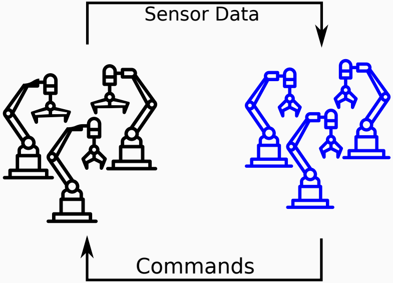
Declarative Lifecycle Management in Digital Twins

Eduard Kamburjan[1]
Nelly Bencomo[2]
Sylvia Lizeth Tapia Tarifa[1]
Einar Broch Johnsen[1]

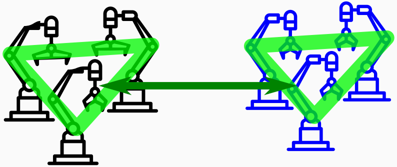
[1]University of Oslo
[2]Durham University

23.09.2024, EDTConf'24, Linz

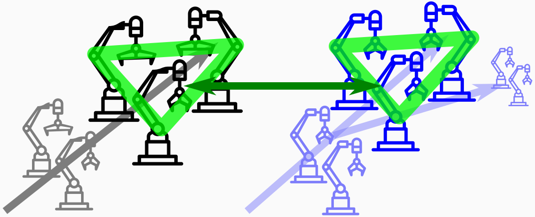
Digital Twins as Self-Adaptive Systems



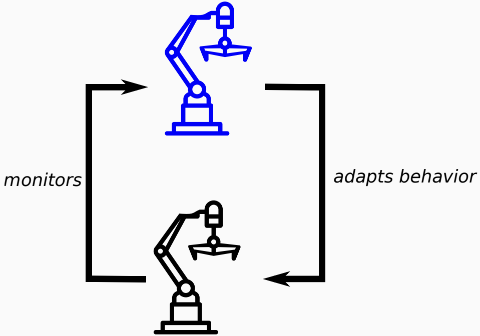
Digital Twins as Self-Adaptive Systems



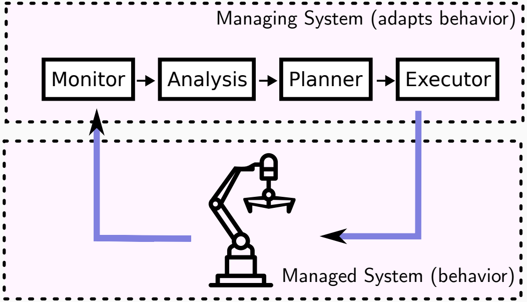
Digital Twins as Self-Adaptive Systems



Digital Twins as Self-Adaptive Systems



Digital Twins as Self-Adaptive Systems

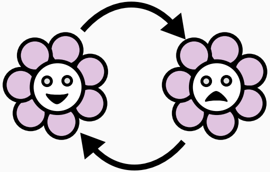


- Managing system itself must adapt to more changes of the managed system
- Changes in structure and lifecycle stage of the managed system

Contribution: A two-layered self-adaptation architecture for lifecycles in DTs

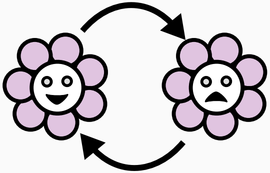
Digital Twins and Structural Self-Adaptation

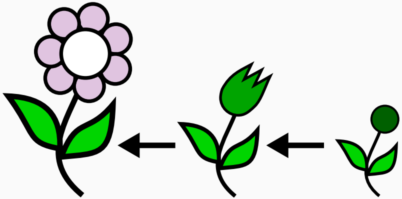
Lifecycles and Structural Self-Adaptation



- Components of the physical twin have different lifecycle stages
- Each lifecycle stage requires a different setup, different MAPE components etc.

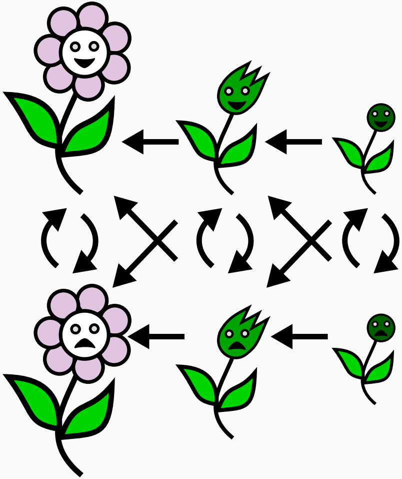
Lifecycles and Structural Self-Adaptation





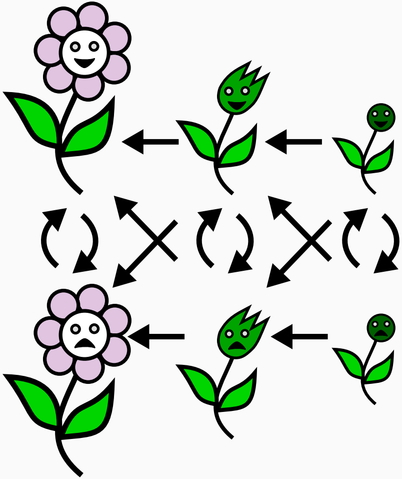
- Components of the physical twin have different lifecycle stages
- Each lifecycle stage requires a different setup, different MAPE components etc.
- May also be part of multiple lifecycles, lifecycles may interact

Lifecycles and Structural Self-Adaptation



- Components of the physical twin have different lifecycle stages
- Each lifecycle stage requires a different setup, different MAPE components etc.
- May also be part of multiple lifecycles, lifecycles may interact
- Do we really need to model the whole transition system?

Lifecycles and Structural Self-Adaptation

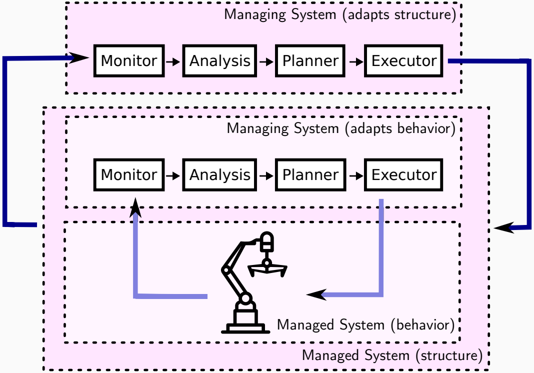


- Components of the physical twin have different lifecycle stages
- Each lifecycle stage requires a different setup, different MAPE components etc.

Operational vs. Declarative Lifecycles

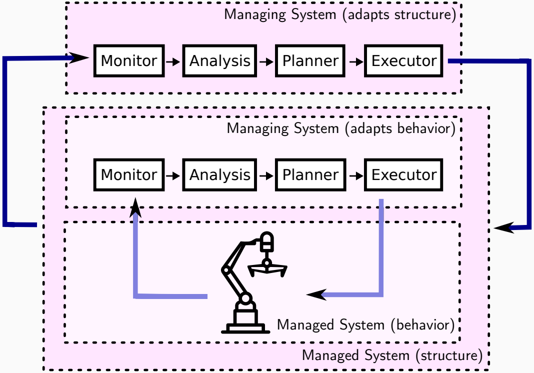
- An operational lifecycle describes how to change between stages
- A declarative lifecycle describes what it means to by at a stage

Digital Twins as Two-Layered Self-Adaptive Systems


Managing System (adapts structure)
Monitor
Analysis
Planner
Executor
Managing System (adapts behavior)
Monitor
Analysis
Planner
Executor
Managed System (behavior)
Managed System (structure)

- Second layer of self-adaptation
- Monitors the *structure* of the level-1 system
- Does also consider the state of the PT
- E.g., given a sick plant, do I have the right components to monitor its specific health requirements?

Digital Twins as Two-Layered Self-Adaptive Systems


Managing System (adapts structure)
Monitor
Analysis
Planner
Executor
Managing System (adapts behavior)
Monitor
Analysis
Planner
Executor
Managed System (behavior)
Managed System (structure)

- Lifecycle stages are declarative, with two elements as their definition
- *membership predicate:* When an asset is considered to be in a stage
- *consistency predicate:* When an asset's assigned components are considered consistent with its stage
- Self-adaptation is generic: Abduct an explanation with which components as asset would be consistent with its detected stage

Declarative Stages

Declarative Lifecycle Stages

Definition (Stage)

Let $\mathcal{A}$ be an asset class. Let $\mathcal{C}$ be a set of component classes.

$$\mathrm{D}_{\mathcal{A},\mathcal{C}} = \langle \mathit{member}, \mathit{consistent} \rangle$$

- $\mathit{member} \subseteq \mathcal{A}$ are the target assets
- $\mathit{consistent} \subseteq \mathit{member} \times 2^{\mathcal{C}}$ are the required components

$$\begin{aligned}
\mathrm{D}_{\mathsf{Sick}} &= \{\mathit{member}_{\mathsf{Sick}}, \mathit{consistent}_{\mathsf{Sick}}\} \\
\mathit{member}_{\mathsf{Sick}} &= \{a \mid \mathrm{ndvi}(a) \leq 0.5\} \\
\mathit{consistent}_{\mathsf{Sick}} &= \{(a, X) \mid a \in \mathit{member}_{\mathsf{Sick}}, \mathrm{analyzer}^{\leq 5}_{\mathit{moisture}}(a) \in X\}
\end{aligned}$$

Lifecycles

Definition (Lifecycle)

Let $\mathcal{A}$ be an asset class and I an index set. A *lifecycle* $\mathsf{L}_{\mathcal{A}}$ for $\mathcal{A}$ is a set of declarative stages $\left(\mathsf{D}_{\mathcal{A},\mathcal{C},i}\right)_{i\in I}$ such that every asset from $\mathcal{A}$ is in exactly one stage:
(1) $\mathcal{A} = \bigcup_{i\in I} \mathit{member}_{\mathsf{D}_{\mathcal{A},\mathcal{C},i}}$ (2) $\forall i, j \in I.\ i \neq j \Rightarrow \mathit{member}_{\mathsf{D}_{\mathcal{A},\mathcal{C},i}} \cap \mathit{member}_{\mathsf{D}_{\mathcal{A},\mathcal{C},j}} = \emptyset$

$$
\begin{aligned}
\mathsf{D}_{\mathsf{Healthy}} &= \{\mathit{member}_{\mathsf{Healthy}}, \mathit{consistent}_{\mathsf{Healthy}}\} \\
\mathit{member}_{\mathsf{Healthy}} &= \{a \mid \mathsf{ndvi}(a) > 0.5\} \\
\mathit{consistent}_{\mathsf{Healthy}} &= \{(a, X) \mid a \in \mathit{member}_{\mathsf{Healthy}}, \mathsf{analyzer}^{\leq 10}_{\mathit{moisture}}(a) \in X\}
\end{aligned}
$$

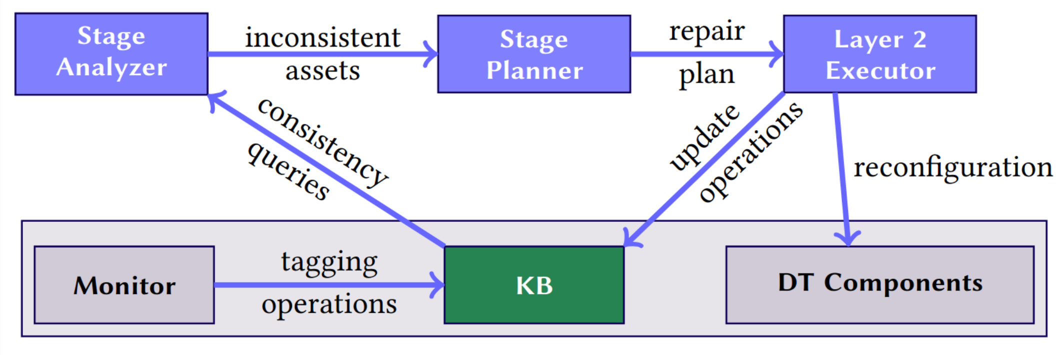
Compatible Lifecycles

Definition (Compatibility)

The stages D_1 and D_2 are *compatible* if, for all $a \in \mathit{member}_{\mathrm{D}_1} \cap \mathit{member}_{\mathrm{D}_2}$ there is some $X \subseteq \overline{\mathcal{C}}$ such that $(a, X) \in \mathit{consistent}_{\mathrm{D}_1}$ and $(a, X) \in \mathit{consistent}_{\mathrm{D}_2}$

- Two lifecycles are compatible if all their stages are compatible
- Compatible stages may restrict each others consistency, but not make it impossible
- Simple composition, akin to cross-product

Specialized Architecture



- KB keeps track of tagged values from physical twin
- KB keeps track of assignment from layer-1 components to assets
- Each change in components must be recorded

Algorithm (simplified)

Definition (Abduction-Based Self-adaptation)

For one asset a, with one lifecycle.

1. Retrieve assigned components X (**Monitor**)
2. Check if $a \in \mathit{member}_{\mathrm{D}} \land (a, X) \notin \mathit{consistent}_{\mathrm{D}}$ (**Analyze**)
3. If so, abduce for which X', we have $(a, X') \in \mathit{consistent}_{\mathrm{D}}$ (**Plan**)
4. Remove components in $X \setminus X'$ (**Execute**)
5. Add components in $X' \setminus X$ (**Execute**)

- Require logical representation of asset and component information
- Full details for multiple lifecycles in paper, requires to abduce over all consistency sets at the same time.

Example

- New sensors value indicates that plant P is sick, but inner loop is still for the healthy one

$$\begin{aligned} &\mathrm{ndvi}(\mathtt{P}) \doteq 0.4, && \mathtt{P} \in \mathit{member}_{\mathsf{Sick}}, \\ &\mathrm{analyzer}^{\leq 10}_{\mathit{moisture}}(\mathtt{P}) \in X, && \mathtt{P} \notin \mathit{consistent}_{\mathsf{Sick}} \end{aligned}$$

- Abduce solution

$$\mathrm{analyzer}^{\leq 5}_{\mathit{moisture}}(\mathtt{P}) \in X$$

- Generate and execute plan

$$\begin{aligned} &\mathrm{ndvi}(\mathtt{P}) \doteq 0.4, && \mathtt{P} \in \mathit{member}_{\mathsf{Sick}}, \\ &\mathrm{analyzer}^{\leq 5}_{\mathit{moisture}}(\mathtt{P}) \in X, && \mathtt{P} \in \mathit{consistent}_{\mathsf{Sick}} \end{aligned}$$

Conclusion

Semantic Architecture

Requirements for Implementation

- Reasoning, especially deductive and abductive reasoning
- Representing structured data
- Easy to query, able to model asset data

Semantic Architecture

Requirements for Implementation

- Reasoning, especially deductive and abductive reasoning
- Representing structured data
- Easy to query, able to model asset data
  ⇒ **Knowledge Graphs**

Semantic Architecture

Requirements for Implementation

- Reasoning, especially deductive and abductive reasoning
- Representing structured data
- Easy to query, able to model asset data
  ⇒ **Knowledge Graphs**

- Evaluation on GreenhouseDT exemplar for self-adaptive systems
- Details in paper

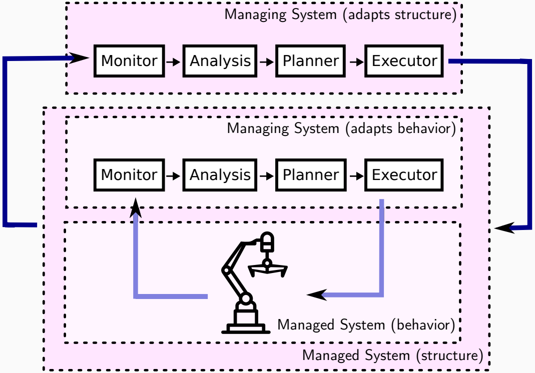
Conclusion

Contributions

- Declarative formalization and management of lifecycles
- Generic, two-layered self-adaptation for digital twins

Future Work

- Further composition operations on declarative stages
- *n*-layered self-adaptive Digital twins



Managing System (adapts structure)
Monitor
Analysis
Planner
Executor
Managing System (adapts behavior)
Monitor
Analysis
Planner
Executor
Managed System (behavior)
Managed System (structure)

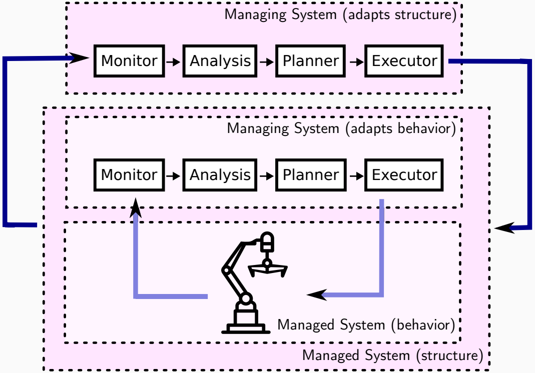
Conclusion

Contributions

- Declarative formalization and management of lifecycles
- Generic, two-layered self-adaptation for digital twins

Future Work

- Further composition operations on declarative stages
- *n*-layered self-adaptive Digital twins


Managing System (adapts structure)
Monitor
Analysis
Planner
Executor
Managing System (adapts behavior)
Monitor
Analysis
Planner
Executor
Managed System (behavior)
Managed System (structure)

Thank you for your attention