Towards Ontological Service-Driven Engineering of Digital Twins

B. Oakes, C. Gomes, E. Kamburjan, G. Abbiati, E.E. Bas, S. Engelsgaard

September 24th, 2024

POLYTECHNIQUE MONTRÉAL TECHNOLOGICAL UNIVERSITY

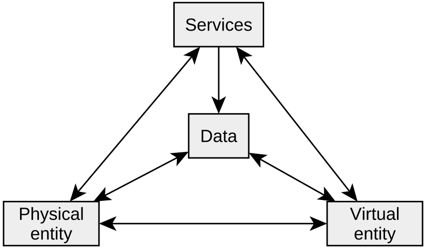
AARHUS UNIVERSITY

R&D TEST SYSTEMS

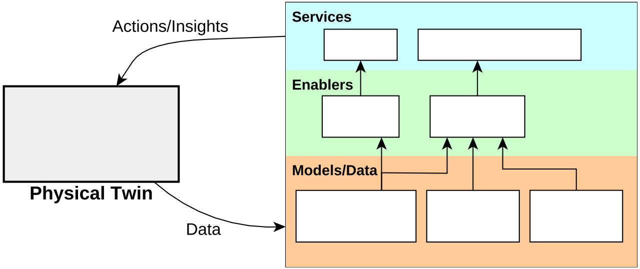
UNIVERSITY OF OSLO

LORC

EUDP



5D DT model, Tao et al. (2018)



DT Constellation

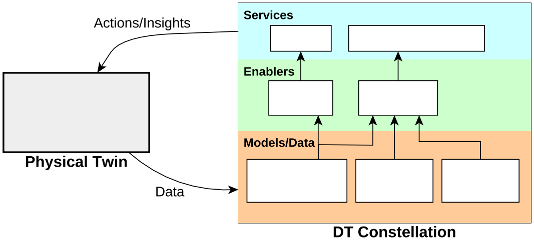
Toward a systematic reporting framework for Digital Twins: a cooperative robotics case study

Santiago Gil, Bentley J Oakes, Cláudio Gomes, Mirgita Frasheri and Peter G Larsen

18 essential DT characteristics to report
Services, automatic actions, fidelity/validity, time-scale, evolution, hosting/deployment, etc

Multiple approaches to start at bottom layer
Instead, leverage DTs as a constellation

- **Pick desired service**, then **recommend components**
- **Guide** users in selection, modelling, deployment
- Focus on **non-software eng. experts**

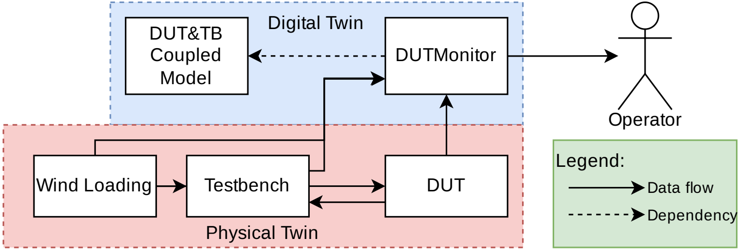


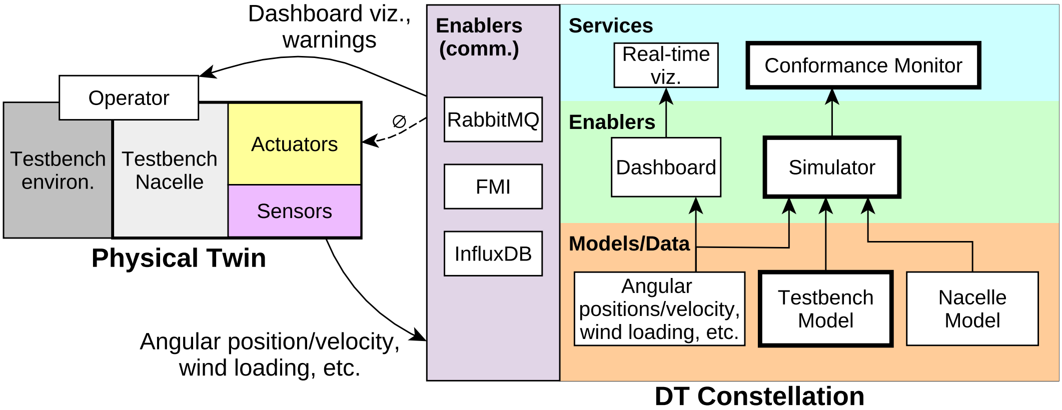
Work-in-progress / vision
Exploratory application

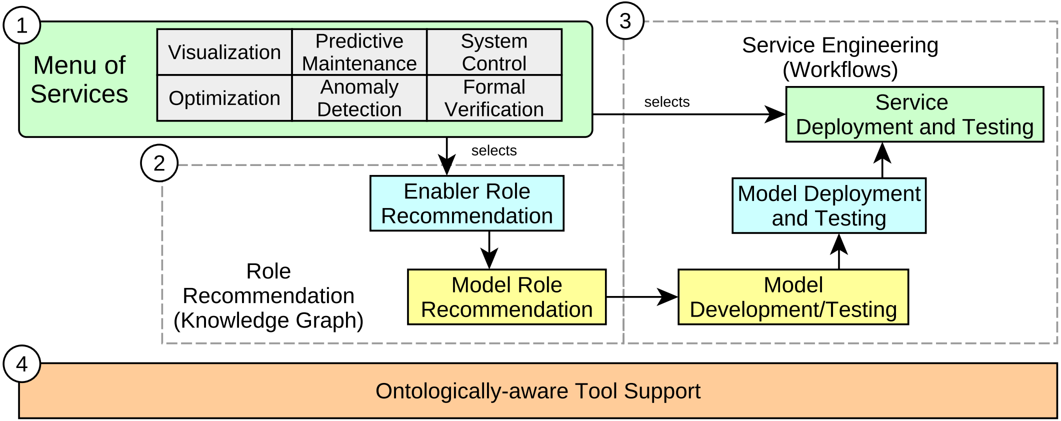
Wind Turbine Testing

- With Aarhus U., U. of Olso, R&D Test Systems, LORC
- Representative testbench for bending/torquing turbine nacelle

Problem: Detect mismatches between dynamics model and testbench, as failure can cause structural damage




Dashboard viz.,
warnings
Operator
Testbench
environ.
Testbench
Nacelle
Actuators
Sensors
Physical Twin
∅
Angular position/velocity,
wind loading, etc.
Enablers
(comm.)
RabbitMQ
FMI
InfluxDB
Services
Real-time
viz.
Conformance Monitor
Enablers
Dashboard
Simulator
Models/Data
Angular
positions/velocity,
wind loading, etc.
Testbench
Model
Nacelle
Model
DT Constellation


1
Menu of Services
Visualization
Predictive Maintenance
System Control
Optimization
Anomaly Detection
Formal Verification
selects
3
Service Engineering (Workflows)
Service Deployment and Testing
selects
2
Enabler Role Recommendation
Model Deployment and Testing
Role Recommendation (Knowledge Graph)
Model Role Recommendation
Model Development/Testing
4
Ontologically-aware Tool Support

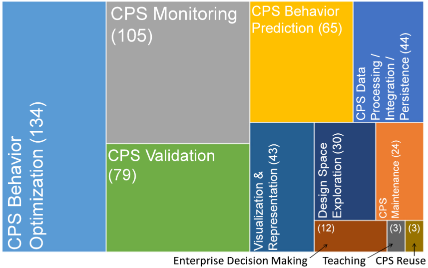
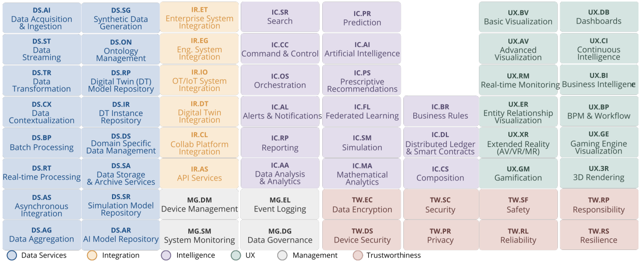


Figure 6, Dalibor et al. 2022 JSS



DT Consortium, DT Capabilities Periodic Table

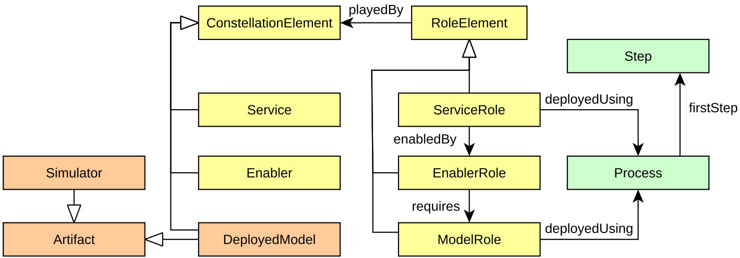
1

Menu of Services

Visualization	Predictive Maintenance	System Control
Optimization	Anomaly Detection	Formal Verification

Multiple ontologies in OML, built in openCAESAR

- This slide is a selection of the ontologies
- Total concepts: 42, Total relations: 25

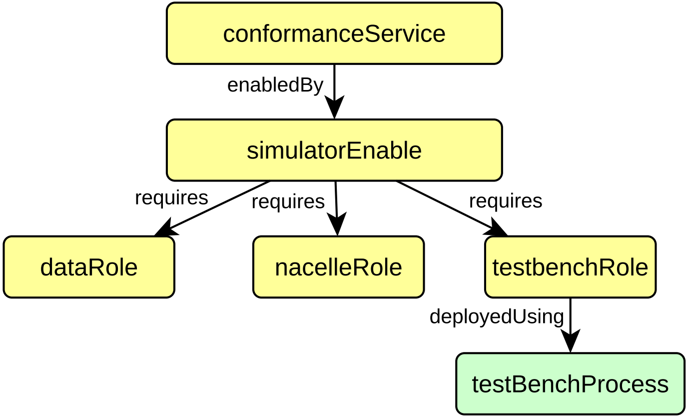


Ontology concepts:

1. Specific: `Artifact`, `Simulator`, `ODEModel`, `DeployedModel`
2. DT/Roles: `System`, `ConstellationElement`, `ServiceRole`
3. Process/notebook: `Process`, `Step`, `DeployStep`, `ProduceArtifact`, `Decision`

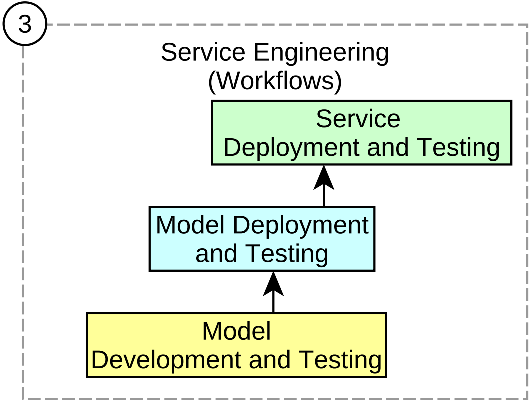
Step 2) Role Recommendation

- From DT ontology, store recommendations in knowledge graph
- Recommend roles for the service to be found or created
- Connect to workflows (next step)

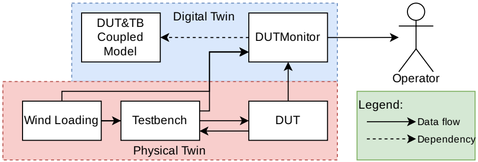


- Can query knowledge graph to find roles not yet filled
- Further work: Connect to repository, additional role constraints (validity, fidelity)

Step 3) Service Engineering Workflows



- Workflows as guided steps for user to find/model/test/deploy components
- Modelled in ontology, enacted in Jupyter notebook
- Concepts: Steps, decisions, artifact production/consumption, simulation steps
  - Model management problems
- Coarse stages, because smaller stages are too rigid for users



Model Development/Deployment Workflow:

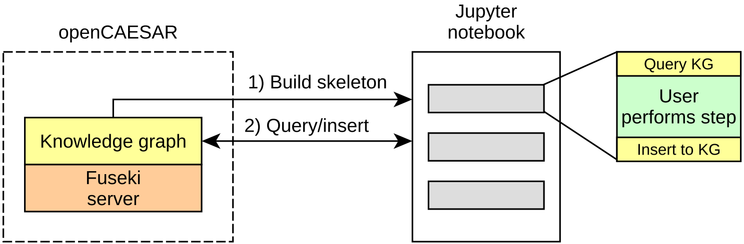
Step	Tests For
Decompose monolithic model	Decoupling error
Encapsulation as sub-models	Shared variables
Package models as FMUs	FMI support
Place FMUs in co-simulation	Co-sim orchestration

Service Development Workflow:

Step	Tests For
Inject fault to test monitor	Monitor
Package as service	Interfaces
Connect service with PT simulation	Real-time cap.
Connect service to PT	Full test

Deployment platform: Maestro (FMI), Docker and RabbitMQ

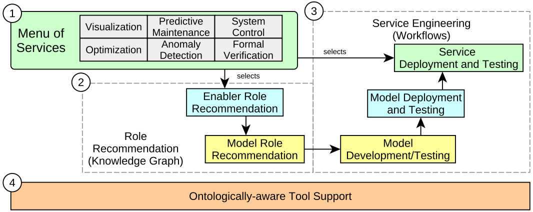
Step 4) Ontologically-aware Tooling



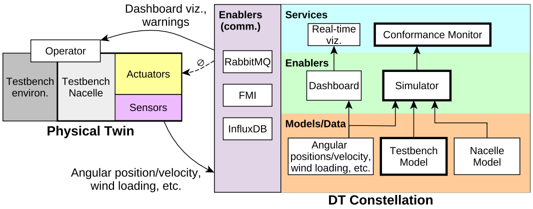
Load step insertion example:

INSERT { $freshIRI a spec:Step;	Create new step node
spec:previous $prev;	Connect to previous step
a spec:concretizes gen:LoadE2;	Relate to generic workflow
spec:loads $sim1;	Create new simulator node
spec:executed $now.	Time-stamp added to graph
$sim1 a spec:Simulator;	
spec:stored $path;	Simulator path
spec:concretizes gen:model2. }	Relate to generic workflow

Conclusion



Proposed approach to develop DT services



Application to conformance service

Takeaways:

- Opportunity to leverage DT characteristics for non-tech personnel
  - DT constellations and semantic richness of services
- Ontologies for heterogeneous/flexible/rich modelling and model management
- Role/workflow/notebook approach welcomed by practitioners
- Future work: Improve richness/tooling, validation

Towards Ontological Service-Driven Engineering of Digital Twins

B. Oakes, C. Gomes, E. Kamburjan, G. Abbiati, E.E. Bas, S. Engelsgaard

Appendix

DT Reporting Framework

C1: System-under-Study	C10: DT Models and Data
C2: Physical acting components	C11: Tooling and Enablers
C3: Physical sensing components	C12: DT constellation
C4: Physical-to-virtual interaction	C13: Twinning process and DT evolution
C5: Virtual-to-physical interaction	C14: Fidelity and validity considerations
C6: DT services	C15: DT technical connection
C7: Twinning time-scale	C16: DT hosting and deployment
C8: Multiplicities	C17: Insights and decision making
C9: Life-cycle stages	C18: Horizontal integration
C19: Data ownership and privacy	
C20: Standardization	
C21: Security and safety considerations	

Legend:

- Reqs/Concept/Design
- Realization
- Deployment
- Operation

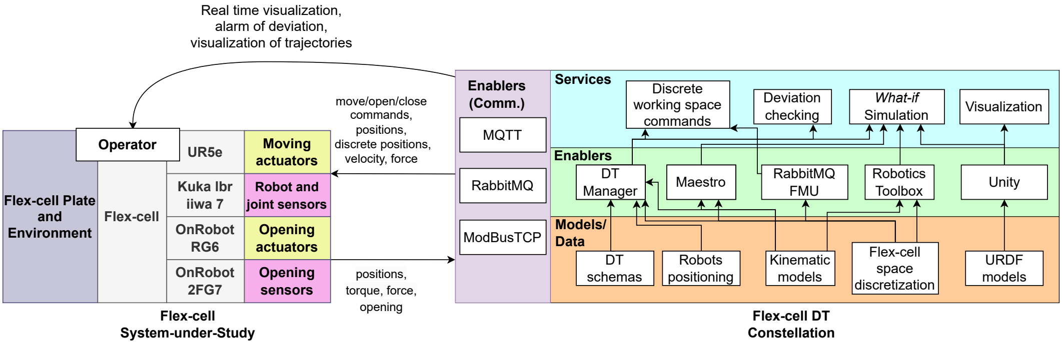
- Reporting framework for essential DT characteristics
- Merges Jones et al. (2020), Dalibor et al. (2022), Oakes et al. (2023)
- Reports robotics exemplar in detail, incubator/mobile robotics in summary
- Challenges/lessons learned from DT engineering and reporting

`Toward a systematic reporting framework for Digital Twins: a cooperative robotics case study, Gil et al., 2024, Simulation journal`

Desktop Robotti example

Table 7. Brief description of the Desktop Robotti DT case study with our proposed DT description framework.

Merged Characteristic	**Desktop Robotti case study**
MC1: System-under-Study	Small prototype of a field (agricultural) robot. A mobile robot.
MC2: Physical acting components	Motors for each wheel.
MC3: Physical sensing components	RPLidar A1, IMU (Inertial Management Unit), and wheel encoders.
MC6: Digital Twin Services	The Desktop Robotti DT provides services for *monitoring: distance-to-obstacle*, *collision avoidance for two cooperative Desktop Robottis*, *Parallel operation: comparing real and predicted location data*, *Fault-injection with hardware in the loop*, and *Runtime model swapping*: swapping FMUs during operation to extend functionality.
MC10: Digital Twin Models and Data	There are a kinematic model of the robot, specifically a bicycle model with the virtual wheels placed a the center of the front and rear axles, and an actuation model for the DC motors expressed as a first-order system. The data of interest are related to robot positioning and velocity.
MC11: Tooling and Enablers	RabbitMQ and the Robot Operating System (ROS)[67] for communication and interfacing. Maestro and RMQFMU to run the co-simulation scenarios. The Model Swap and Fault Injection plugins to run the DT services related to fault injection[68] and runtime model swapping[69]. RViz for visualization.
MC13: Twinning Process and Digital Twin Evolution	The DT was engineered based on an existing prototype of a large-scale agricultural robot (Robotti[70]) with a subsequent engineering approach. The evolution presents five milestones: the setup of the parallel operation, enhancement with fault-injection, time discrepancy detection (between real and simulated/DT time), runtime model-swapping, collision zone detection for a fleet of DRs.


Real time visualization,
alarm of deviation,
visualization of trajectories
move/open/close
commands,
positions,
discrete positions,
velocity, force
positions,
torque, force,
opening
Operator
Flex-cell Plate
and
Environment
Flex-cell
UR5e
Moving
actuators
Kuka lbr
iiwa 7
Robot and
joint sensors
OnRobot
RG6
Opening
actuators
OnRobot
2FG7
Opening
sensors
Flex-cell
System-under-Study
Enablers
(Comm.)
MQTT
RabbitMQ
ModBusTCP
Services
Discrete
working space
commands
Deviation
checking
What-if
Simulation
Visualization
Enablers
DT
Manager
Maestro
RabbitMQ
FMU
Robotics
Toolbox
Unity
Models/
Data
DT
schemas
Robots
positioning
Kinematic
models
Flex-cell
space
discretization
URDF
models
Flex-cell DT
Constellation

Elaasar et al (2023, October). openCAESAR: Balancing agility and rigor in model-based systems engineering. In 2023 MODELS-C (pp. 221-230). IEEE.

Ontological Modelling Language (OML):

- Essentially a DSL over OWL, removes accidental complexity
- Vocabulary (like meta-model) and descriptions (like model)
- Consistency checking and inference rules, *a-posteriori typing*
- Text-based for version control, federation of ontologies
- Trick: Closes world for analysis, becomes specification model

```
concept Mission :> base:IdentifiedElement
concept Objective :> base:IdentifiedElement
relation entity Pursues [
    from Mission
    to Objective
    forward pursues
    reverse isPursuedBy
    asymmetric
    irreflexive
]
```

openCAESAR:

- OML editor (Rosetta)
- Fuseki server for knowledge graph