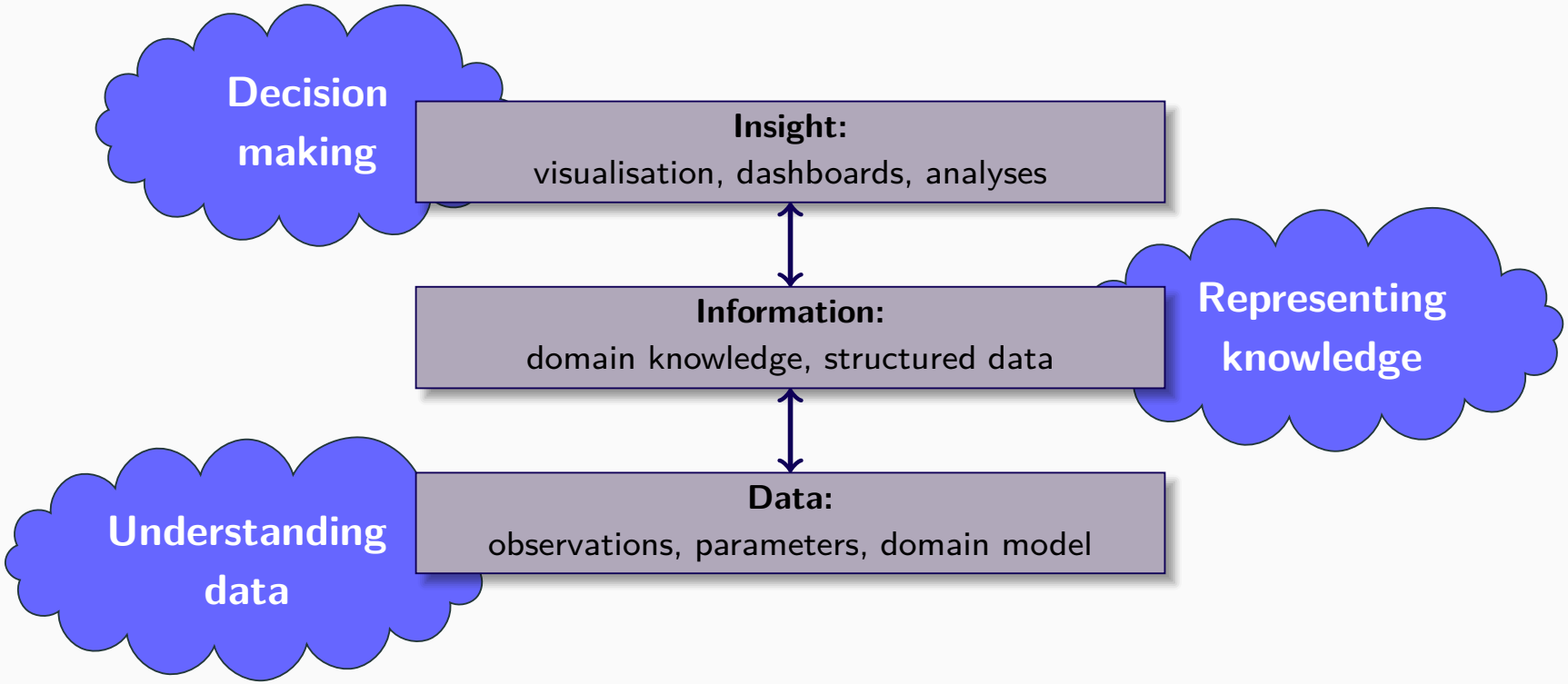
The Conceptual Layers of a Digital Twin



The Conceptual Layers of a Digital Twin

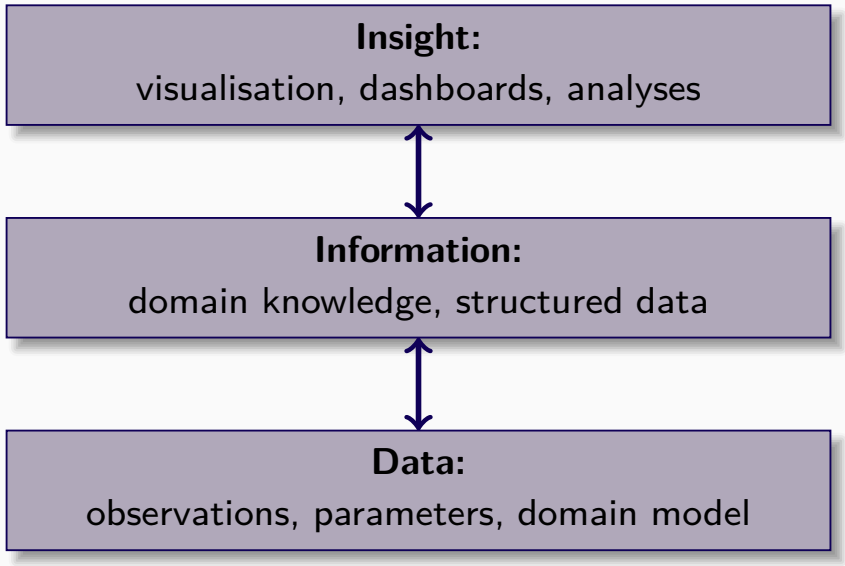


The Conceptual Layers of a Digital Twin



- **Descriptive:** Insight into the past ("what happened" scenarios)
- **Predictive:** Understanding the future ("what may happen" scenarios)
- **Prescriptive:** Advise on possible outcomes ("what if" scenarios)
- **Reactive:** Automated decision making

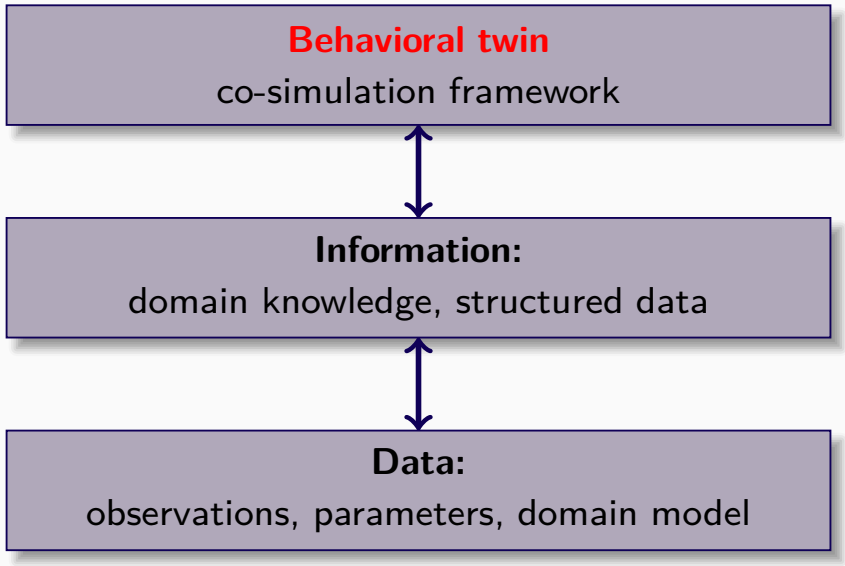
From Information to Insight (and Back Again)



Between information and insight

- Interesting to explore relations between the different layers: information and insight
- We are currently exploring connections between behavioral analyses (e.g., using simulators) and knowledge representation in the information layer

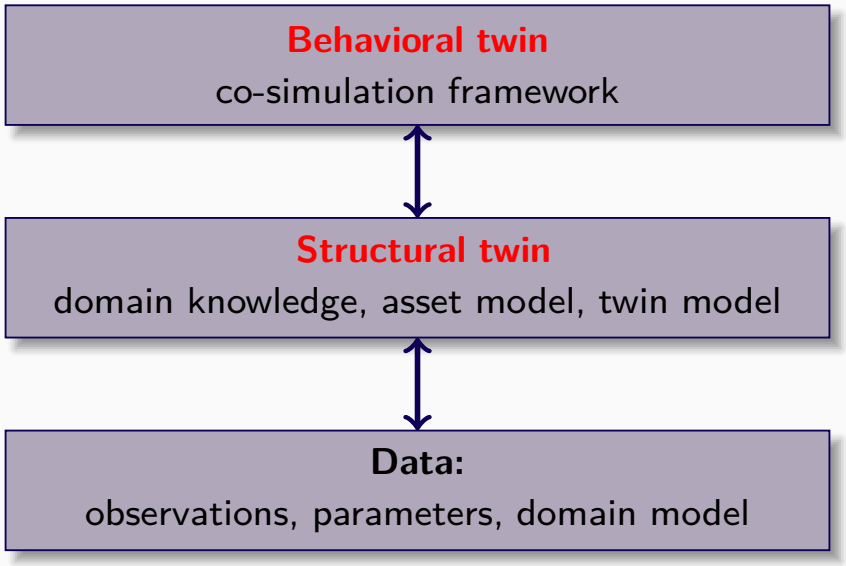
From Information to Insight (and Back Again)



Between information and insight

- Interesting to explore relations between the different layers: information and insight
- We are currently exploring connections between behavioral analyses (e.g., using simulators) and knowledge representation in the information layer

From Information to Insight (and Back Again)



Between information and insight

- Interesting to explore relations between the different layers: information and insight
- We are currently exploring connections between behavioral analyses (e.g., using simulators) and knowledge representation in the information layer

DTs & Formal Methods

What is the role of formal methods in digital twins?

DTs & Formal Methods

What is the role of formal methods in digital twins?

- Conceptual clearness, semantics, compositionality
- Correctness
- Better tool support
- Beyond simulation: worst-case, what-if scenarios, etc

DTs & Formal Methods

What is the role of formal methods in digital twins?

- Conceptual clearness, semantics, compositionality
- Correctness
- Better tool support
- Beyond simulation: worst-case, what-if scenarios, etc

What is the role of knowledge representation in digital twins?

DTs & Formal Methods

What is the role of formal methods in digital twins?

- Conceptual clearness, semantics, compositionality
- Correctness
- Better tool support
- Beyond simulation: worst-case, what-if scenarios, etc

What is the role of knowledge representation in digital twins?

- Structural twin: uniformly represent knowledge about PT and DT
- Reasoning support that can exploit this knowledge
- Allows correctness properties to be expressed as relations between DT and PT

Outline: The Semantically Reflected Digital Twin

In the following, we describe a digital twin architecture using formal methods based on three technologies/techniques.

SWT

Semantic Web Technologies for uniform knowledge representation and integration of domain knowledge (part II).

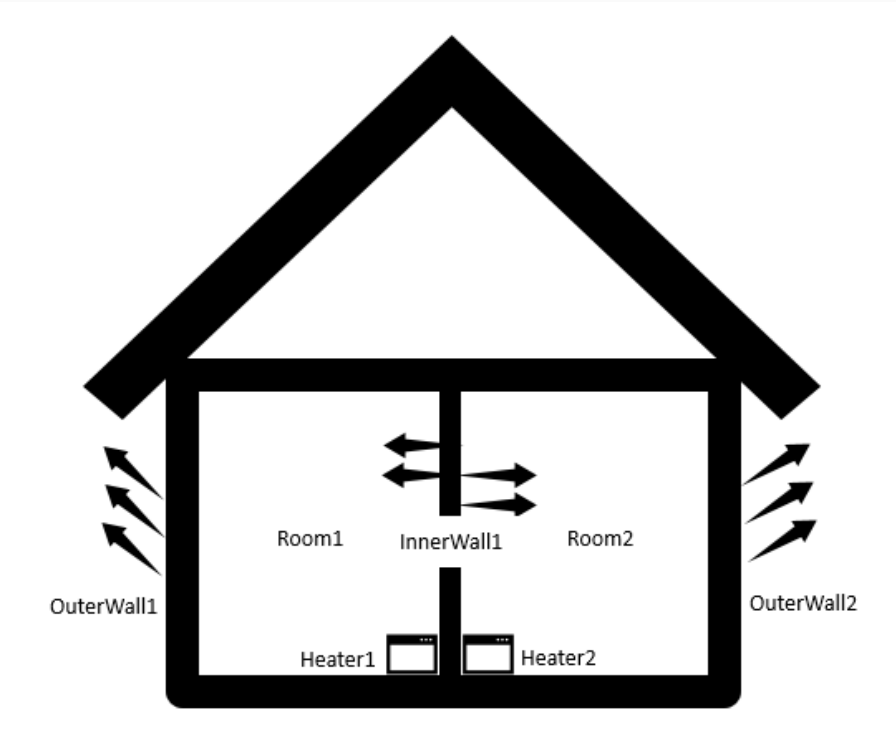
FMI

The Functional Mock-Up Interface standard for interfaces between PT and DT, as well as simulations (part III).

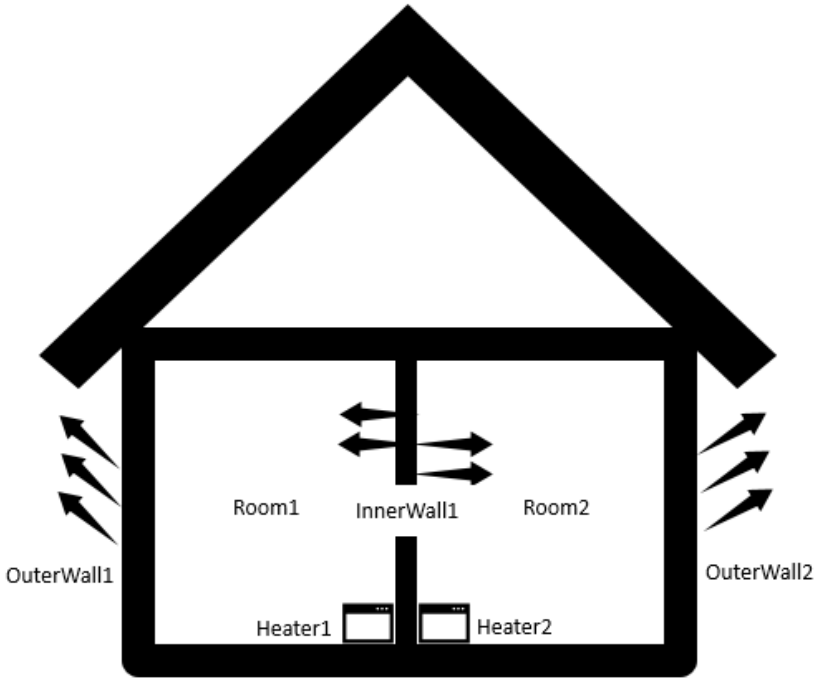
SMOL

Semantic Reflection to reason about PT and DT through the integration of SWT and FMI into a programming language (part IV).
The system is implemented in **SMOL**, a unique language designed specifically for integration of SWT and programming.

Example: House heating



Example: House heating



Structural twin

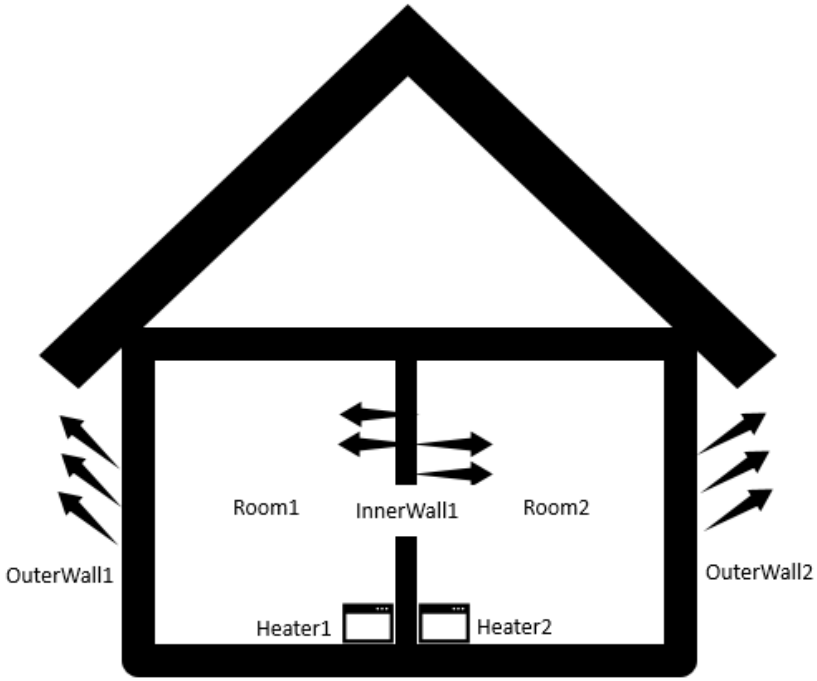
- **Asset model: Domain knowledge**

connects the rooms, heaters, walls into a "house", with corresponding simulators, etc

- **Asset model: Instance**

instance of the domain knowledge for a particular house

Example: House heating



Structural twin

- **Asset model: Domain knowledge**

connects the rooms, heaters, walls into a "house", with corresponding simulators, etc

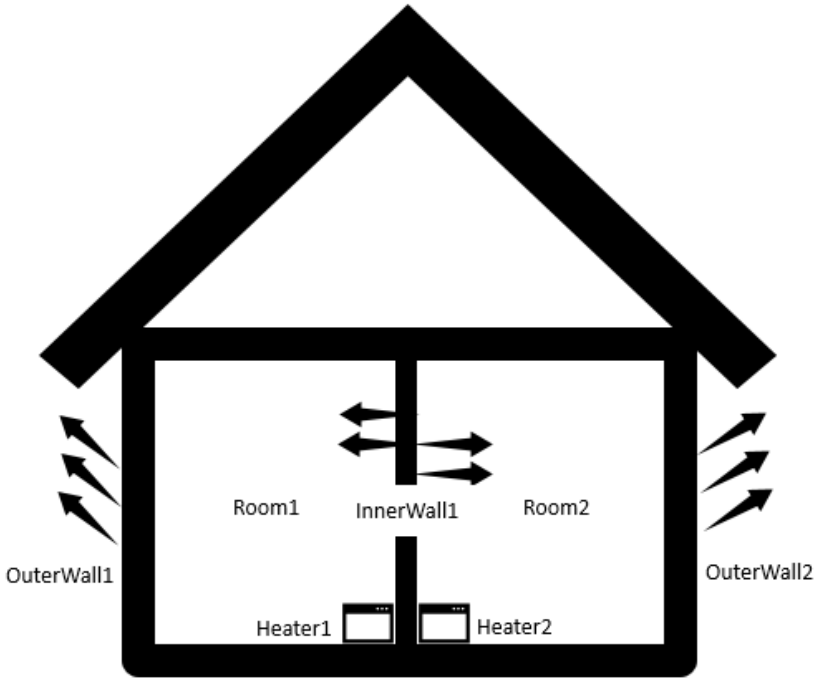
- **Asset model: Instance**

instance of the domain knowledge for a particular house

Behavioral twin

1. **Digital twin infrastructure:** coordinates simulation units
2. **Twin configuration:** coupled simulation units

Example: House heating



Structural twin

- **Asset model: Domain knowledge**
connects the rooms, heaters, walls into a "house", with corresponding simulators, etc
- **Asset model: Instance**
instance of the domain knowledge for a particular house
- **Twin model: Domain & Instance**
instance of the domain knowledge for the behavioral twin

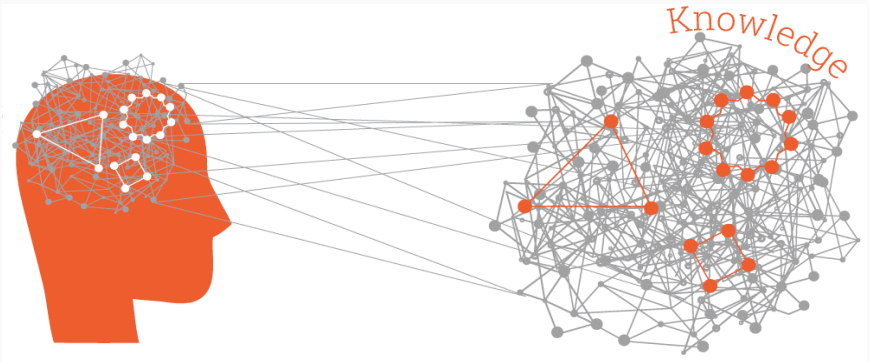
Behavioral twin

1. **Digital twin infrastructure:** coordinates simulation units
2. **Twin configuration:** coupled simulation units

Today:

- **Part I** Digital Twins Introduction:
  Concepts and Engineering Perspectives
- **Part II Modelling Knowledge using Semantic Technologies**
- **Part III** Semantically Reflected Digital Twins

Semantic Technologies



- Knowledge can be described ad hoc or in a structural manner
- Semantic Technologies facilitate the description of structured knowledge, consistency checking and reasoning
- W3C standards and well known technologies:
  - For data: RDF (Resource description framework)
  - For knowledge: OWL (Web Ontology language)
  - For queries: SPARQL(an RDF query language)

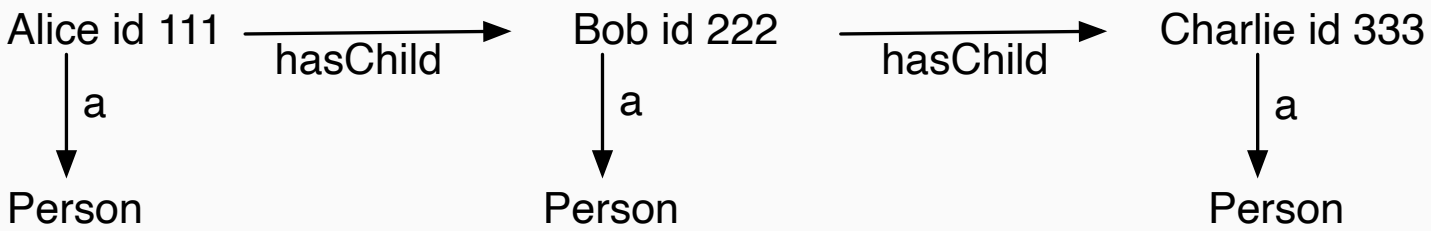
RDF (Resource description framework)

Data in **RDF** is expressed using a triple pattern, which consists of a *subject*, a *predicate*, and an *object*

RDF (Resource description framework)

Data in **RDF** is expressed using a triple pattern, which consists of a *subject*, a *predicate*, and an *object*

Example:



Here 'Alice' is subject, 'a' is predicate, 'Person' is object,
'Alice' is subject, 'id' is predicate, '111' is object, ...

OWL (Web Ontology language)

OWL: knowledge representation languages to build ontologies.

OWL (Web Ontology language)

OWL: knowledge representation languages to build ontologies.

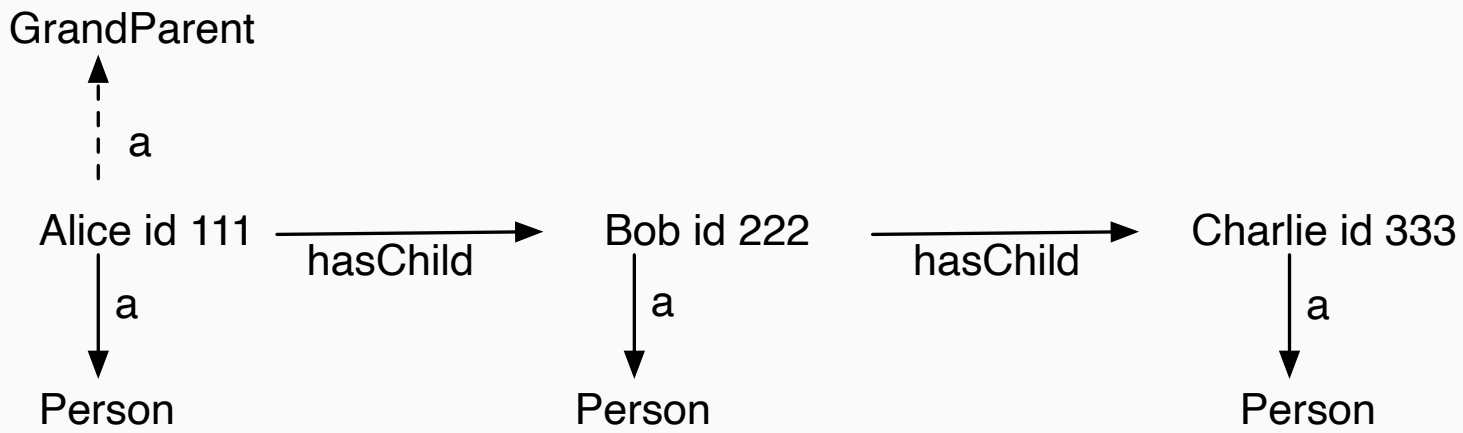
- Ontologies are logics for knowledge representation

OWL (Web Ontology language)

OWL: knowledge representation languages to build ontologies.

- Ontologies are logics for knowledge representation

Example:



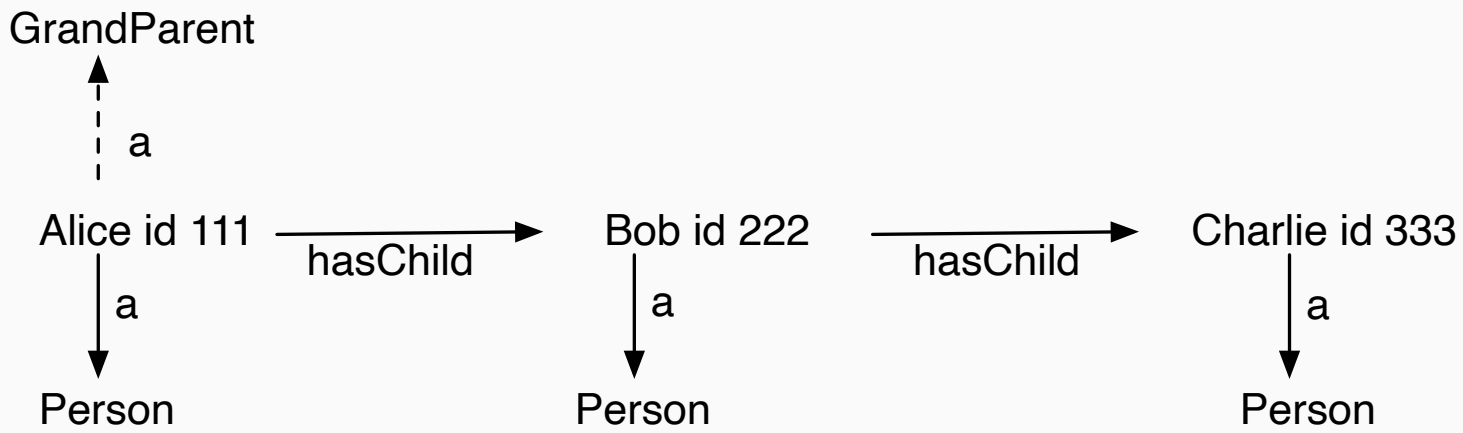
$$\forall x \exists y \exists z \cdot \mathit{hasChild}(x, y) \wedge \mathit{hasChild}(y, z) \wedge \mathit{Person}(z) \implies \mathit{GrandParent}(x)$$

OWL (Web Ontology language)

OWL: knowledge representation languages to build ontologies.

- Ontologies are logics for knowledge representation

Example:



$$\forall x \exists y \exists z \cdot hasChild(x, y) \land hasChild(y, z) \land Person(z) \implies GrandParent(x)$$

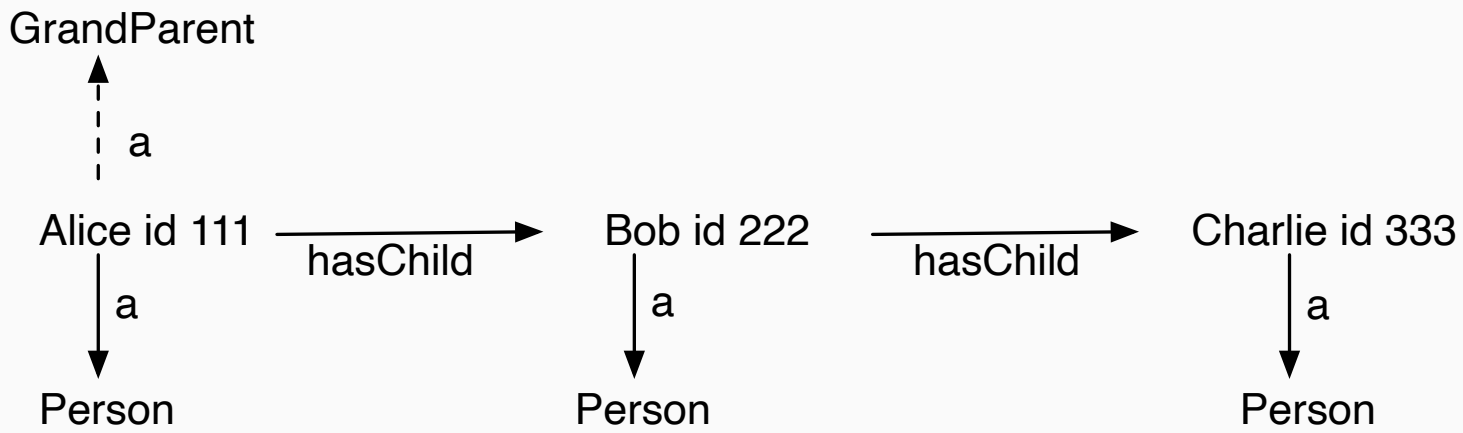
`hasChild` **some** (`hasChild` **some** `Person`) **subClassOf** `GrandParent`

OWL (Web Ontology language)

OWL: knowledge representation languages to build ontologies.

- Ontologies are logics for knowledge representation

  Example:



$$\forall x \exists y \exists z \cdot \mathit{hasChild}(x, y) \land \mathit{hasChild}(y, z) \land \mathit{Person}(z) \implies \mathit{GrandParent}(x)$$

`hasChild` **some** (`hasChild` **some** `Person`) **subClassOf** `GrandParent`

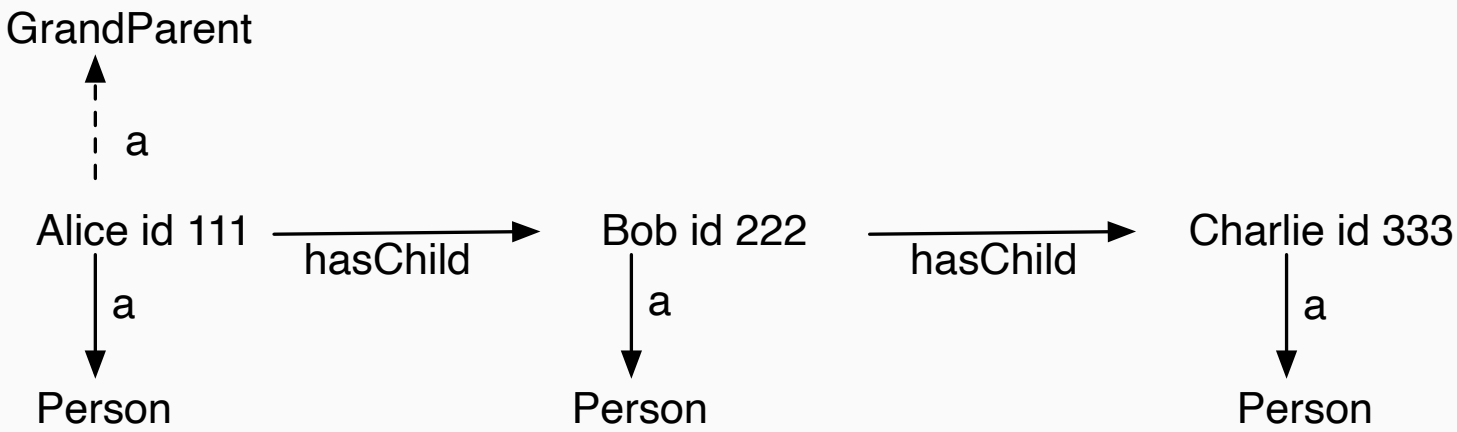
- Ontologies represent knowledge that is incremented over time

SPARQL

SPARQL is an RDF query language:
a query language for databases stored in RDF format

SPARQL

SPARQL is an RDF query language:
a query language for databases stored in RDF format

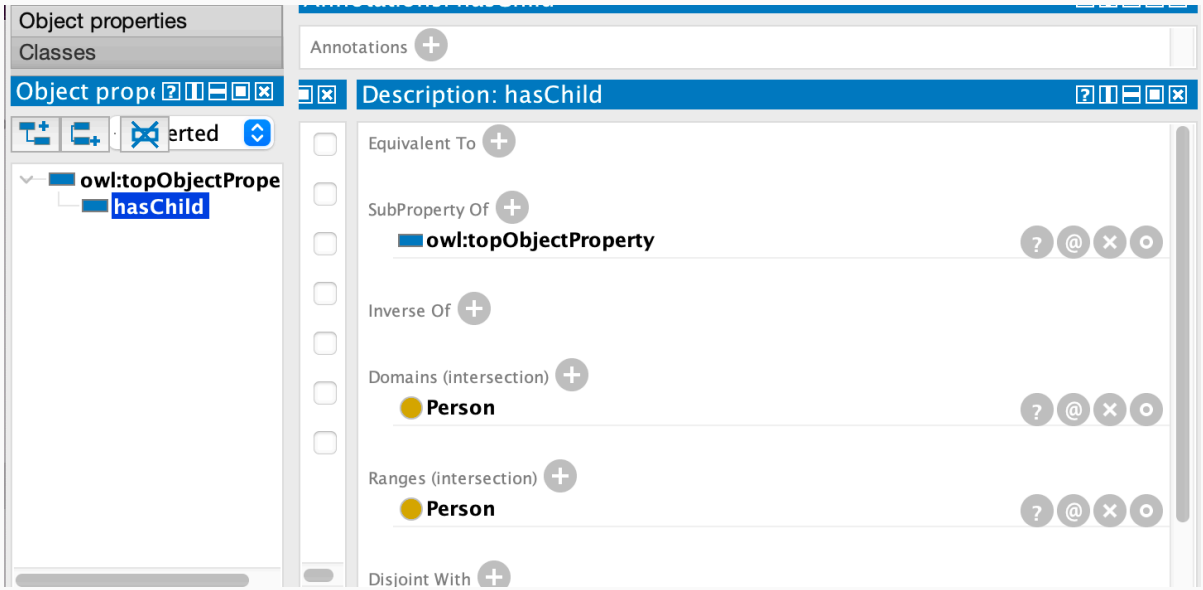


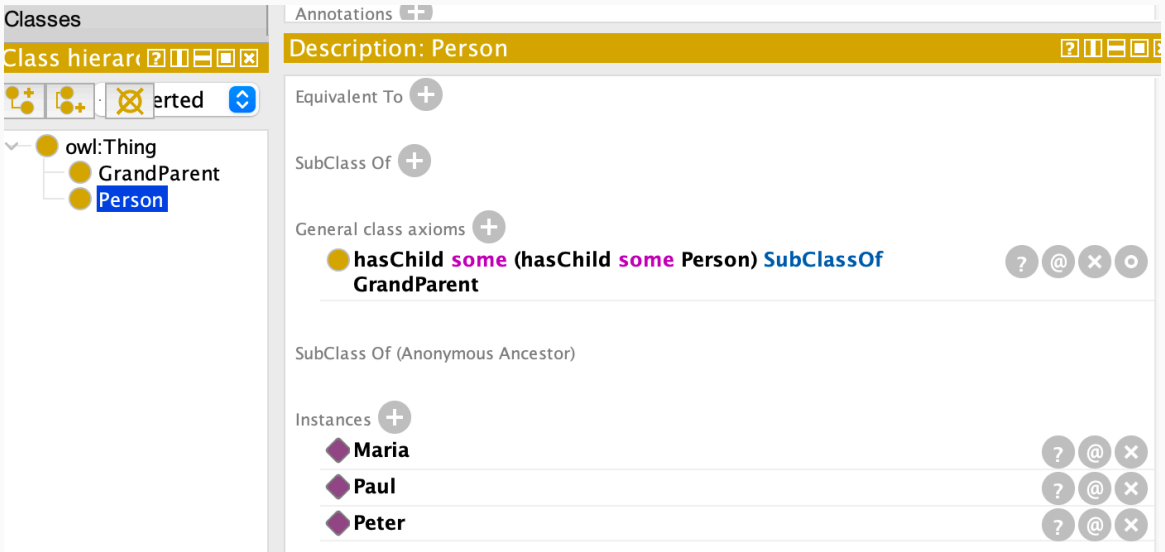

```
SELECT ?x WHERE { ?x a :Person }

SELECT ?x ?y WHERE { ?x a :Person. ?x :hasChild ?y }

SELECT ?x WHERE { ?x a :GrandParent }
```

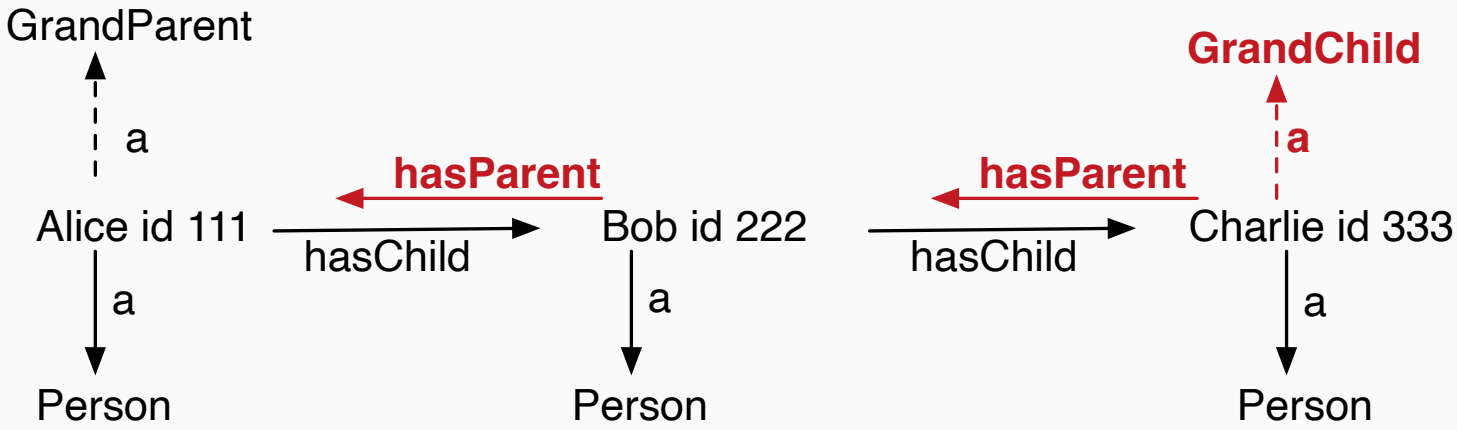
Example in Protégé





Exercise

Add the GrandChild Class and the hasParent property.



Hint:

$$\forall x \exists y \exists z \cdot hasParent(x, y) \wedge hasParent(y, z) \wedge Person(z) \implies GrandChild(x)$$

Download the example from: `https://github.com/smolang/SemanticObjects/blob/master/examples/tutorialfiles.zip`

File: example1a.ttl

Asset modelling

Asset model in the engineering domain

An asset model is an organized, digital description of the composition and properties of an asset

Asset modelling

Asset model in the engineering domain

An asset model is an organized, digital description of the composition and properties of an asset

- In the engineering domain it is common practice to build asset models to support, e.g., maintenance, operations, design etc.
- There are currently several industry initiatives that endorse the use of ontologies for asset modelling, e.g., in the Industry 4.0

Asset modelling

Asset model in the engineering domain

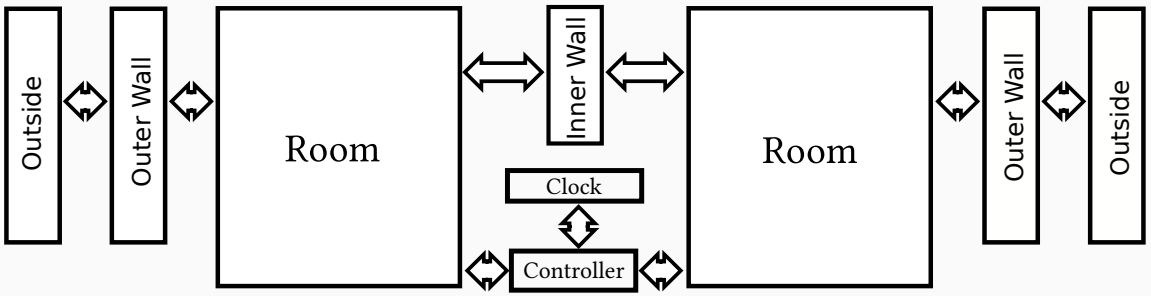
An asset model is an organized, digital description of the composition and properties of an asset

- In the engineering domain it is common practice to build asset models to support, e.g., maintenance, operations, design etc.
- There are currently several industry initiatives that endorse the use of ontologies for asset modelling, e.g., in the Industry 4.0

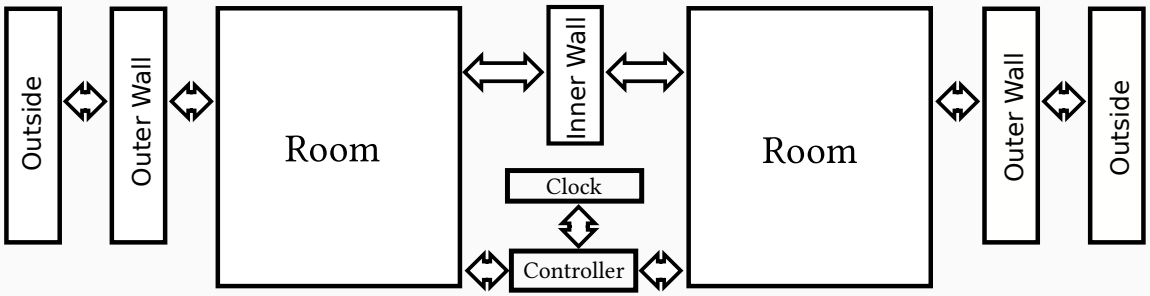
Asset models & digital twins

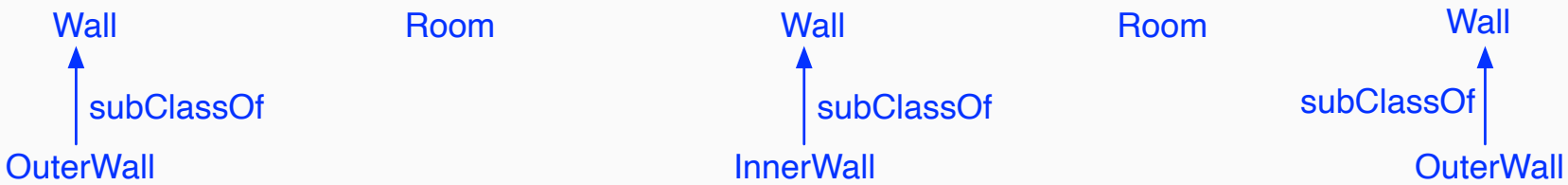
Assets models are any object of interest in a digital twin. They provide the twin with knowledge about the static structure that can be used for the twin's simulation model

The House Asset Use Case

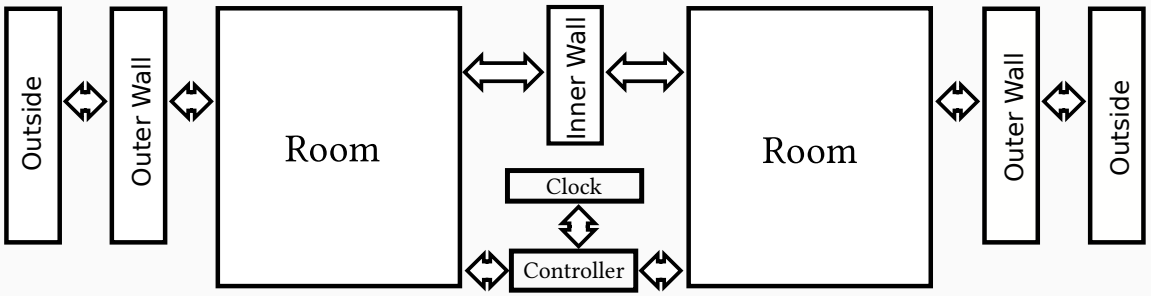


The House Asset Use Case





The House Asset Use Case



w1 id 21

Wall

subClassOf

OuterWall

r1 id 13

Room

w2 id 22

Wall

subClassOf

InnerWall

r2 id 12

Room

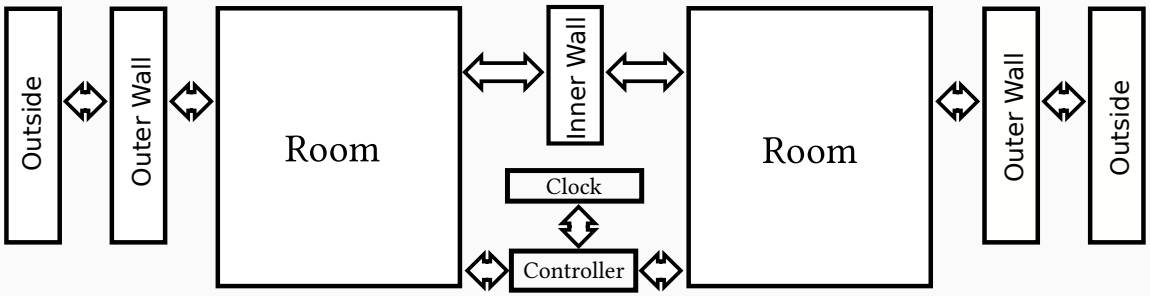
w3 id 23

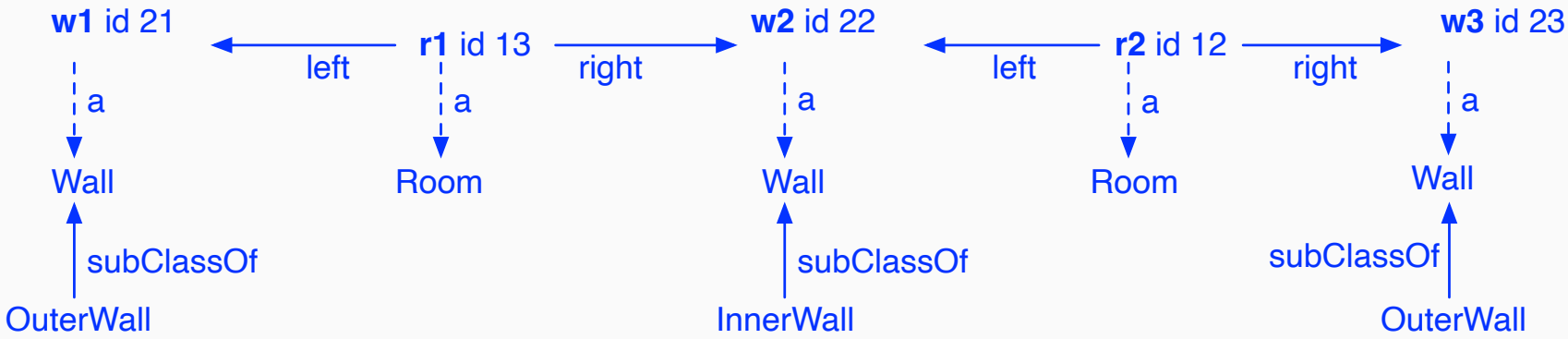
Wall

subClassOf

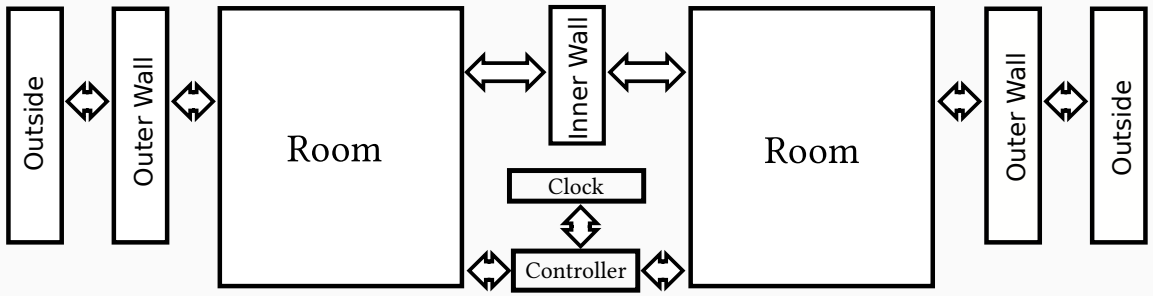
OuterWall

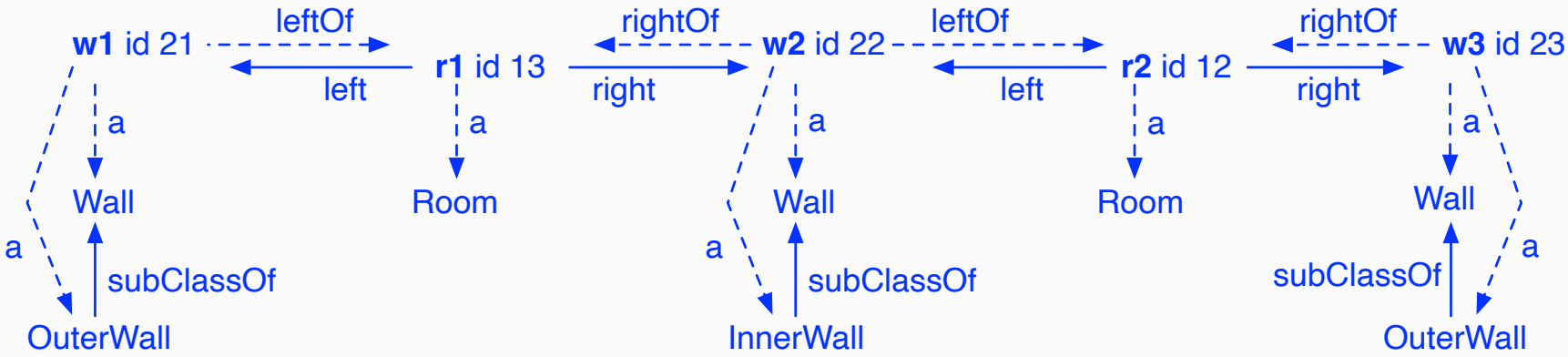
The House Asset Use Case



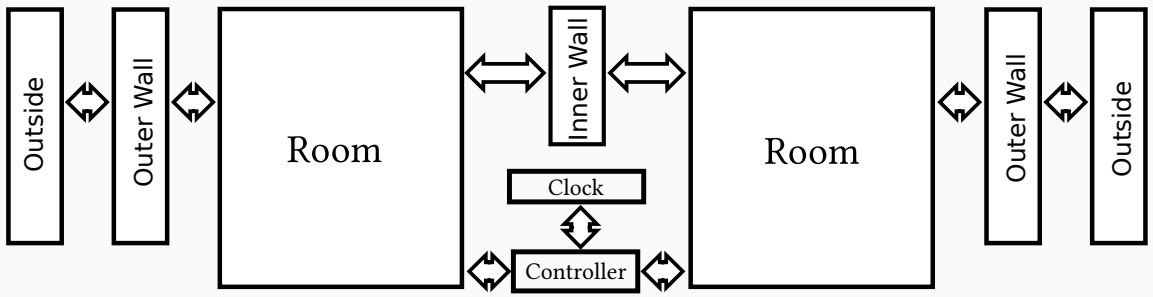


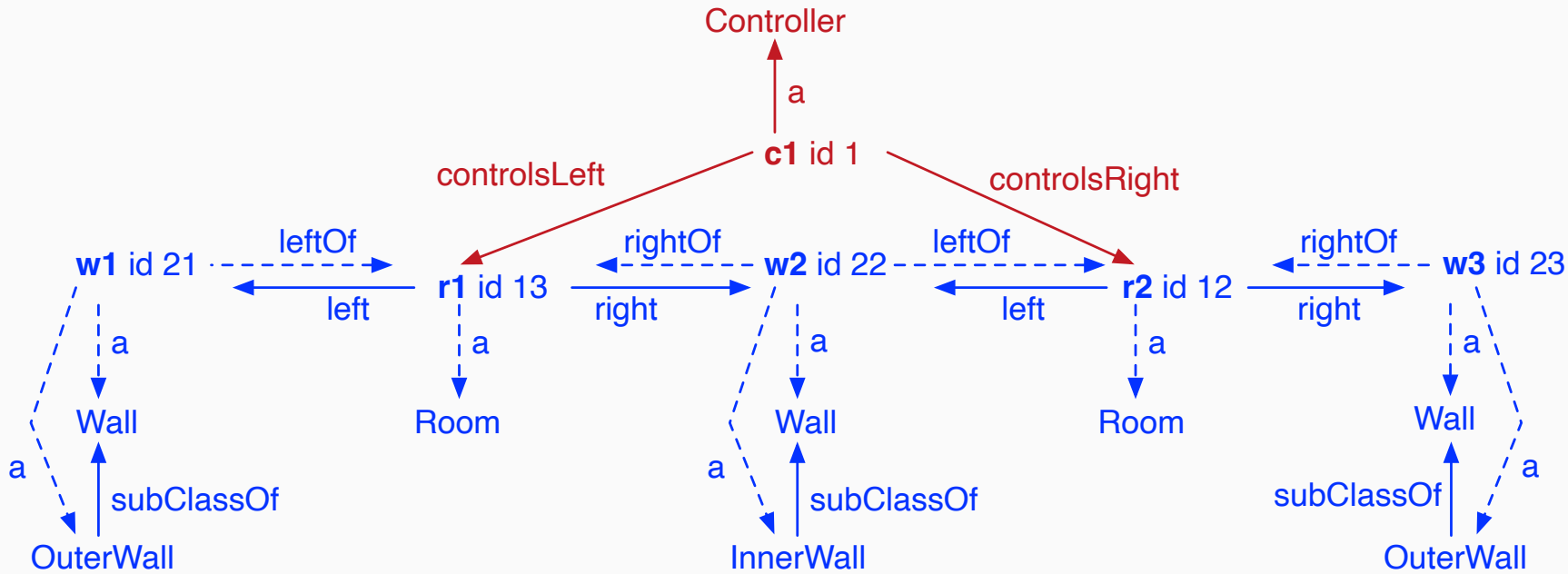
The House Asset Use Case





Exercise: The House Asset Use Case

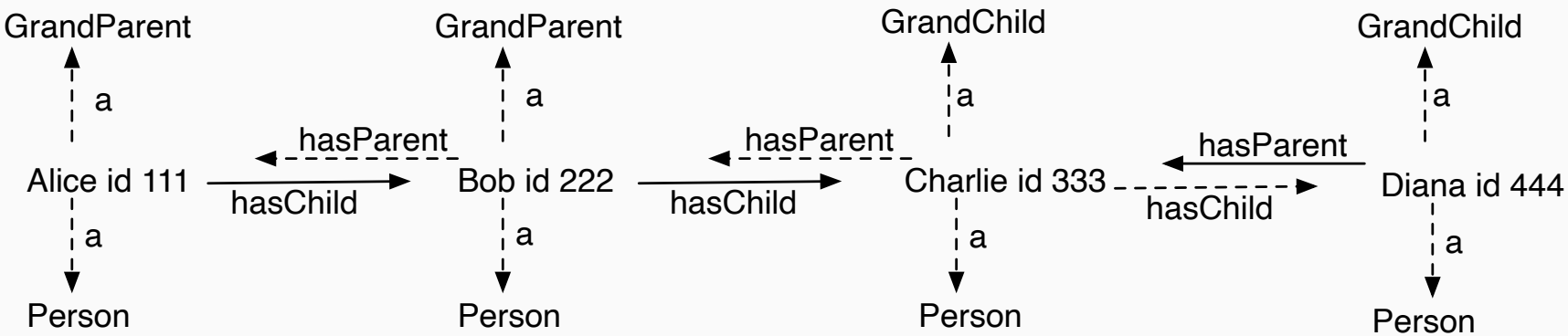




Download the house asset model from: `https://github.com/smolang/SemanticObjects/blob/master/examples/tutorialfiles.zip`

File: house.ttl

Homework: The Family Tree



SPARQL in SMOL

```
main
    List<Int> results =  access(
                    "SELECT ?obj {?a a asset:Room.
                                  ?a asset:id ?obj}");
    while results != null do
        Int current = results.content;
        results = results.next;
        print(current);
    end
end
```

Yesterday:

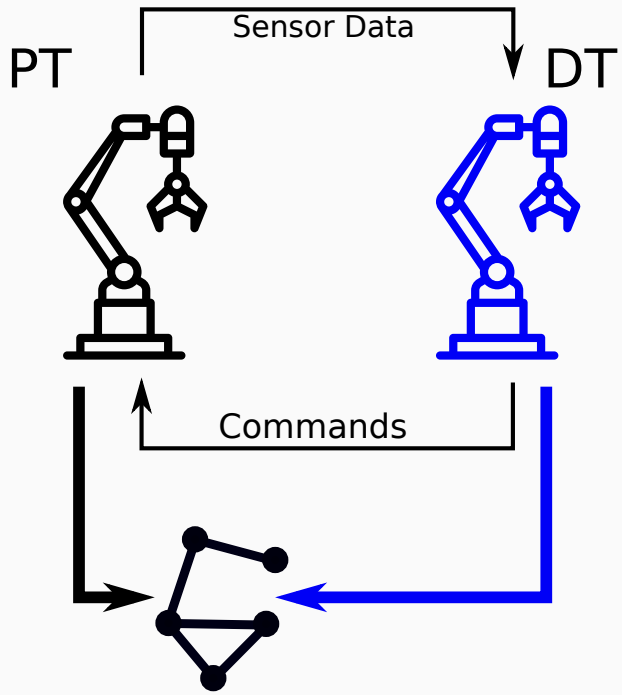
- **Part I** Digital Twins Introduction: Concepts and Engineering Perspectives
- **Part II** Modelling Knowledge using Semantic Technologies
- **Part III Semantically Reflected Digital Twins**

Structural Self-Adaptation

- We can access the sensors of the physical system (FMI),
- access the structure of the physical system (SWT), and
- simulate the digital design (FMI).

Structural Self-Adaptation

- We can access the sensors of the physical system (FMI),
- access the structure of the physical system (SWT), and
- simulate the digital design (FMI).

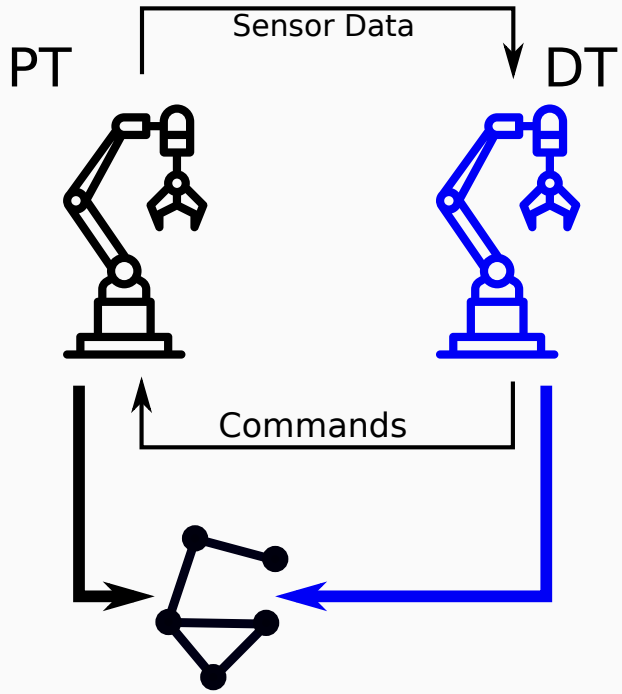


Structural Self-Adaptation

- We can access the sensors of the physical system (FMI),
- access the structure of the physical system (SWT), and
- simulate the digital design (FMI).

Putting it all together

- Compare simulations to sensors
- Compare digital with physical structure
- Self-adapt to changes in physical system

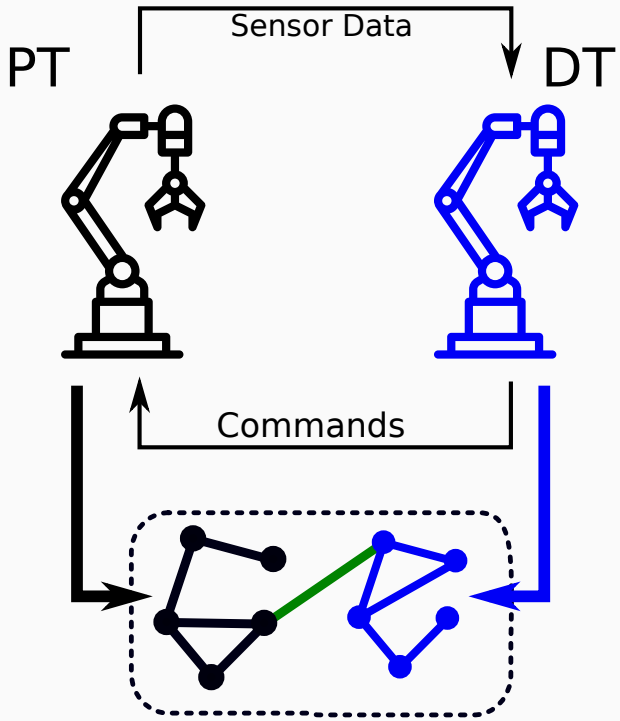


Structural Self-Adaptation

- We can access the sensors of the physical system (FMI),
- access the structure of the physical system (SWT), and
- simulate the digital design (FMI).

Putting it all together

- Compare simulations to sensors
- Compare digital with physical structure
  What is the digital structure?
- Self-adapt to changes in physical system

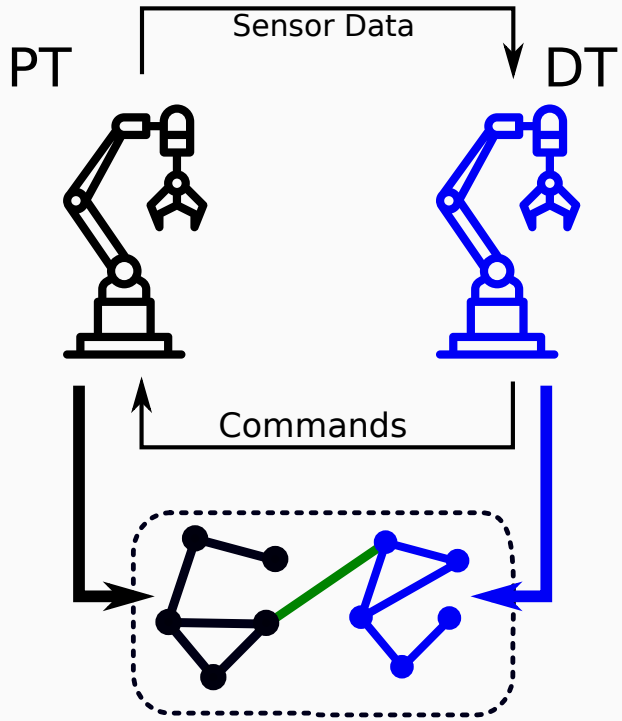


Structural Self-Adaptation

- We can access the sensors of the physical system (FMI),
- access the structure of the physical system (SWT), and
- simulate the digital design (FMI).

Putting it all together

- Compare simulations to sensors
- Compare digital with physical structure
  What is the digital structure?
- Self-adapt to changes in physical system
  Self-adaptation?



Self-Adaptation (I)

Digital Twins: Self-Adaptation

Self-adaptation means to *automatically* reestablish some property of a system, by reacting to outside stimuli. For Digital Twins, the "outside" is the physical system.

Two kinds of self-adaptation to reestablish the *twinning* property:

- Behavioral self-adaptation if sensors and simulators mismatch
- Structural self-adaptation if *structures* mismatch

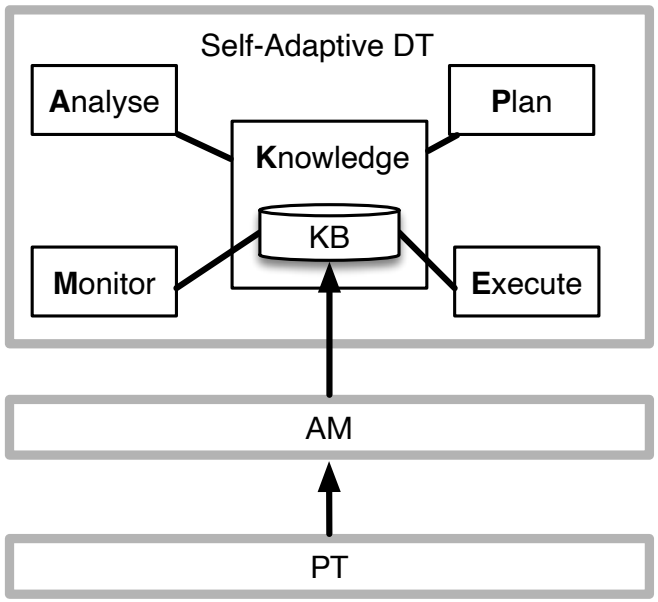
MAPE-K

MAPE-K is an established conceptual framework to structure self-adaptive systems.

MAPE-K

MAPE-K is an established conceptual framework to structure self-adaptive systems.

- A **K**nowledge component keeps track of information and goals for the self-adaptation loop:
- **M**onitor the situation
- **A**nalyze whether the situation requires adaptation
- **P**lan the adaptation
- **E**xecute the plan



Self-Adaptation (II)

Behavioral Self-Adaptation

Simulated (=expected) behavior of certain components does not match the real (=measured) behavior of the sensors.

- Monitor sensors
- Analyze the relation to simulation
- Plan repair by, e.g., finding new simulation parameters
- Exchange simulators or send signal to physical system

Reasons

- Sensor drift
- Modeling errors
- Faults
- Unexpected events

Self-Adaptation (III)

Structural Self-Adaptation

Simulated structure of digital system does not match real (= expressed in asset model) structure.

Self-Adaptation (III)

Structural Self-Adaptation

Simulated structure of digital system does not match real (= expressed in asset model) structure.

Semantically Lifted Programs

We need to express the program structure, so we can *uniformly* access it together with the asset model. How to apply semantic web technologies on programs? $\Rightarrow$ Semantical lifting.

Self-Adaptation (III)

Structural Self-Adaptation

Simulated (= lifted) structure of digital system does not match real (= expressed in asset model) structure.

Semantically Lifted Programs

We need to express the program structure, so we can *uniformly* access it together with the asset model. How to apply semantic web technologies on programs? $\Rightarrow$ Semantical lifting.

Semantical lifting is a mechanism to automatically generate the knowledge graph of a program state.

Self-Adaptation

Repair

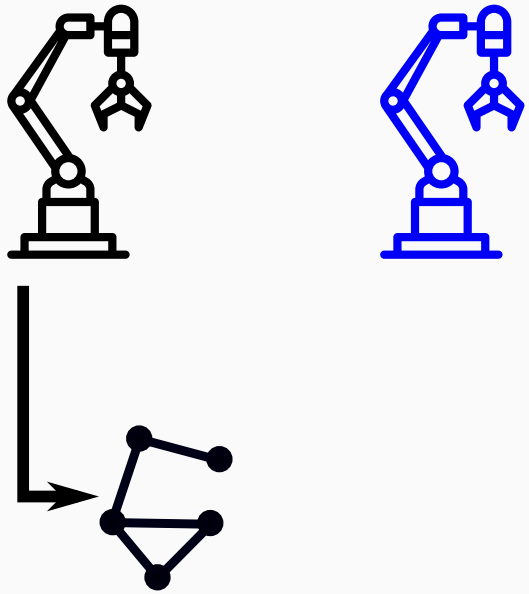
To self-adapt we must (1) detect broken twinning and (2) repair it.

Self-Adaptation

Repair

To self-adapt we must (1) detect broken twinning and (2) repair it.

- Access PT structure through asset model

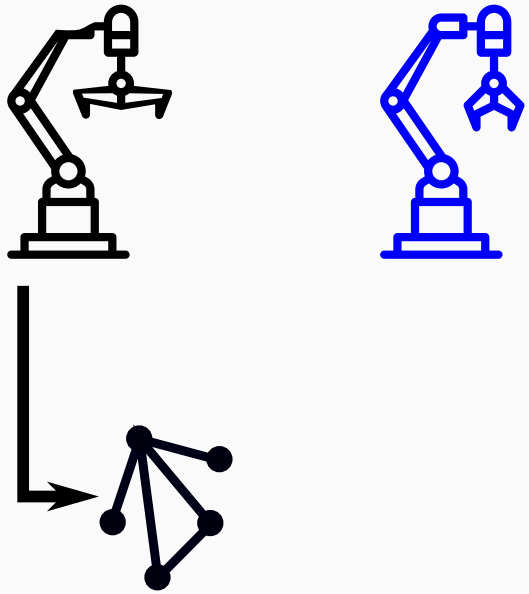


Self-Adaptation

Repair

To self-adapt we must (1) detect broken twinning and (2) repair it.

- Access PT structure through asset model
- Changes of PT are visible in asset model

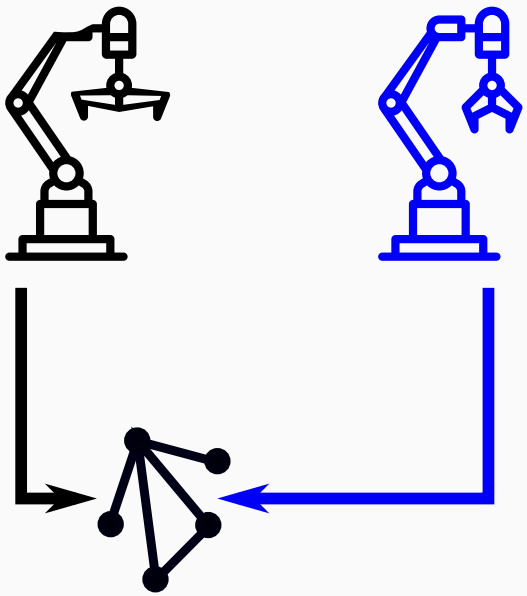


Self-Adaptation

Repair

To self-adapt we must (1) detect broken twinning and (2) repair it.

- Access PT structure through asset model
- Changes of PT are visible in asset model
- Asset model accessible directly to DT

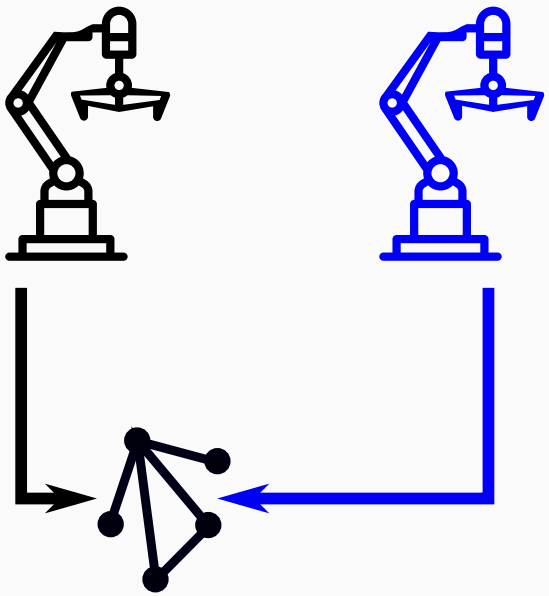


Self-Adaptation

Repair

To self-adapt we must (1) detect broken twinning and (2) repair it.

- Access PT structure through asset model
- Changes of PT are visible in asset model
- Asset model accessible directly to DT

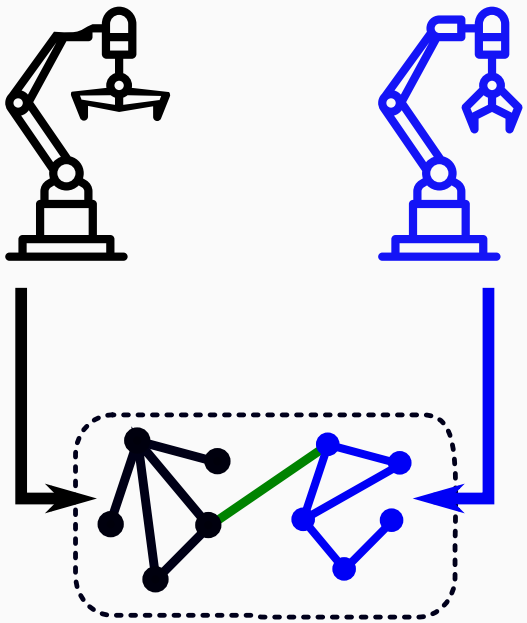


Self-Adaptation

Repair

To self-adapt we must (1) detect broken twinning and (2) repair it.

- Access PT structure through asset model
- Changes of PT are visible in asset model
- Asset model accessible directly to DT
- Detect changes through combined knowledge graph

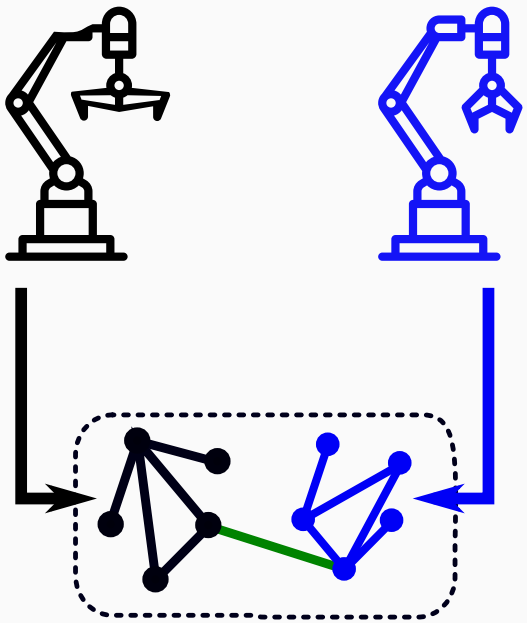


Self-Adaptation

Repair

To self-adapt we must (1) detect broken twinning and (2) repair it.

- Access PT structure through asset model
- Changes of PT are visible in asset model
- Asset model accessible directly to DT
- Detect changes through combined knowledge graph

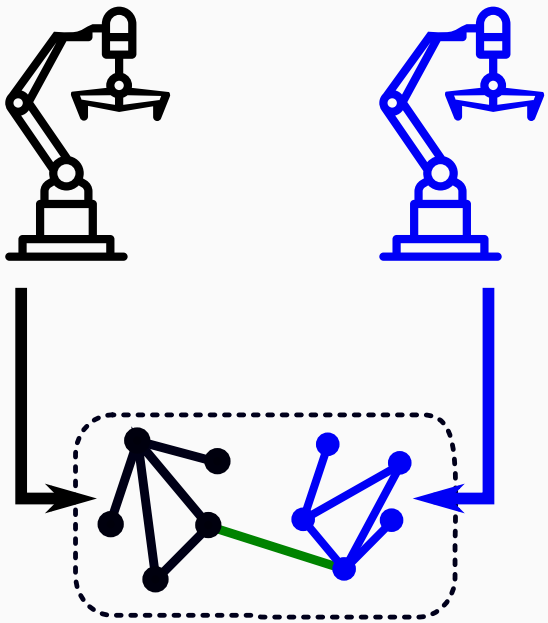


Self-Adaptation

Repair

To self-adapt we must (1) detect broken twinning and (2) repair it.

- Access PT structure through asset model
- Changes of PT are visible in asset model
- Asset model accessible directly to DT
- Detect changes through combined knowledge graph
- Information for repair available there!

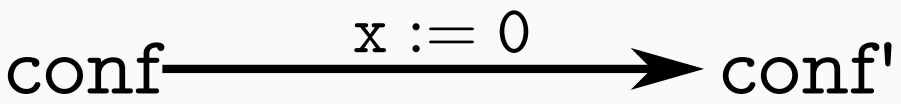


Semantically Lifted States

A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.

Semantically Lifted States

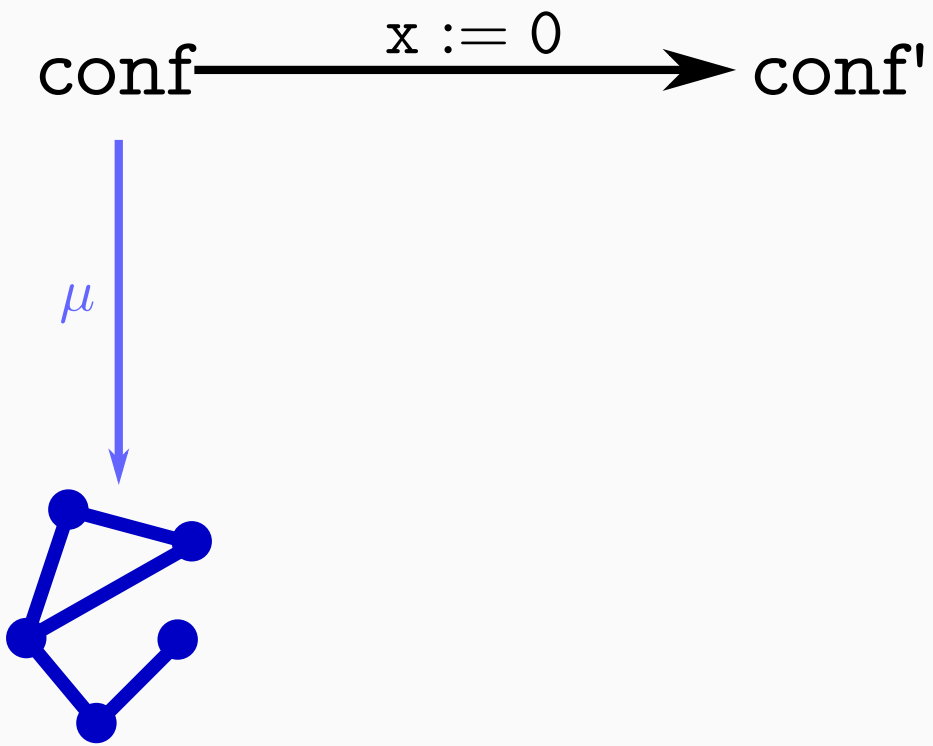
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



Programming and Debugging with Semantically Lifted States, Kamburjan et al. [ESWC'21]

Semantically Lifted States

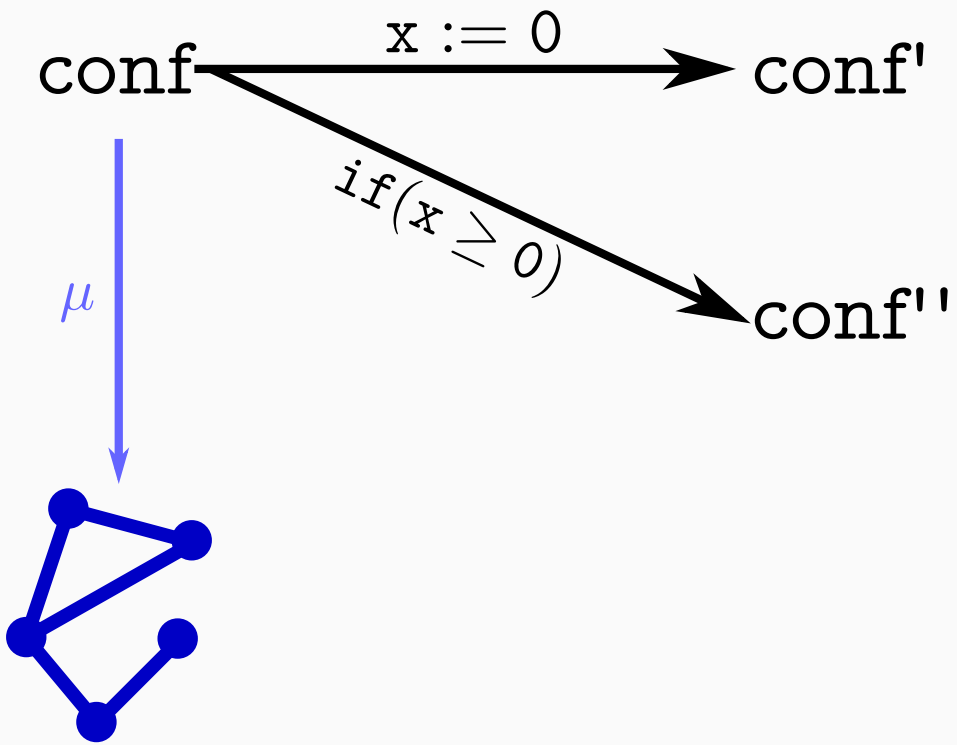
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



Programming and Debugging with Semantically Lifted States, Kamburjan et al. [ESWC'21]

Semantically Lifted States

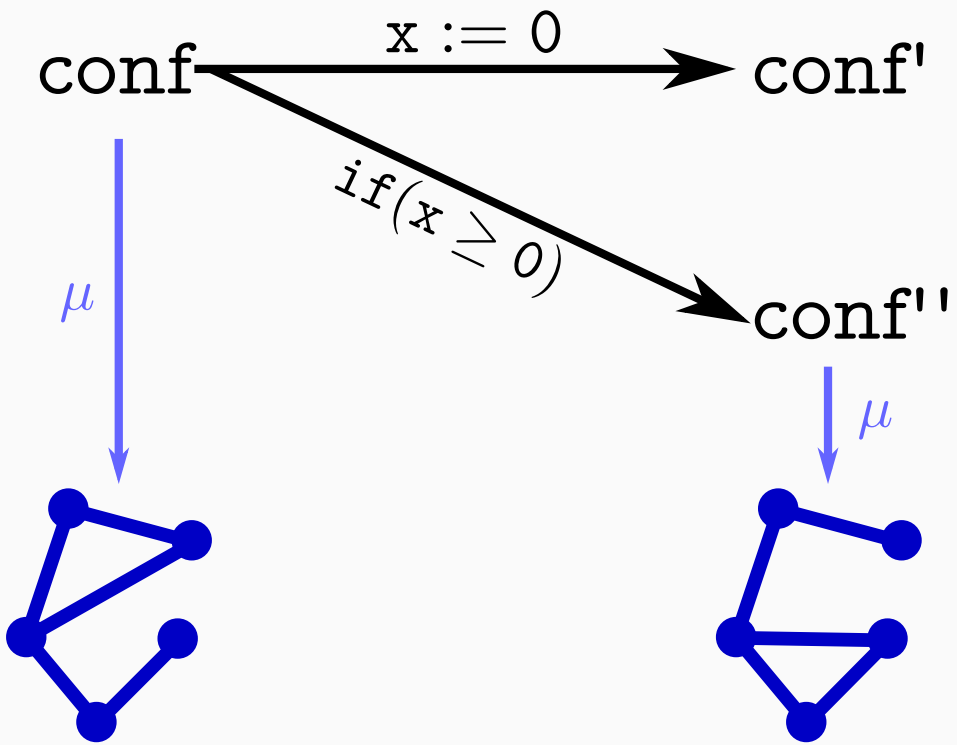
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



Programming and Debugging with Semantically Lifted States, Kamburjan et al. [ESWC'21]

Semantically Lifted States

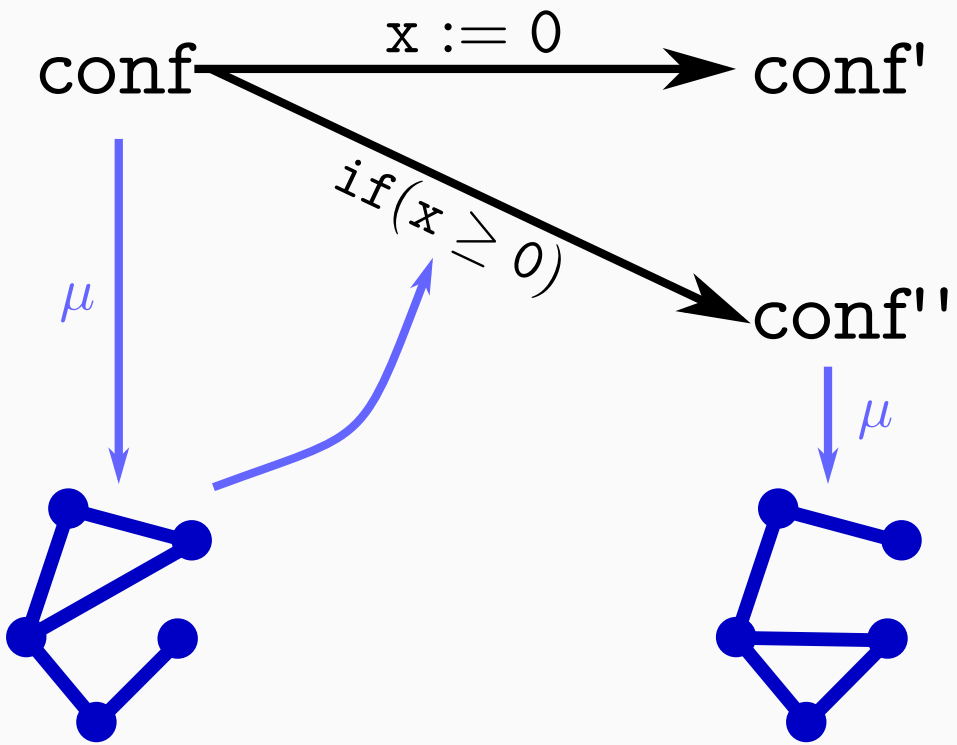
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



Programming and Debugging with Semantically Lifted States, Kamburjan et al. [ESWC'21]

Semantically Lifted States

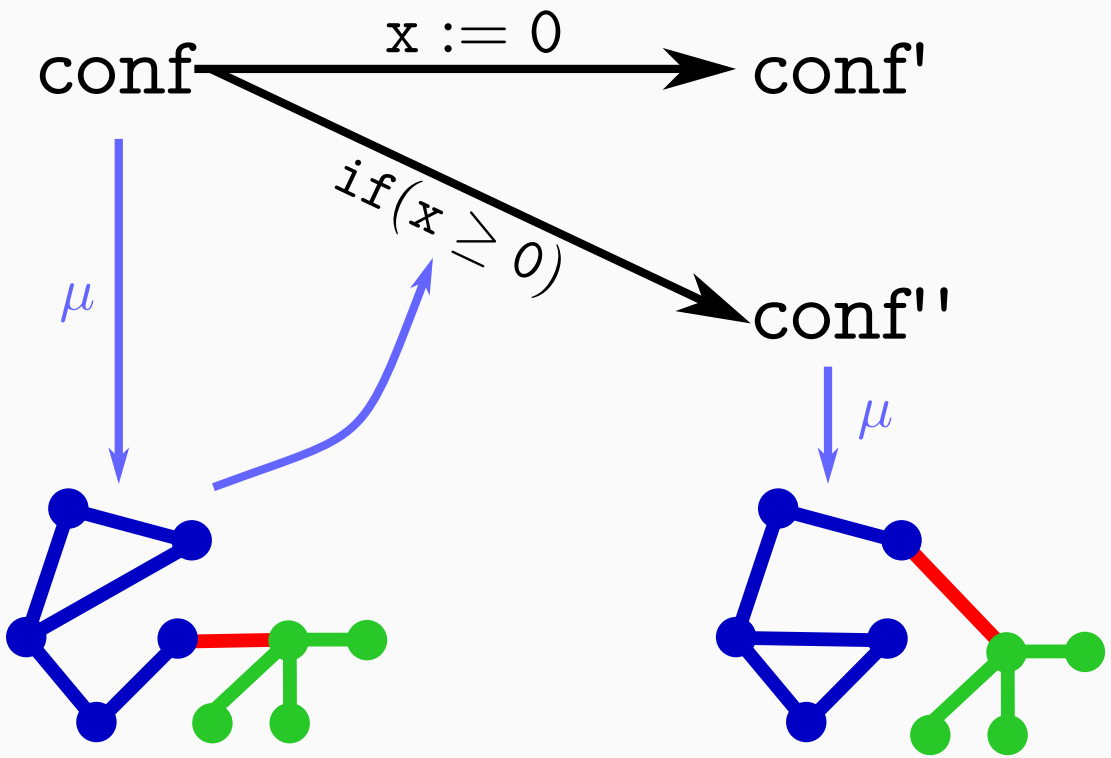
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



Programming and Debugging with Semantically Lifted States, Kamburjan et al. [ESWC'21]

Semantically Lifted States

A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



Programming and Debugging with Semantically Lifted States, Kamburjan et al. [ESWC'21]

Example

```
1 class C (Int i)
2   Unit inc() this.i = this.i + 1; end
3 end
4 main
5   C c = new C(5);
6   Int i = c.inc();
7 end
```

Example

```
1 class C (Int i)
2   Unit inc() this.i = this.i + 1; end
3 end
4 main
5   C c = new C(5);
6   Int i = c.inc();
7 end
```

```
prog:C a prog:class. prog:C prog:hasField prog:C_i.
run:obj1 a prog:C.   run:obj1 prog:C_i 5.
...
prog:C prog:hasMethod prog:C_inc.
prog:inc prog:hasBody prog:s;
...
run:stack run:top run:frame1. run:frame1 run:executes prog:inc.
...
```

SMOL

Implementation

Semantical lifting and reflection is implemented in the **S**emantic **M**icro **O**bject **L**anguage, `smolang.org`.

Given the lifted state, we can use it for multiple operations.

- **Access it** to retrieve objects without traversing pointers.
- **Enrich it** with an ontology, perform logical reasoning and retrieve objects using a query *using the vocabulary of the domain*.
- **Combine it** with another knowledge graph and access external data based on information from the current program state.

Semantic Programming

```
1 class Server(List<Task> taskList) ... end
2 class Scheduler(List<Platform> serverList)
3   Unit reschedule()
4     List<Server> l
5       := access("SELECT ?x WHERE {?x a :Overloaded}");
6     this.adapt(l);
7   end
8 end
```

Semantic Programming

```
1 class Server(List<Task> taskList) ... end
2 class Scheduler(List<Platform> serverList)
3   Unit reschedule()
4     List<Server> l
5       := access("SELECT ?x WHERE {?x a :Overloaded}");
6     this.adapt(l);
7   end
8 end
```

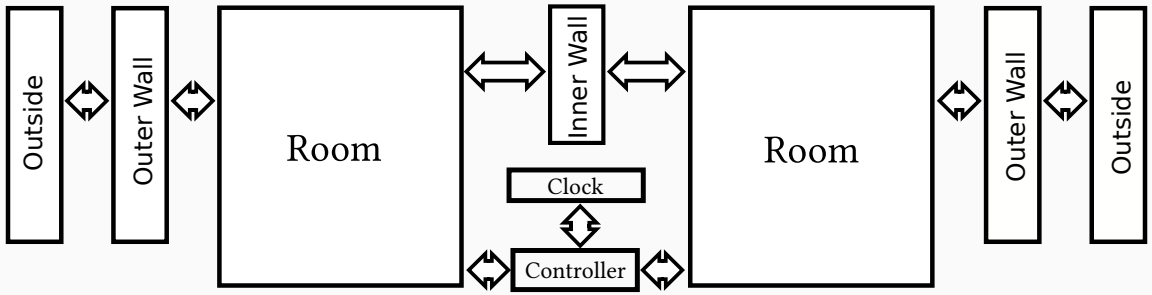
```
:Overloaded
  owl:equivalentClass [
    owl:onProperty (:taskList, :length);
  owl:minValue 3;
].
```

Demo

Semantic Reflection

Example

Back to digital twins



- Monitor consistency
- Monitor twinning
- Adapt to addition of new rooms

Digital Twin Reconfiguration Using Asset Models, Kamburjan et al. [ISoLA'22]

Ensuring Correctness for Self-Adaptive Digital Twins, Kamburjan et al. [ISoLA'22]

Model Description

```
<fmiModelDescription fmiVersion="2.0" modelName="Example" ...>
  <CoSimulation needsExecutionTool="true" .../>
  <ModelVariables>
    <ScalarVariable name="p" variability="continuous"
                    causality="parameter">
      <Real start="0.0"/>
    </ScalarVariable>
    <ScalarVariable name="input" variability="continuous"
                    causality="input">
      <Real start="0.0"/>
    </ScalarVariable>
    <ScalarVariable name="val" variability="continuous"
                    causality="output" initial="calculated">
      <Real/>
  </ModelVariables>
  <ModelStructure> ... </ModelStructure>
</fmiModelDescription>
```

SMOL and FMI

Functional Mock-Up Objects (FMOs)

Tight integration of simulation units using FMI into programs.

```
1 //setup
2 Cont[out Double val] shadow =
3     simulate("Sim.fmu", input=sys.val, p=1.0);
4 Cont[out Double val] sys = simulate("Realsys.fmu");
5 Monitor m = new Monitor(sys,shadow); m.run(1.0);
```

SMOL and FMI

Functional Mock-Up Objects (FMOs)

Tight integration of simulation units using FMI into programs.

```
1 //setup
2 Cont[out Double val] shadow =
3     simulate("Sim.fmu", input=sys.val, p=1.0);
4 Cont[out Double val] sys = simulate("Realsys.fmu");
5 Monitor m = new Monitor(sys,shadow); m.run(1.0);
```

Integration

- Type of FMO directly checked against model description
- Variables become fields, functions become methods
- Causality reflected in type

Knowledge Structures over Simulation Units, Kamburjan and Johnsen [ANNSIM'22]

SMOL and FMI

Functional Mock-Up Interface (FMI)

Standard for (co-)simulation units, called function mock-up units (FMUs). Can also serve as interface to sensors and actuators.

SMOL and FMI

Functional Mock-Up Interface (FMI)

Standard for (co-)simulation units, called function mock-up units (FMUs). Can also serve as interface to sensors and actuators.

```
1 //simplified shadow
2 class Monitor(Cont[out Double val] sys,
3               Cont[out Double val] shadow)
4   Unit run(Double threshold)
5     while shadow != null do
6       sys.doStep(1.0);  shadow.doStep(1.0);
7       if(sys.val - shadow.val >= threshold) then ... end
8     end ...
```

Is this twinning something? Is this setup correctly?

SMOL and FMI

SMOL with FMOs

FMOs are objects, so they are part of the knowledge graph.

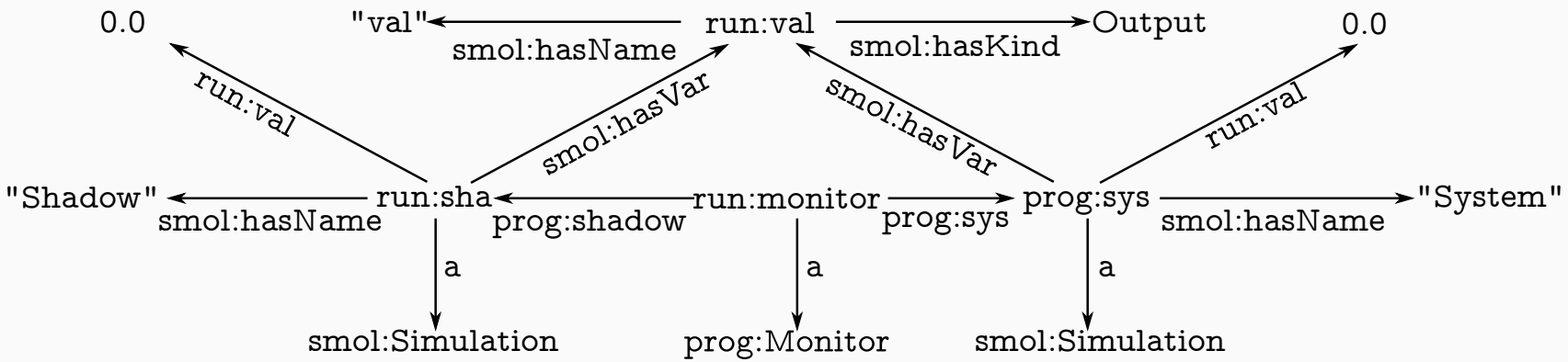
```
1 class Monitor(Cont[out Double val] sys,
2               Cont[out Double val] shadow)
```

SMOL and FMI

SMOL with FMOs

FMOs are objects, so they are part of the knowledge graph.

```
1 class Monitor(Cont[out Double val] sys,
2               Cont[out Double val] shadow)
```



Using the Semantical Lifting

SPARQL

Define structural requirements as queries in SPARQL on *combined* knowledge graph, to use domain constraints on digital twin.

Query to detect non-sensical setups:

```
SELECT ?room WHERE { ?ctrl a prog:Controller.
                     ?ctrl prog:Controller_left ?room.
                     ?ctrl prog:Controller_right ?room }
```

Using the Semantical Lifting

SPARQL

Define structural requirements as queries in SPARQL on *combined* knowledge graph, to use domain constraints on digital twin.

Query to check structural consistency for heaters:

```
SELECT * WHERE { ?o1 prog:Room_id ?id1. ?h1 asset:id ?id1.
                 ?o2 prog:Room_id ?id2. ?h2 asset:id ?id2.
                 ?h1 htLeftOf ?h2.
                 ?c a prog:Controller.
                 ?c prog:Controller_left ?o1.
                 ?c prog:Controller_right ?o2}
```

Demo

Inconsistent Twinning

Self-Adapting to Structural Drift

Detecting Structural Drift

The previous query can detect that some mismatch between asset model and program state exists.
How to detect where the mismatch is and how to repair it?

- Retrieve all assets, and their connections by id (**M**)
- Remove all ids present in the digital twin
- If any id is left, assets needs to be twinned (**A**)
- Find kind of defect to plan repair (**P**)
- Execute repair according to connections (**E**)
- Monitor connections using previous query
- (And v.v. to detect twins that must be removed)

Example: Adding a New Room

- Get all (asset) rooms and their neighboring walls
- Remove all (twinned) rooms with the same id
- Use the information about walls to
- Assumption: at least one new room is next to an existing one

```
1 class RoomAsrt(String room, String wallLt, String wallRt) end
2 ....
3 List<RoomAsrt> newRooms =
4  construct(" SELECT ?room ?wallLt ?wallRt WHERE
5   { ?x a asset:Room;
6        asset:right [asset:Wall_id ?wallRt];
7        asset:left [asset:Wall_id ?wallLt]; asset:Room_id ?room.
8     FILTER NOT EXISTS {?y a prog:Room; prog:Room_id ?room.} }");
```

Demo

Repair

Remarks

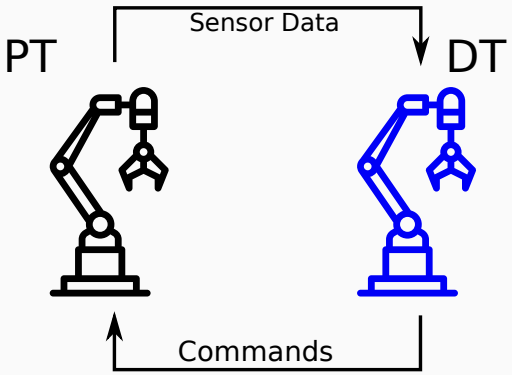
Assumptions

- We know all the possible modifications up-front
  E.g., how to deal with a heater getting new features?
- We know how to always correct structural drift
- Changes do not happen faster than we can repair

Monitoring is still needed to (a) ensure that repairs work correctly, and (b) detect loss of twinning due to, e.g., unexpected structural drift.

Wrap-Up

Summary

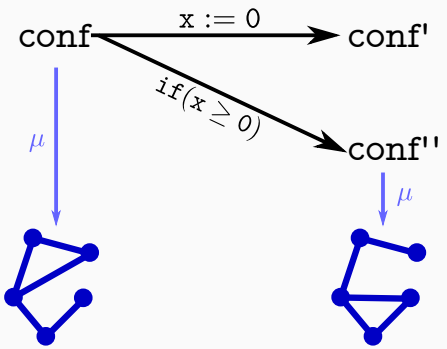


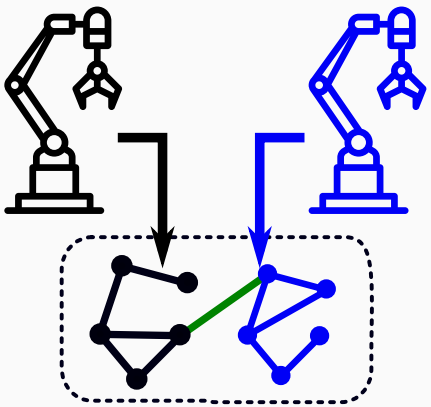
Digital Twins and the FMI

Modern systems with interconnected physical asset and digital model.

Semantic Lifting and Asset Models

Interpret program state as knowledge graph to connect with asset model – use industrial standards.





Structural Self-Adaptation

Use semantic technologies to query and monitor combined knowledge graph from asset model and program state.

Summary

What have we used to construct a self-adaptive, semantically reflected Digital Twin?

Technologies

- Semantic Web technologies
  - OWL/Protege
  - RDF, SPARQL
- Physical modeling, interfacing
  - Modelica, FMI
- SMOL

Concepts

- Digital Twins
- Self-Adaptation through MAPE-K loop
- Semantically lifted programs
- Asset models

Current Research Questions

Digital Twins and Formal Methods

- How to use the fully formal setting for static analysis?
- How to generate digital twins automatically?
- How to deal with concurrency?

Current Research Questions

Digital Twins and Formal Methods

- How to use the fully formal setting for static analysis?
- How to generate digital twins automatically?
- How to deal with concurrency?

Digital Twins@UiO

If you are interested in semantic technologies for programs or digital twins, contact us under

$$\texttt{X@ifi.uio.no},\ \texttt{X} \in \{\text{einarj, sltarifa, rudi, eduard}\}$$

Current Research Questions

Digital Twins and Formal Methods

- How to use the fully formal setting for static analysis?
- How to generate digital twins automatically?
- How to deal with concurrency?

Digital Twins@UiO

If you are interested in semantic technologies for programs or digital twins, contact us under

$$\texttt{X@ifi.uio.no},\ \texttt{X} \in \{\text{einarj, sltarifa, rudi, eduard}\}$$

Thank you for your attention