Using UML and OCL Models to realize High-Level Digital Twins

Paula Muñoz
Universidad de Málaga, Spain

Javier Troya
Universidad de Málaga, Spain

Antonio Vallecillo
Universidad de Málaga, Spain
A Digital Twin is a comprehensive digital representation of an actual system, service or product (the Physical Twin), synchronized at a specified frequency and fidelity [1].

Our approach

Digital Twins are **pretty complex software** systems since they need to emulate the actual physical system faithfully.

Raise the level of abstraction using **software models** during their development.

Digital Twin (DT) -- Data -- Physical Twin (PT)

Control
Running example: a Lego Mindstorms Car

Ultrasonic sensor

Pose-provider

Touch sensor

Light Sensor
A framework for defining and deploying DTs

moveForward(2)
rotate(45)
stop()

{ X: 10, Y: 20,
  angle: 20 }

{ distance: 3, lightValue: 20,
isPressed: 0 }
A framework for defining and deploying DTs

**Data Lake**
- It connects the PT and DT
- It implements the Blackboard architectural pattern

**Drivers**
- Transform the data into formats that each component understands

**Service components**
- Implement additional functionality for the system (dashboards, IA algorithms)

**Analysis components**
- Implement different type of tests on the physical entity, the twin or even on service components
Implementation of the framework: UML Model of the Car
Implementation of the framework: Snapshots

**CarSnapshot**
- twinId : String
- xPos : Real
- yPos : Real
- angle : Real
- speed : Real
- isMoving : Boolean
- distance : Integer
- bump : Boolean
- light : Integer
- action : Action
- processingQueue : Boolean

**Snapshot**
- timestamp : Integer
- executionId : String

**InputCarSnapshot**
- setValue(c : Car)

**OutputCarSnapshot**
- init(car : Car, now : Integer)
Implementation of the framework: Data Lake

```
HGETALL NXJCar:1627484055:1627484375
1) "twinId" 2) "NXJCar"
3) "bump" 4) "0"
5) "processingQueue" 6) "0"
7) "light" 8) "45"
9) "angle" 10) "-0.52"
11) "executionId" 12) "1627484055"
13) "yPos" 14) "-0"
15) "speedFactor" 16) "31.28"
17) "isMoving" 18) "0"
19) "xPos" 20) "0"
21) "action" 22) "Rotate"
23) "distance" 24) "40"
25) "timestamp" 26) "1627484375"
```
Implementation of the framework: Connections

Publish-subscribe service (Jedis) and a socket

Publish-subscribe service (Jedis) and a USE Plugin
Implementation of the framework: Example of Analysis component
In our contribution, we show how it is possible to use UML and OCL models for the specification of DTs to verify their expected behavior in the early stages of development.

Advantages
- The framework allows replacing the high-level models with lower-level implementations
- The DT model can be specified at the needed fidelity level depending on the type of analysis that we want to perform.
- It also allows to analyze and validate any part of the software independently

Future work
- Validate the proposal with further physical systems
- Create more analysis and services modules.
- Evaluate the framework performance under stressful conditions to determine its scalability and applicability to larger systems
Thank you for your attention

Using UML and OCL Models to realize High-Level Digital Twins

Paula Muñoz
Universidad de Málaga, Spain

Javier Troya
Universidad de Málaga, Spain

Antonio Vallecillo
Universidad de Málaga, Spain