Digital Twin software infrastructure – practical approach

Using *ModelConductor* – an open source digital twin framework

Wapice | CREATING A SMARTER FUTURE TODAY

Panu Aho

Disclaimer

- The presented work has been carried out in scope of a master’s thesis for Turku University of Applied Sciences while the author was employed there as a research engineer
- ModelConductor framework source code and the thesis are distributed as open source:
    - Master’s thesis: https://trepo.tuni.fi/handle/10024/118591
    - Source code (MIT License): https://github.com/donkkis/modelconductor

What is a digital twin?

› *"A **Digital Twin** is an integrated multiphysics, multiscale, probabilistic **simulation of an as-built vehicle or system** that uses the best available physical models, sensor updates, fleet history, etc., to **mirror the life of its corresponding flying twin**."* [1]

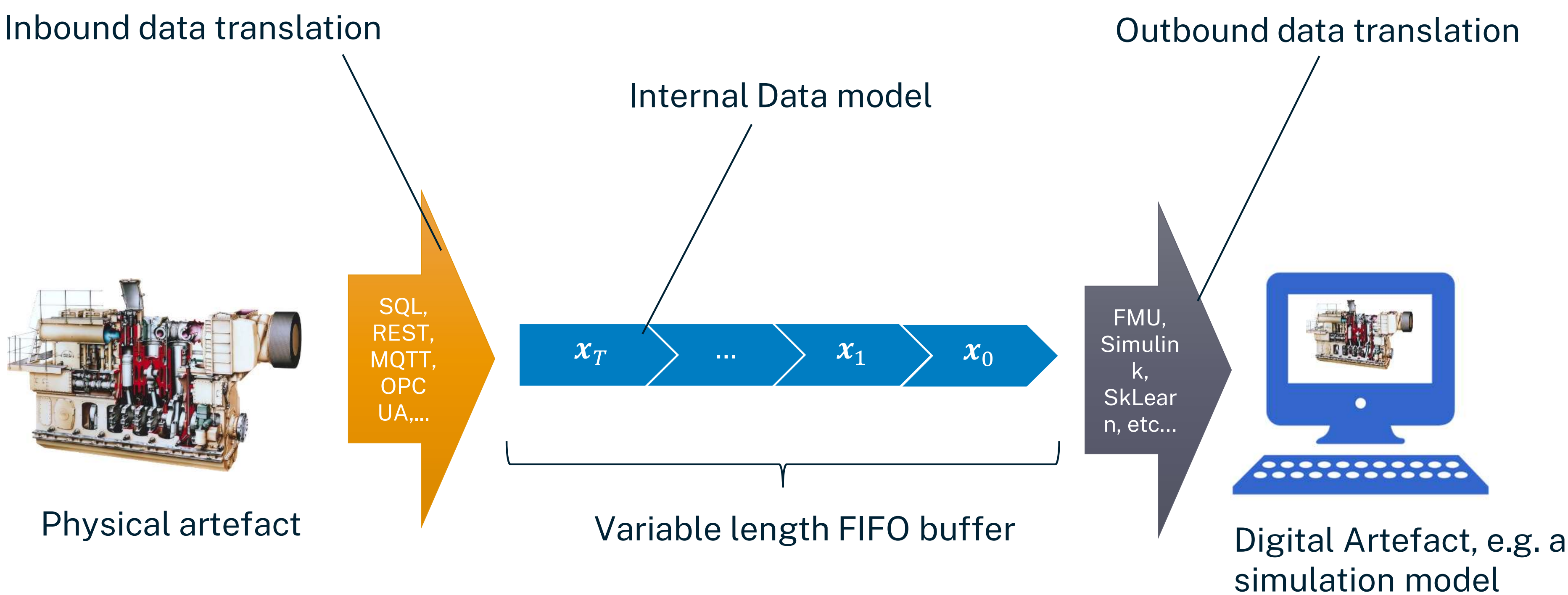
[1] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," in 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference - Special Session on the Digital Twin, 2012, pp. 1–14.

… In other words …

- Digital Twins (DT's) are a particular class of simulation artefacts with characteristics of:
  - **Representativity**: Each DT is a *digital artefact* that has a corresponding paired *physical artefact*. The digital counterpart is expected to be representative of the physical one
  - **Inspectability**: At any time, any attribute entailed in the DT should be easily accessible.
  - **Contemporality**: Active efforts should be taken by the DT itself and/or an encapsulating middleware layer to reflect changes during lifetime

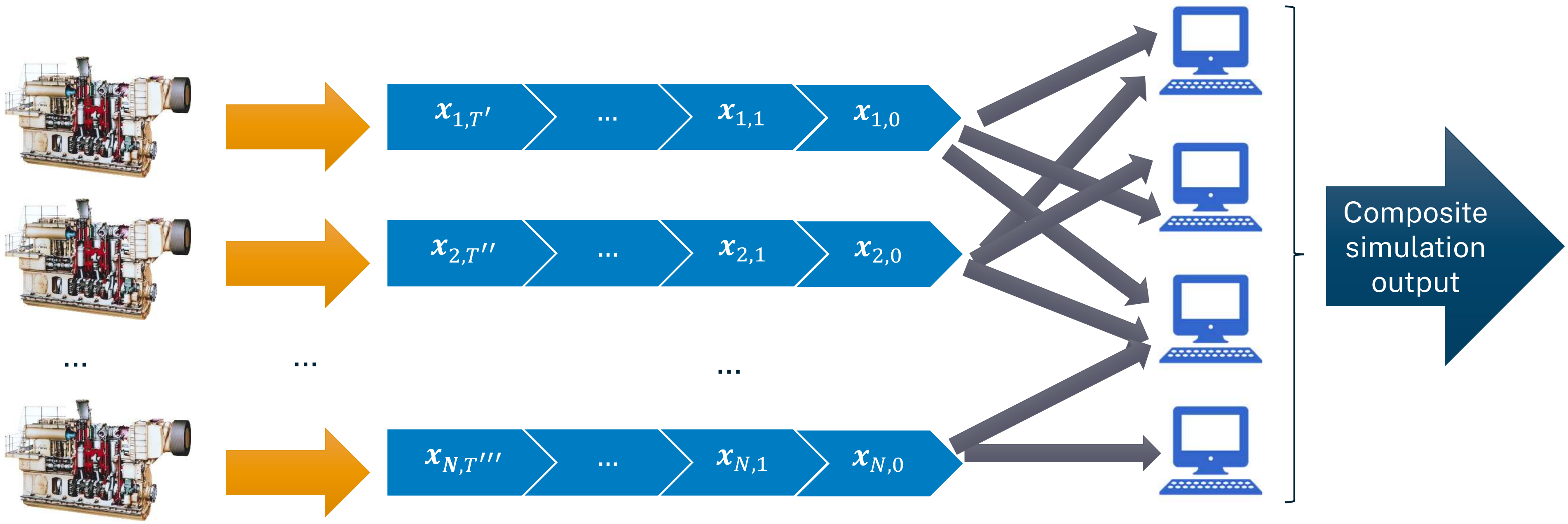
ModelConductor approach

The simplest use case – one-to-one mapping



ModelConductor approach

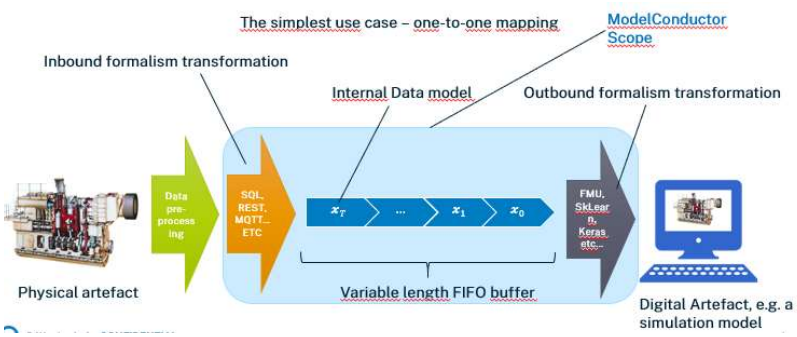
Can be extended to support arbitrarily complex many-to-many scenarios
- hence the postfix ”Conductor”



ModelConductor high level architecture

- The generic workflow translated to object oriented sofware modules
- Abstract classes can be extended to support an arbitrary number of formalisms

Requirements: User stories



Features: software structure and functionality

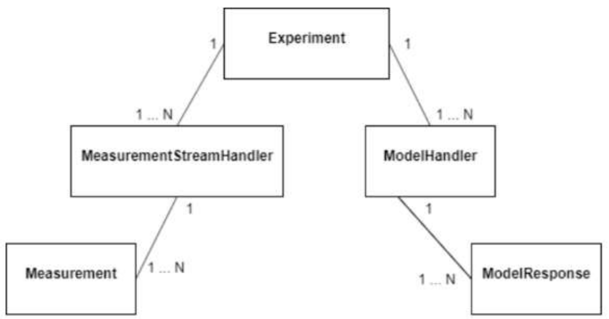
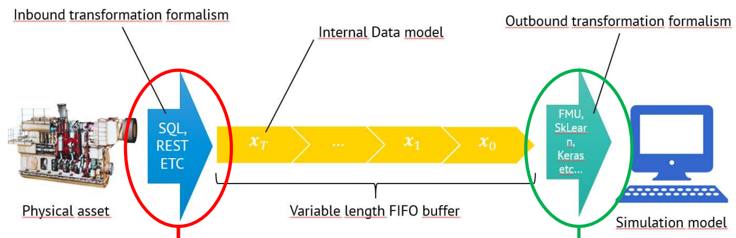


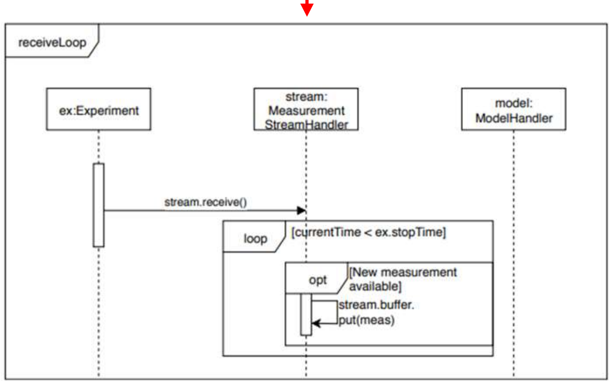
Figure 13. *Specification-level class diagram of the ModelConductor library.*

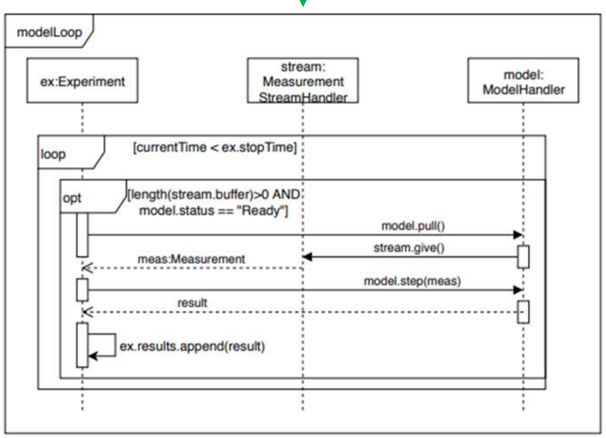
Asynchronous publish/consume mechanism

The buffer(s) are populated and emptied dynamically by threaded execution of respective loops

The `receive` and `step` methods are extended from a base class for each formalism (e.g. REST API, SQL query...)







Case: Turku University of Applied Sciences - Internal Combustion Engine Laboratory

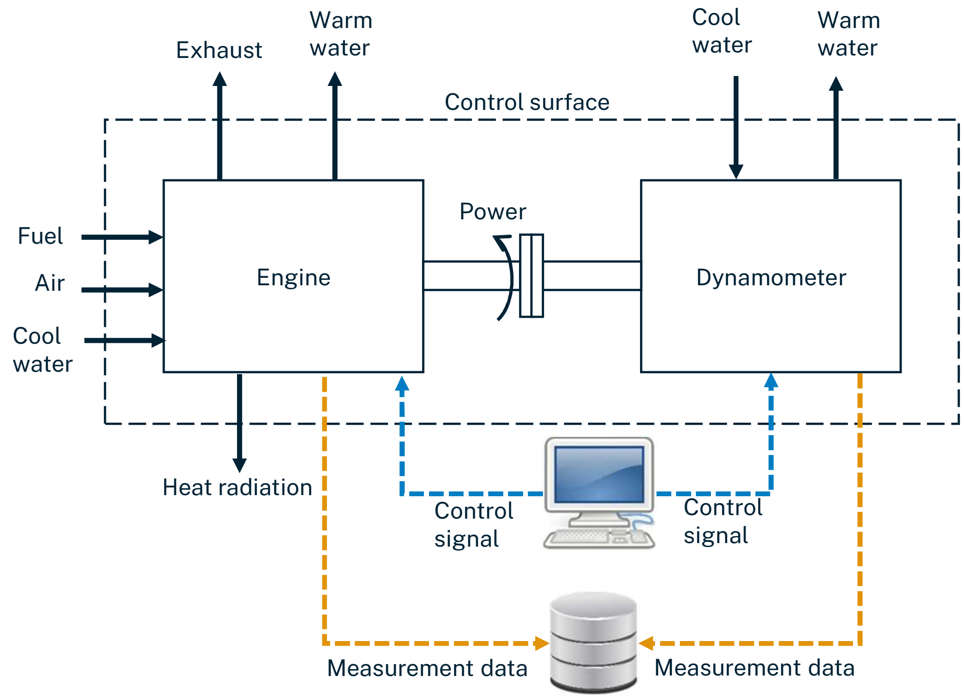
Building a simple machine learning based digital twin with ModelConductor

TUAS Internal Combustion Engine Laboratory

- Experimental internal combustion engine studies are routinely carried out at TUAS ICEL
- Test engines are fitted to dynamometers by research personnell
- The necessary instrumentation is installed to measure the e.g. the performance and emission values of the engine

Obtaining the control data

- Engines are driven in varying loading cycles, while registering operational data, e.g.:
  - Engine speed and torque
  - Air intake and fuel consumption rates.
  - Temperatures and pressures at various locations
  - Environmental pollutant's concentrations in exhaust gases
  - Various attributes regarding engine auxiliaries such as the turbocharger
  - Attributes read directly from the Engine Control Unit (ECU) via CAN bus
  - Atmospheric conditions



Example setup: Diesel engine real-time NO_x estimation

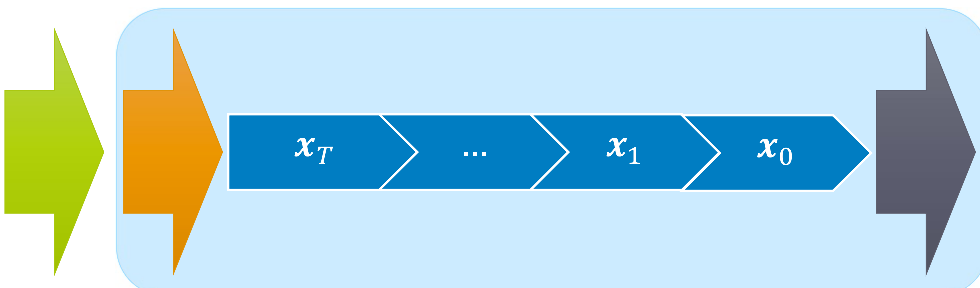
Table 1. Key characteristics of the target engine for simulation

Attribute	Description
Engine type	In-line, 4-cylinder, 4-stroke, turbocharged & intercooled
Displacement (dm^3)	4.9
Bore (mm)	108
Stroke (mm)	134
Max. Torque (Nm)	860
Rated power (kW)	148
Fuel type and fuel injection system type	Diesel, common rail

Table 2. Results from training various ML models for the NOx prediction problem

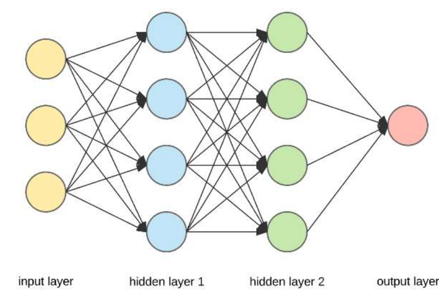
Model type	Training Error[6] (MAE)	Test Error[7] (MAE)
Linear regression	93.7	96.5
Polynomial regression (2nd order)	35.6	43.5
Random forest (100 trees, maximum depth 25)	4.6	12.1

A 4.9 dm^3 diesel tractor engine



$$\min_f |y - \hat{y}| \, , \hat{y} = f(\boldsymbol{x}_T)$$

$$\boldsymbol{x}_T \in \mathbb{R}^{157}$$



A machine learning model to predict engine-out NO_x emissons based on engine operation parameters

Experimental results

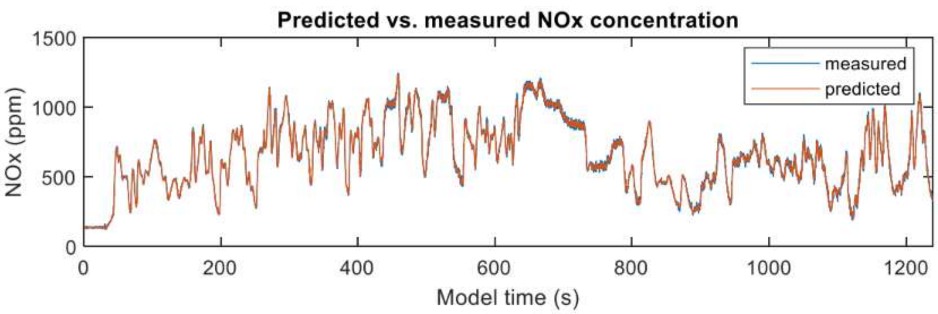


Figure 25. *Simulation results*

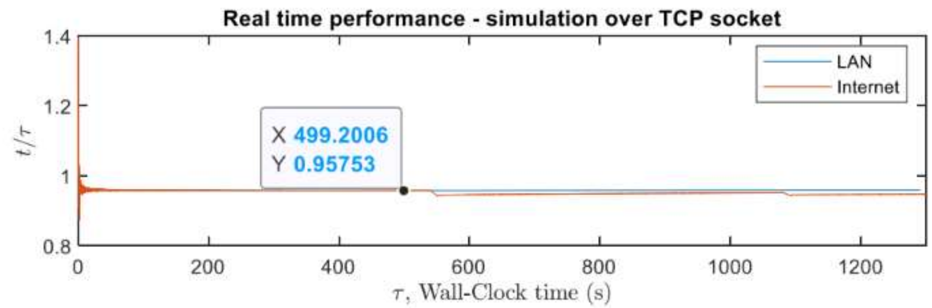


Figure 29. *Real time performance when streaming input data over TCP socket from a remote client*

More information:

- Master’s thesis
  - https://trepo.tuni.fi/handle/10024/118591
- Source code
  - https://github.com/donkkis/modelconductor
- The work presented has been funded by E3Power project (Business Finland)
  - http://e3power.fi/

CREATING A SMARTER FUTURE TODAY

Panu Aho
Data science consultant
+358 50 4129 150
panu.aho@wapice.com