Twinning-by-Construction:

Ensuring Correctness for Self-Adaptive Digital Twins

Eduard Kamburjan[1]
Crystal Chang Din[2]
Rudolf Schlatte[1]
S. Lizeth Tapia Tarifa[1]
Einar Broch Johnsen[1]

[1]University of Oslo
[2]University of Bergen

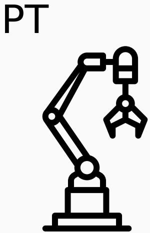
26.10.2022, ISoLA 2022

Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.

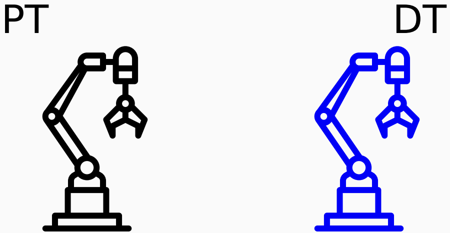
Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



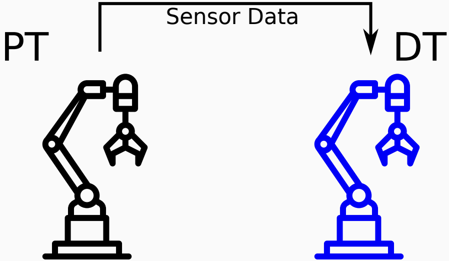
Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



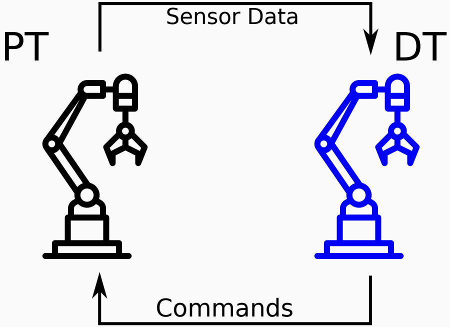
Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



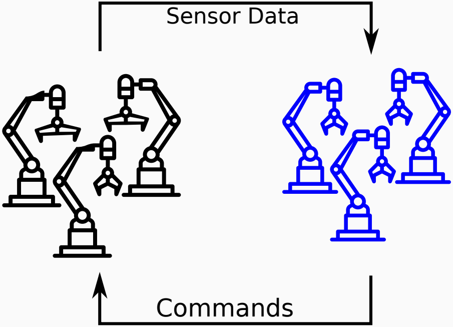
Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



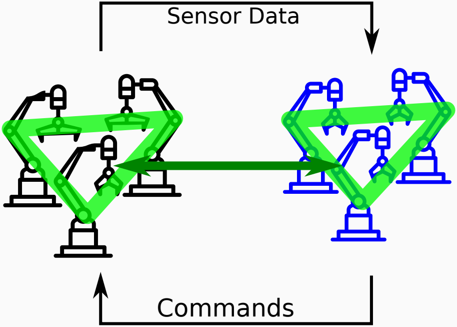
Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



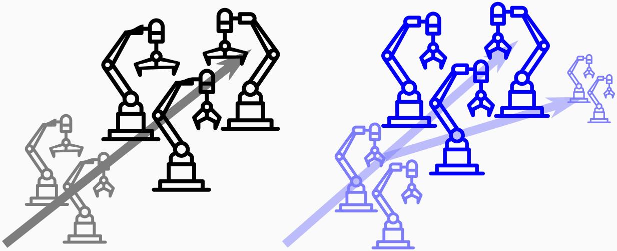
Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



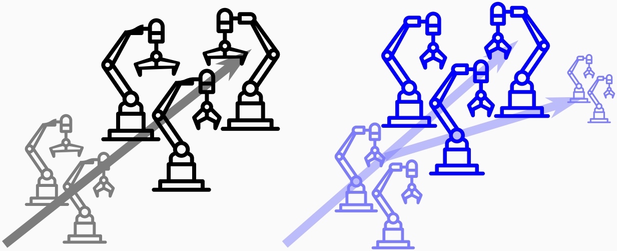
Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



Digital Twins and Twinning

A digital twin system connects a physical asset with its own (simulation) models using data streams and commands.



- How to ensure correctness/twinning in this setting?
- How to express twinning?

Digital Thread and Life-cycle Management

The Digital Twin Evolves in Tandem with the Asset

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

Reconfiguration in the Operation Phase

- Behavioral drift: the twin's components need to be adjusted
- Structural drift: composition of components needs to change

Challenges

Challenge 1: Formalizing Properties

- How to express twinning?
- How to represent physical and digital twin?

Solution: Semantic Technologies and Ontologies

Challenge 2: Self-Adaptation

- How to adapt to changes in the physical twin?
- How to use semantic technologies for self-adaptation?

Solution: Use MAPE-K framework from robotics

Challenge 3: Digital Thread

- Can we express twinning over the digital thread?

Solution: Integrate, Record and Monitor into Semantic Twin

Semantically Lifted Programs and Digital Twins

Semantic Technologies

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

Semantic Technologies

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

Semantic Technologies

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

RDF:

```
Peter a Person. Paul a Person. Maria a Person.
Peter hasChild Paul. Paul hasChild Maria.
```

Semantic Technologies

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

RDF:

```
Peter a Person. Paul a Person. Maria a Person.
Peter hasChild Paul. Paul hasChild Maria.
```

OWL:

`hasChild` **some** (`hasChild` **some** `Person`)

subClassOf `GrandParent`

Semantic Technologies

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

RDF:

```
Peter a Person. Paul a Person. Maria a Person.
Peter hasChild Paul. Paul hasChild Maria.
```

OWL:

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

SPARQL:

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

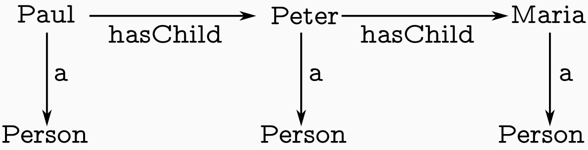
Knowledge Graphs

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.



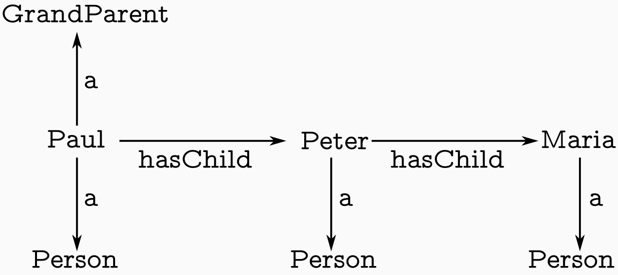
Knowledge Graphs

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.



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

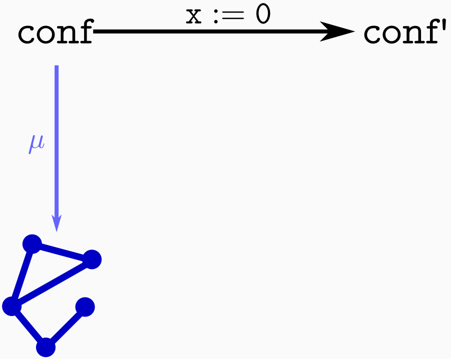
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.

$$\text{conf} \xrightarrow{\texttt{x} := 0} \text{conf'}$$

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

Semantically Lifted States

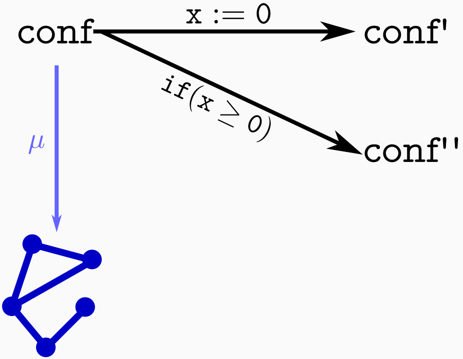
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



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

Semantically Lifted States

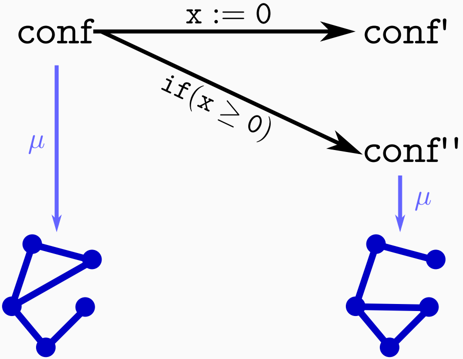
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



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

Semantically Lifted States

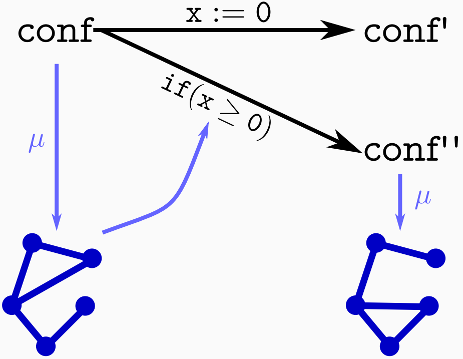
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



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

Semantically Lifted States

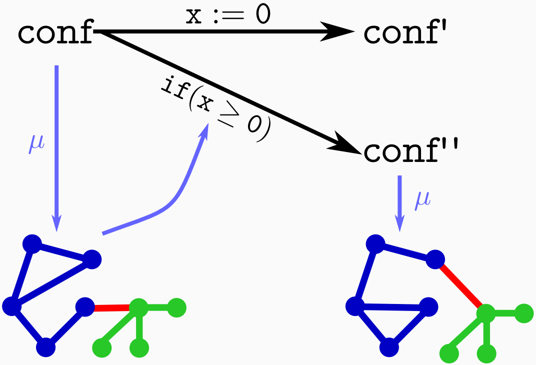
A semantically lifted program can interpret its own program state as a knowledge graph and reflect on itself through it.



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]

Semantic Programming

```
class Platform(List<Server> serverList) ... end
class Server(List<Task> taskList) ... end
class Scheduler(List<Platform> platformList)
 Unit reschedule()
  List<Platform> l
    := access("SELECT ?x WHERE {?x a :Overloaded}");
  this.adaptPlatforms(l);
 end
end
```

Semantic Programming

```
class Platform(List<Server> serverList) ... end
class Server(List<Task> taskList) ... end
class Scheduler(List<Platform> platformList)
 Unit reschedule()
  List<Platform> l
    := access("SELECT ?x WHERE {?x a :Overloaded}");
  this.adaptPlatforms(l);
 end
end
```

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

Knowledge Graphs and Asset Models

Asset Model

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

Our Asset Model

For now: A knowledge graph describing the current structure of the physical twin.

Knowledge Graphs and Asset Models

Asset Model

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

Our Asset Model

For now: A knowledge graph describing the current structure of the physical twin.

```
ast:w1 a ast:Wall. ast:w2 a ast:Wall.
ast:w1 ast:id 13. ast:w2 ast:id 12.
ast:w1 ast:leftOf ast:w2.
```

Checking the Twinning Property

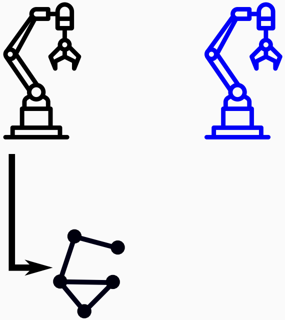
Combining the Knowledge

- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge

Checking the Twinning Property

Combining the Knowledge

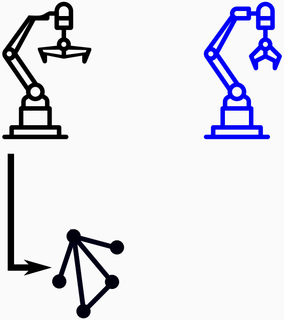
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Checking the Twinning Property

Combining the Knowledge

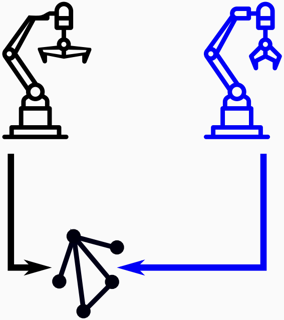
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Checking the Twinning Property

Combining the Knowledge

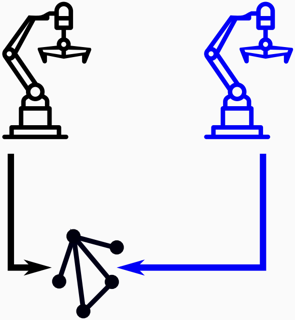
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Checking the Twinning Property

Combining the Knowledge

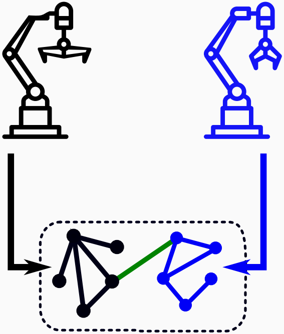
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Checking the Twinning Property

Combining the Knowledge

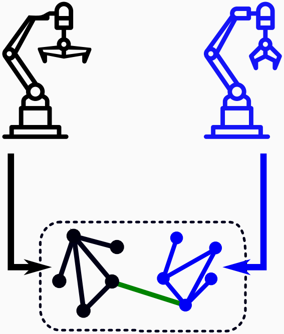
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Checking the Twinning Property

Combining the Knowledge

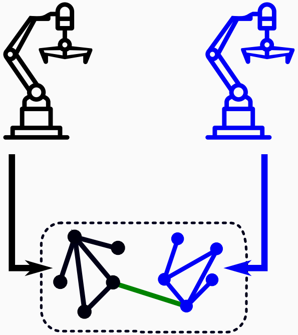
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Checking the Twinning Property

Combining the Knowledge

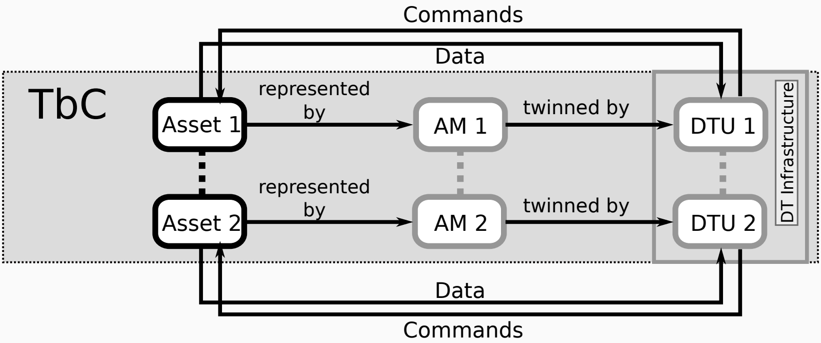
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Checking the Twinning Property

Combining the Knowledge

- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



MAPE-K

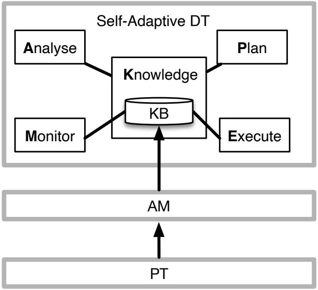
MAPE-K

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? ⇒ 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? ⇒ Semantical lifting.

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

MAPE-K for Semantic Digital Twins

Monitoring Twinning

Monitor

Check whether all assets in production (`as:InProd`) are twinned (`dti:twin`) by some DTU (`dti:DTUnit`).

```
SELECT ?x { ?x a as:InProd.
  FILTER NOT EXISTS (?y a dti:DTUnit. ?y dti:twins ?x.)
}
```

Monitoring Twinning

Monitor

Check that all DTUs that exist are connected the same way as their PTs.

```
SELECT ?dtu { ?dtu a dti:DTUnit. ?dtu dti:twins ?asset.
  OPTIONAL(
    ?asset as:leftOf ?right.
    FILTER NOT EXISTS (
      ?dtuRight a dti:DTUnit.
      ?dtu dti:leftOf ?dtuRight.
      ?dtuRight dti:twins ?right.
    )
  )
}
```

Simple Twinning

Analyze

Both queries most return an empty set. Let their conjunction be denoted `SIMPLE`. If $\texttt{SIMPLE} = \emptyset$, then the system is simply twinned.

Planning

Plan creation and reconnection of DTUs according to query results. Eventually, run additional queries.

Execution

Retwin system: create DTUs, initialize them and reconnect if necessary.

Simple Twinning

Analyze

Both queries most return an empty set. Let their conjunction be denoted `SIMPLE`. If $\texttt{SIMPLE} = \emptyset$, then the system is simply twinned.

Planning

Plan creation and reconnection of DTUs according to query results. Eventually, run additional queries.

Execution

Retwin system: create DTUs, initialize them and reconnect if necessary.

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

Temporal Twinning

Beyond Simple Twinning

- It is easy to see that $\square \mathtt{SIMPLE} = \emptyset$ does not hold.
- Additionally to twinning, one monitors the temporal property that within n time steps simple twinning is reestablished

$$m\text{−Change} \equiv \square\left(\mathtt{CHANGE} \rightarrow \neg\Diamond_{(0,m]}\mathtt{CHANGE}\right)$$

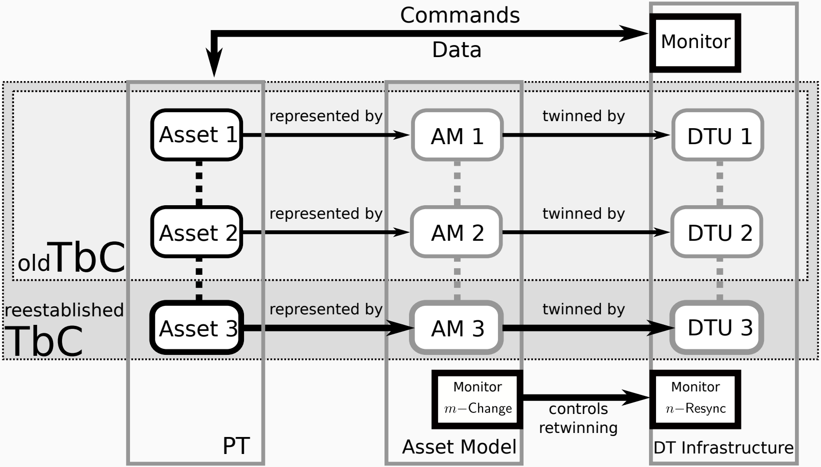
Twinning

Twinning takes some time, and the asset changes regularly – we must retwin faster than the asset model changes

$$n\text{−Resync} \equiv \square\left(\mathtt{SIMPLE} \not\doteq \emptyset \rightarrow \Diamond_{[0,n]}\mathtt{SIMPLE} \doteq \emptyset\right)$$

$$\mathtt{TEMP}_{m,n} \equiv m > n \land m\text{−Change} \land n\text{−Resync}$$

Monitoring Structure



Digital Thread

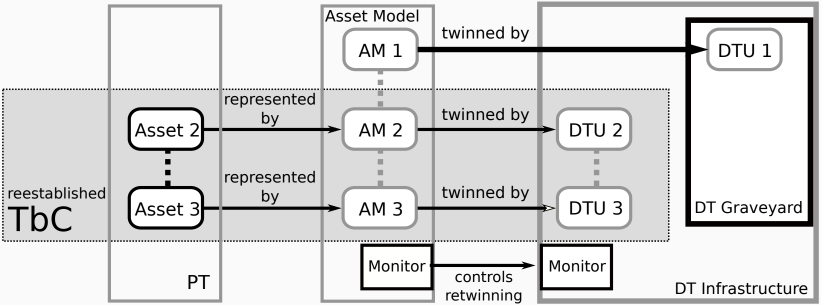
So far we checked the addition of assets, what about further operation recorded in the digital thread?

1. Build walls `as:w1` and `as:w2`
2. Build wall `as:w3`
3. Replace `as:w2` with `as:w3`
4. Decommission `as:w3`

```
as:w1 a as:Wall. as:w2 a as:Wall. as:w3 a as:Wall.
as:w1 a as:InProd. as:w2 a as:Decom. as:w3 a as:InProd.
as:w1 as:leftOf as:w3. as:w3 as:replaces as:w2.
```

Digital Thread

Idea: keep “old” DTUs, mirror structure of thread



General Observation

Digital Twin Infrastructure becomes more Thread-like

- Graveyard is twinning decommissioned assets
- Monitors are twinning requirements

Queries over the Digital Thread

Queries over the Digital Thread

What is the wall left of whatever is in the place of wall `w2` now?

```
SELECT ?x {?x as:leftOf [as:replaces* as:w2]}
```

Queries over the Reflective Twin

Which reconfiguration are triggered by water damage?

```
SELECT ?reconf {
       ?a as:observed ?ev. ?ev a as:WaterDamage.
       ?d dti:twins ?asset. ?d dti:reconfiguration ?rcf.
       ?ev as:at ?dt. ?rcf dti:at ?dt.}
```

Twinning and the Digital Thread

A system has the simple temporal twinning property, if the DTU of every removed asset is moved to the graveyard. (TEMPSIMPLE)

```
SELECT ?x { ?x a dti:DTUnit.
  FILTER NOT EXISTS(
    ?x dti:twins ?asset. ?asset a as:Decom.
    ?asset as:removedAt [a as:Removal; as:at ?dt1].
    ?x dti:removal [a dti:Remove; dti:at ?dt2].
    FILTER ( microsec(?dt2) - microsec(?dt1) < 5*60*1000 )
  )
}
```

$$\mathtt{TEMP}_{m,n} \land \Box\big(\mathtt{TempSimple} \not\doteq \emptyset \rightarrow \Diamond_{[0,n]}\mathtt{TempSimple} \doteq \emptyset\big)$$

Conclusion

Conclusion

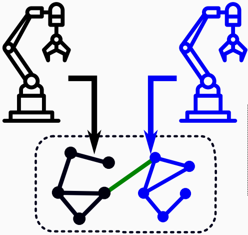
Self-Adaptive, Semantically Reflected, Digital Twins

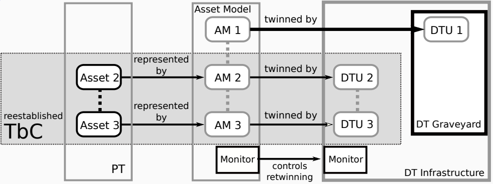
- Combining knowledge representation and programming
- Generate (correct) twin from asset model, monitor that twinning property is uphold
- Future work: formal verification of twinning

Conclusion

Self-Adaptive, Semantically Reflected, Digital Twins

- Combining knowledge representation and programming
- Generate (correct) twin from asset model, monitor that twinning property is uphold
- Future work: formal verification of twinning

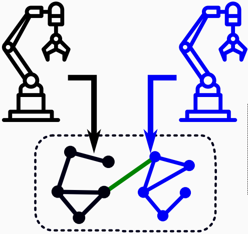


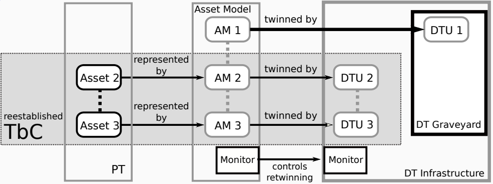


Conclusion

Self-Adaptive, Semantically Reflected, Digital Twins

- Combining knowledge representation and programming
- Generate (correct) twin from asset model, monitor that twinning property is uphold
- Future work: formal verification of twinning





Thank you for your attention