Towards Automating the Life Cycle Management of Digital Twins

Gwendal Beaumont,
Antoine Beugnard, Salvador Martínez, Christelle Urtado, Sylvain Vauttier

Lab-STICC UMR 6285, P4S team, IMT Atlantique, Brest

44th International Conference on Conceptual Modeling
October 23, 2025

Towards Automating the Life Cycle Management of Digital Twins

Gwendal Beaumont,
Antoine Beugnard, Salvador Martínez, Christelle Urtado, Sylvain Vauttier

Lab-STICC UMR 6285, P4S team, IMT Atlantique, Brest

44th International Conference on Conceptual Modeling
October 23, 2025

- Growing interests in *Digital Twins*
  → 73k+ results on ScienceDirect
  → 90k+ results on GitHub[1]
  → Deployment of DTs is projected to see an average growth of 36% within five years

Figure: Fischertechnik Training Factory

[1] https://github.com/Gwendal-Beaumont/DT-Mining

Everyone Does DTs... 2 / 18
• Growing interests in *Digital Twins*
→ 73k+ results on ScienceDirect
→ 90k+ results on GitHub[2]
→ Deployment of DTs is projected to see an average growth of 36% within five years
Figure: Fischertechnik Training Factory
[2] https://github.com/Gwendal-Beaumont/DT-Mining

- Growing interests in *Digital Twins*
  → 73k+ results on ScienceDirect
  → 90k+ results on GitHub[2]
  → Deployment of DTs is projected to see an average growth of 36% within five years

- **Life cycle management**
  → Lack of a common DT life cycle
- **Non-explicit life cycle**
  → Hard to manipulate and reason upon
- **Re-usability and modularity**
  → Digital Twins can hardly be duplicated or re-used

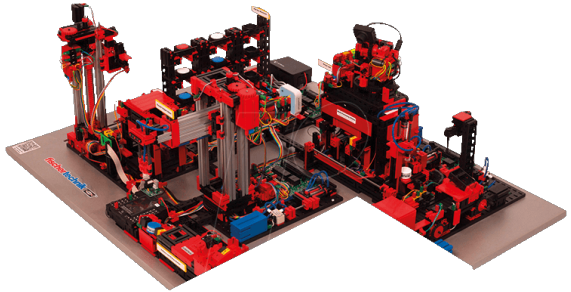


Figure: Fischertechnik Training Factory without SLD module

...In Heterogeneous Ways 3 / 18
• Life cycle management
→ Lack of a common DT life cycle
• Non-explicit life cycle
→ Hard to manipulate and reason upon
• Re-usability and modularity
→ Digital Twins can hardly be duplicated or re-used
Figure: Fischertechnik Training Factory without SLD module

1. However, as we ventured in the realms of DTs, we noticed they have not been created equally
   - Common guidelines
   - Common structure
2. This leads to problems such as: re-usability and modularity
   → hence there's a need for more reusable and modular DTs

- Lack of generic life cycle management system
  → Based on a survey and personal experiments we created our own *generic life cycle operations*

This comprises operations such as:

- `start`
- `configure`
- `connect`
- `synchronize`
- `clone`
- `save`
- `stop`
- `archive`
- `restore`
- `delete`

Declarative Lifecycle Management in Digital Twins

Eduard Kamburjan, Nelly Bencomo, Silvia Lizeth Tapia Tarifa, Einar Broch Johnsen

JuNo-OPS: A DevOps Framework for the Engineering of Digital Twins for Built Assets

Sara Aissat, Jonathan Beaulieu, Francis Bordeleau

Digital Twin as a Service (DTaaS): A Platform for Digital Twin Developers and Users

Prinad Talkhestani, Cláudio Gomes, Peter Hägi Mikkelsen, Santiago Gil Arboleda, Eduard Kamburjan, Peter Gorm Larsen

Systems Engineering

WILEY

COMMUNICATION OPEN ACCESS

Model-Driven Engineering for Digital Twins: Opportunities and Challenges

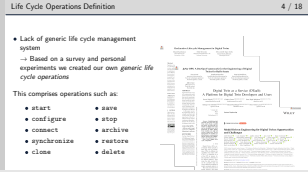
Judith Michael, Luca Cleophas, Steffen Zschaler, Tony Clark, Benoit Combemale, Thomas Godfrey, Djamel Eddine Khelladi, Vinay Kulkarni, Daniel Lehner, Bernhard Rumpe, Manuel Wimmer, Andreas Wortmann, Shankar Ali, Heiko Barr, Ian Barnard, Nelly Bencomo, Francis Bordeleau, Georg Grossmann, Gabor Kiesel, Oliver Kopp, Reinhard Kugler, Parth Mahajan, Alfonso Pierantonio, Ulrich A. C. Schulz, Martin Schindler, Holger Schlingloff, Markus Stumptner, Antonio Vallecillo, Hans Vangheluwe, Mark van den Brand

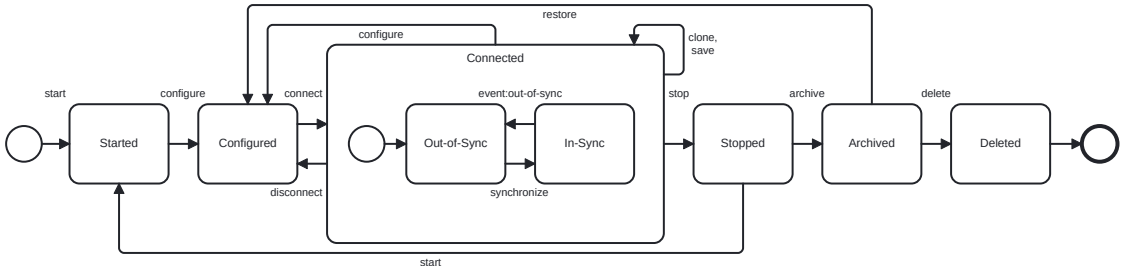
Correspondence: Judith Michael (michael@se-rwth.de)

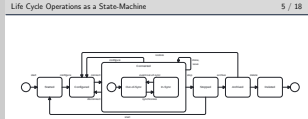
Received: 10 March 2025 | Revised: 1 February 2026 | Accepted: 10 March 2026

Funding: This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) and the Agence Nationale De La Recherche (ANR)—France—Model Based DevOps—509969070 and ANR-22-CHR-0006, and the Key Digital Technologies (KDT) Joint Undertaking through the European Union's Horizon Europe project MATHEO, grant agreement No. 101140216.

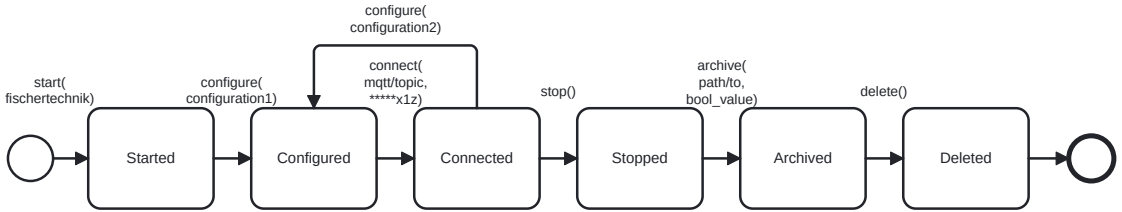
Keywords: cyber physical systems | digital twin | model-driven engineering | systems engineering

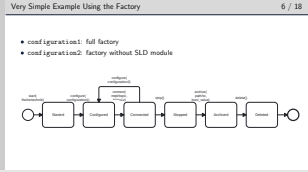



restore
configure
clone,
save
Connected
start
configure
connect
event:out-of-sync
stop
archive
delete
Started
Configured
Out-of-Sync
In-Sync
Stopped
Archived
Deleted
disconnect
synchronize
start

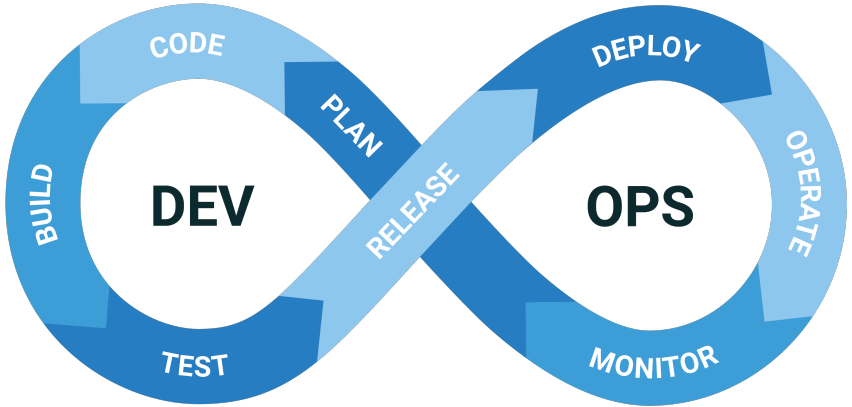


- `configuration1`: full factory
- `configuration2`: factory without SLD module

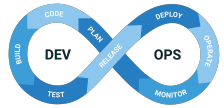




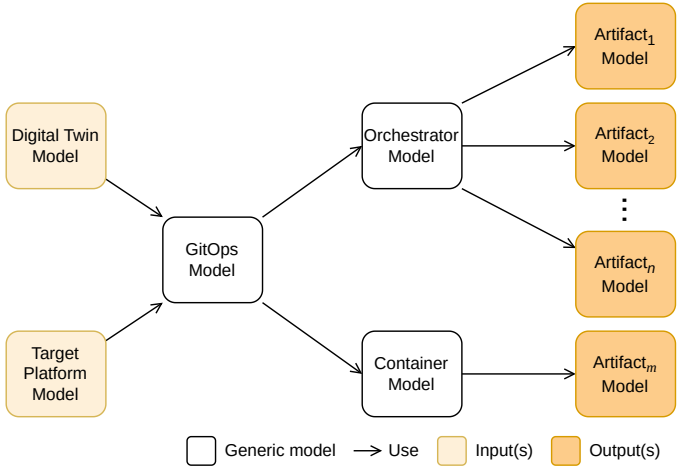
→ Focus on the **operational** phase

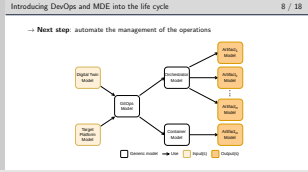


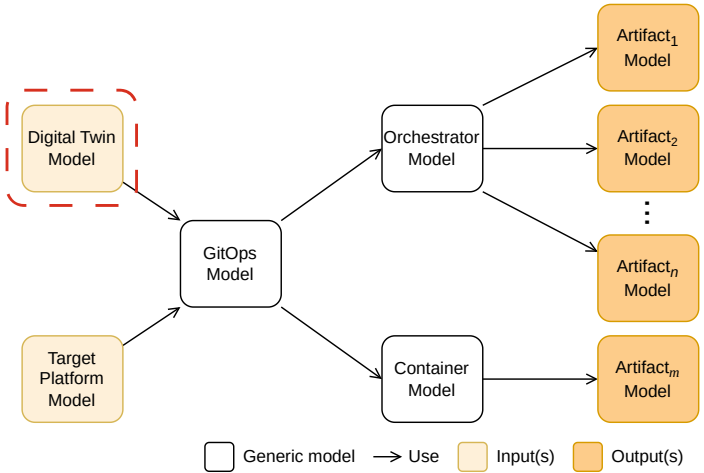
Towards Automation 7 / 18
→ Focus on the **operational** phase

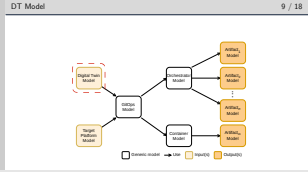


→ **Next step**: automate the management of the operations

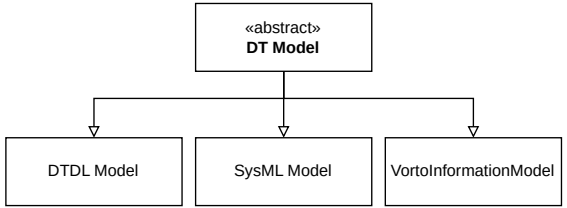




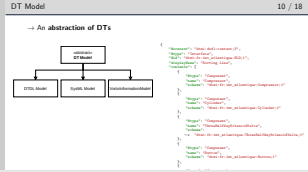

Digital Twin Model
GitOps Model
Target Platform Model
Orchestrator Model
Container Model
Artifact$_1$ Model
Artifact$_2$ Model
Artifact$_n$ Model
Artifact$_m$ Model
Generic model
Use
Input(s)
Output(s)

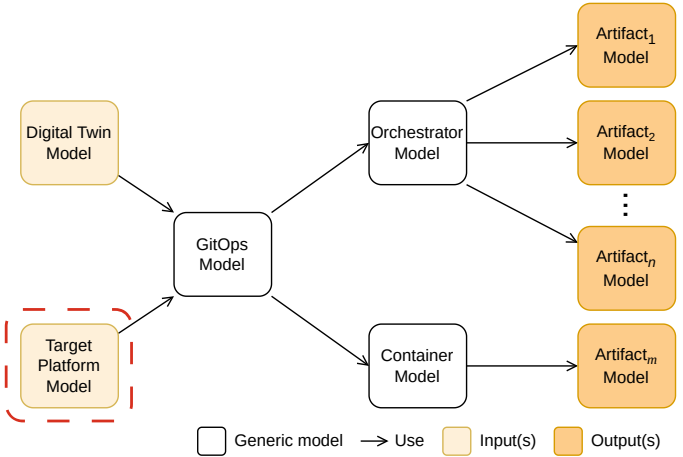


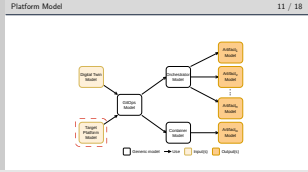
→ An **abstraction of DTs**




```
{
    "@context": "dtmi:dtdl:context;3",
    "@type": "Interface",
    "@id": "dtmi:fr:imt_atlantique:SLD;1",
    "displayName": "Sorting Line",
    "contents": [
        {
            "@type": "Component",
            "name": "Compressor",
            "schema": "dtmi:fr:imt_atlantique:Compressor;1"
        },
        {
            "@type": "Component",
            "name": "Cylinder",
            "schema": "dtmi:fr:imt_atlantique:Cylinder;1"
        },
        {
            "@type": "Component",
            "name": "ThreeHalfWaySolenoidValve",
            "schema":
            ↪  "dtmi:fr:imt_atlantique:ThreeHalfWaySolenoidValve;1"
        },
        {
            "@type": "Component",
            "name": "Button",
            "schema": "dtmi:fr:imt_atlantique:Button;1"
        },
        {
```

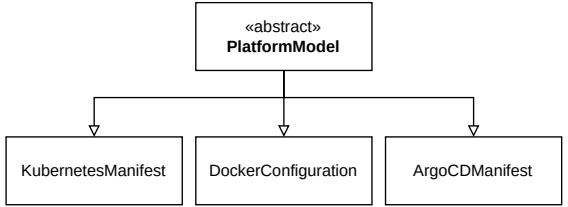




Digital Twin Model
GitOps Model
Target Platform Model
Orchestrator Model
Container Model
Artifact$_1$ Model
Artifact$_2$ Model
Artifact$_n$ Model
Artifact$_m$ Model
Generic model
Use
Input(s)
Output(s)



- Goal: model the target platform (tools, configurations)

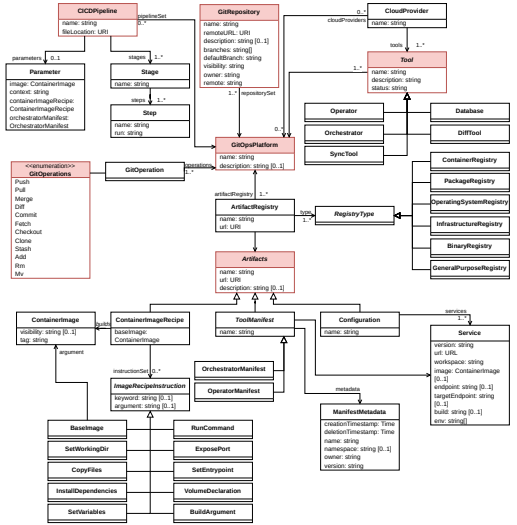
  → Different platform models have been proposed (e.g., Colantoni et al.)



Platform Model 12 / 18
• Goal: model the target platform (tools, configurations)
→ Different platform models have been proposed (e.g., Colantoni et al.)

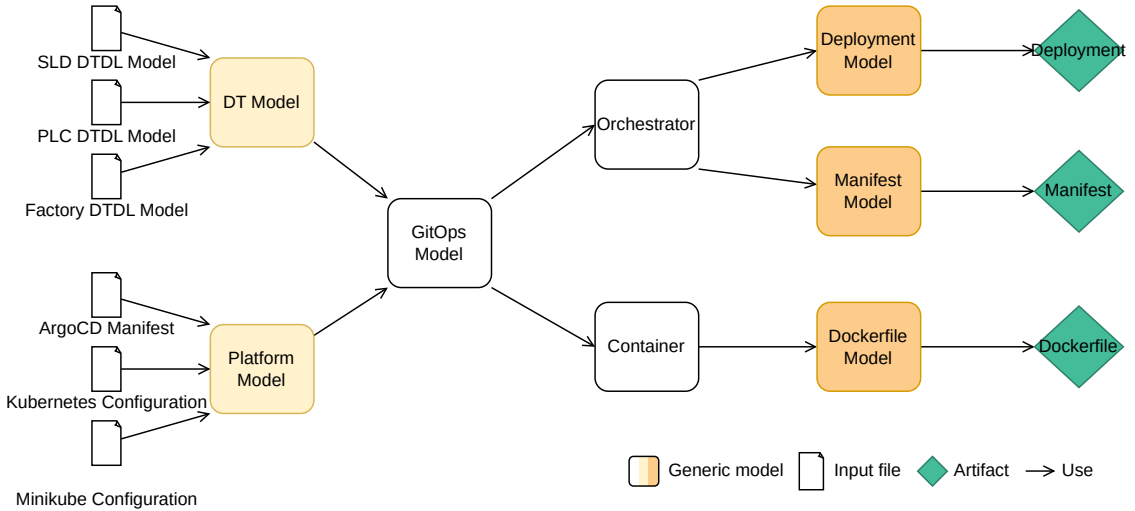
Key classes:

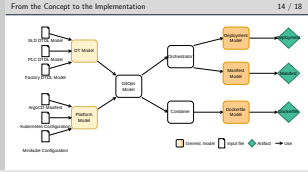
- `GitOpsPlatform`
- `GitRepository`
- `CICDPipeline`
- `GitOperations`
- `Tools`
- `Artifacts`



1. The goal here is not to show the entire graph, it is to present the main classes

DevOps Model 13 / 18
Key classes:
- GitOpsPlatform
- GitRepository
- CICDPipeline
- GitOperations
- Tools
- Artifacts


SLD DTDL Model
PLC DTDL Model
Factory DTDL Model
DT Model
ArgoCD Manifest
Kubernetes Configuration
Platform Model
Minikube Configuration
GitOps Model
Orchestrator
Container
Deployment Model
Manifest Model
Dockerfile Model
Deployment
Manifest
Dockerfile
Generic model
Input file
Artifact
Use

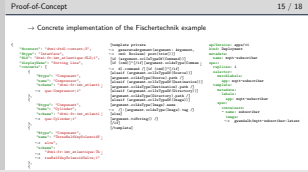


→ Concrete implementation of the Fischertechnik example

```
{
    "@context": "dtmi:dtdl:context;3",
    "@type": "Interface",
    "@id": "dtmi:fr:imt_atlantique:SLD;1",
    "displayName": "Sorting Line",
    "contents": [
        {
            "@type": "Component",
            "name": "Compressor",
            "schema": "dtmi:fr:imt_atlanti
            ↪  que:Compressor;1"
        },
        {
            "@type": "Component",
            "name": "Cylinder",
            "schema": "dtmi:fr:imt_atlanti
            ↪  que:Cylinder;1"
        },
        {
            "@type": "Component",
            "name": "ThreeHalfWaySolenoidV
            ↪  alve",
            "schema":
            ↪  "dtmi:fr:imt_atlantique:Th
            ↪  reeHalfWaySolenoidValve;1"
        },
        {
```

```
[template private
↪  generateArgument(argument: Argument,
↪  cmd: Boolean) post(trim())]
[if (argument.oclIsTypeOf(Command))]
[if (cmd)]"[/if][argument.oclAsType(Comman
↪  d).command /][if (cmd)]"[/if]
[elseif (argument.oclIsTypeOf(Source))]
[argument.oclAsType(Source).path /]
[elseif (argument.oclIsTypeOf(Destination))]
[argument.oclAsType(Destination).path /]
[elseif (argument.oclIsTypeOf(Directory))]
[argument.oclAsType(Directory).path /]
[elseif (argument.oclIsTypeOf(Image))]
[argument.oclAsType(Image).name
↪  /]:[argument.oclAsType(Image).tag /]
[else]
[argument.toString() /]
[/if]
[/template]
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mqtt-subscriber
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mqtt-subscriber
  template:
    metadata:
      labels:
        app: mqtt-subscriber
    spec:
      containers:
      - name: subscriber
        image:
        ↪  gwendalb/mqtt-subscriber:latest
```

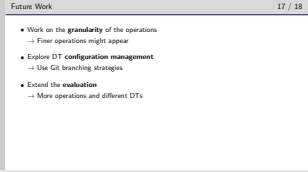



- **Semi-automatic approach**
  → A few variables require human intervention
- **Completeness of configuration knowledge**
  → Extend the DTDL language to add missing properties

Discussion 16 / 18
• Semi-automatic approach
→ A few variables require human intervention
• Completeness of configuration knowledge
→ Extend the DTDL language to add missing properties

- Work on the **granularity** of the operations
  → Finer operations might appear
- Explore DT **configuration management**
  → Use Git branching strategies
- Extend the **evaluation**
  → More operations and different DTs

- **Granularity** of the operations
  → Finer operations might appear
- DT **configuration management**
  → Use Git branching strategies
- Extend the **evaluation**
  → More operations and different DTs



Motivations

- Plenty of DTs exist
- Mostly built in *ad hoc* manners
  - → life cycle management issues
  - → re-usability & modularity issues

Our solution

- Operational life cycle of DTs
- Approach to automatically generate GitOps artifacts based on generic models
- Experimented in a PoC using a Training Factory