Digital Twin – Modeling Interrelated Devices

Authors:

Juan Asenjo
IoT Architect Evangelist
Tata Consulting Services (TCS)
juan.asenjo@tcs.com

Dr. Chellury Sastry
North America IoT Consulting Practice Head
Tata Consulting Services (TCS)
chellury.sastry@tcs.com

OVERVIEW

Digital Twin, i.e. the real time representation of physical things/devices in a cloud environment, is at the core of Internet of Things (IoT) solutions. The digital twin of a thing/device could be as simple as a JavaScript Object Notation (JSON) document or as complex as a CAD simulated model that interact with the real time data.

In this article we focus on public cloud vendors such as Microsoft Azure®, Amazon AWS® and Google GCP® as they are big enablers for IoT applications because they provide cost effective store and compute infrastructure for batch and real time analytics.

Connected Devices and Connected Factory IoT solutions model real physical devices/things such as pumps, generators, vending machines, etc., to their corresponding digital twin representation.

The picture below shows the JSON representation of a physical device (a Power Generator Gen Set) through its digital twin equivalent in a prototypical Industrial Internet Reference Architecture (IIRA)[1] based IoT platform in the cloud.

Public cloud IoT platforms work very well for 'atomic' representation of things/devices but they lack the ability to model hierarchical relationships among the thing/devices, e.g., modeling a pump station that contains a pump and a tank device as digital twin in the IoT platform.

The challenges of modeling interrelated devices in digital twins includes:

- Modeling the relationship between system and components.
- Modeling the Inheritance between system components.
- Issues related to embodiment of system structure into digital twin.
- Current tools are inadequate due to incomplete knowledge transfer of requirements between OT and IT teams.
- Leverage existing semantic representation of plant floor devices (OPC UA Information Modeling, SensorML, and S95).
- Etc.

This article presents a unique approach on how these challenges could be addressed to generate the digital twin representation of a complex system, such as a factory.

[1] **IIRA**: The Industrial Internet Reference Architecture (IIRA) is a standards-based open architecture defined by the Industrial Internet Consortium (IIC). Its objective is to have broad applicability to drive interoperability, map applicable technologies and to guide technology and standards development. http://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf

MODELING COMPLEX DIGITAL TWIN RELATIONSHIPS

Let's work with a use case that illustrates how the digital twin plays a central role in an Industrial IoT (IIoT) solution implementation. The figure below depicts a simplified representation of a water pump station. We follow the same color scheme to indicate static, commands and reported telemetry values.

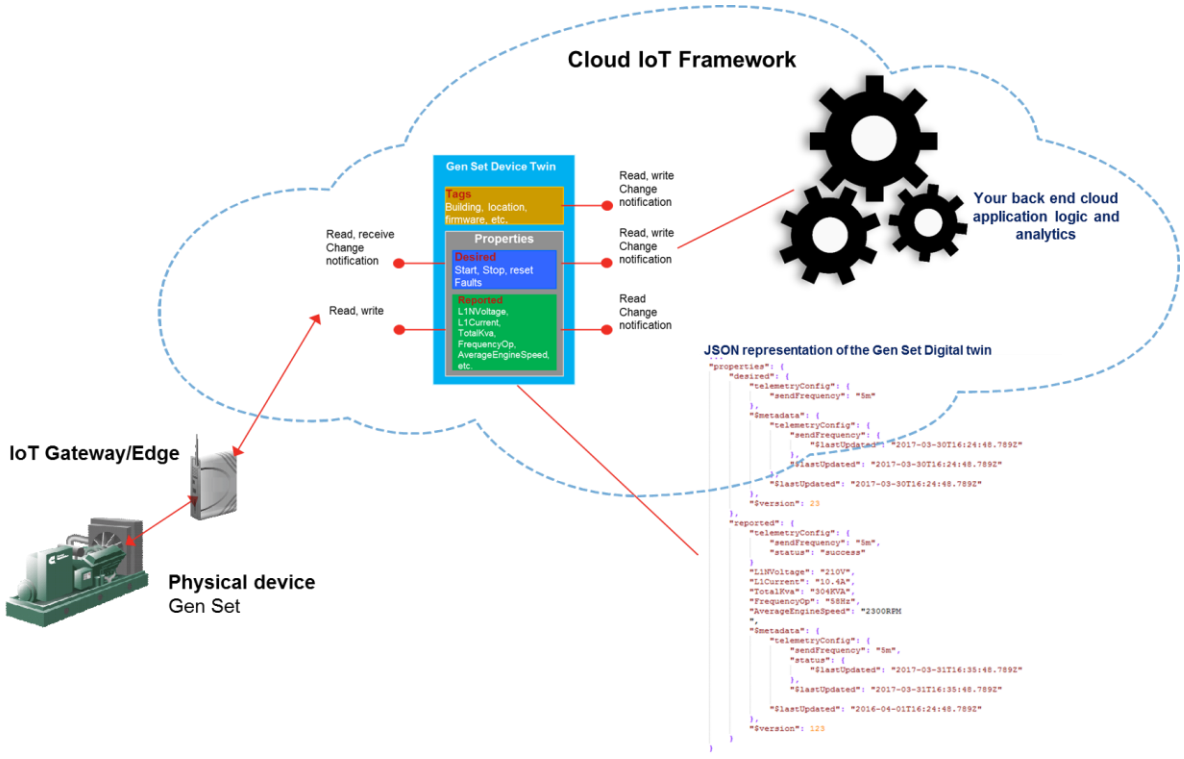

Figure 1 Gen Set Digital Twin

50 ft.	Static property that seldom changes (i.e., height of the tank)
36.2 ft.	Dynamic property (i.e., tank level)
Running	Command to device

From a device standpoint we distinguish at least two devices as part of the pump station:

Pump station device = Tank Device + Pump Device

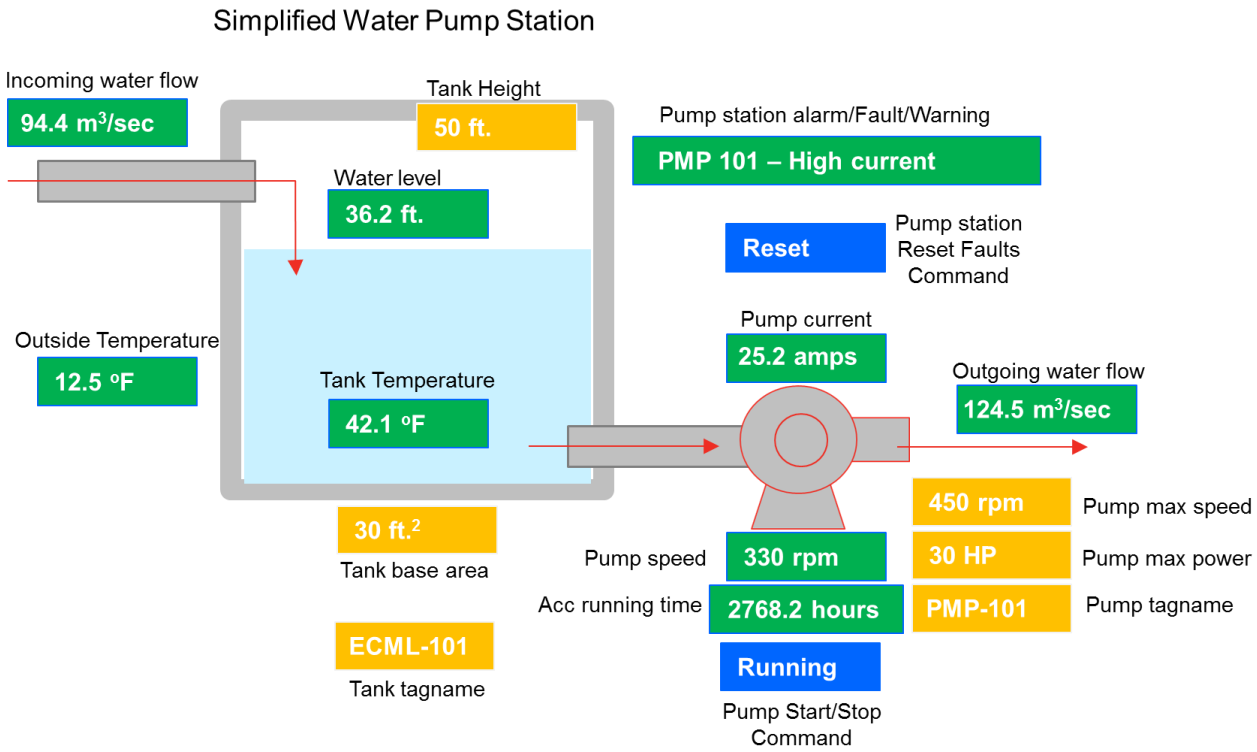

Figure 2 Modeling digital twin for a water pump station

Let's create the basic JSON representation for each device and then the pump station.

First, the JSON digital twin representation for the Tank device (Figure 3).

As can be seen, the JSON representation of the tank has some static properties (**tags**) that rarely change (name, location, tank height, etc.), commands (**desired**) none for the tank and some telemetry properties (**reported**) like outside temperature, water level, incoming water flow, etc. **GUID#1** is a generated unique ID that identifies this device.

```
{
    "deviceId": "GUID#1",
    "tags": {
        "TankTagname": "ECML-101",
        "TankHeight":  {50, ft},
        "TankBaseArea": {30, ft2},
        "deploymentLocation": {
            "GPS": "N40° 44.9064', W073° 59.0735'"}
    },
    "properties": {
        "desired": {
            "$version": 1
        },
        "reported": {
            "telemetryConfig": {
                "sendFrequency": "10s",
                "status": "success"
            }
            "$IncomingWaterFlow" : {94.4, mt3/sec}
            "$OutsideTemperature" : {12.5, oF}
            "$WaterLevel" : {36.2 ft}
            "$TankTemperature" : {42.1, oF}
            "$version": 4
        }
    }
}
```

Figure 3 Tank JSON Digital Twin model

We can derive other operational information such as:

- Available instantaneous water volume
- Capacity to support water flow using available water volume
- Model outgoing water flow based on historical data

Next the JSON digital twin representation for the pump device is:

```
{
    "deviceId": "GUID#2",
    "tags": {
        "PumpTagname": "PMP-101",
        "PumpMaxSpeed":  {450, rpm},
        "PumpMaxPower": {30, HP},
        "deploymentLocation": {
            "GPS": "N40° 44.9064', W073° 59.0735'"}
    },
    "properties": {
        "desired": {
            "$StartPump" : 1,
            "$StopPump" : 0,
            "$version": 1
        },
        "reported": {
            "telemetryConfig": {
                "sendFrequency": "10s",
                "status": "success"
            }
            "$PumpCurrent" : {25.2, amps}
            "$PumpSpeed" : {330, rpm}
            "$AccRunningTime" : {2768.2,  hours}
            "$PumpStatus" : {Running}
            "$version": 4
        }
    }
}
```

Figure 4 Pump JSON Digital Twin model

Similar to the tank device, this JSON model shows static properties (**tags**), telemetry data (**reported**) and Commands (**desired**). GUID#2 is a generated unique ID that identifies this device. Again we could derive operational information such as:

- Calculate if the pump can keep up with the incoming supply of water
- Predictive maintenance alerts for the pump using historical power, flow or accumulated run time, etc.

PROPOSED REPRESENTATION OF THE PUMP STATION

The Pump station Device is the sum (and more) of the Tank and Pump devices. For our discussion, we **propose** the following representation of the Pump station.

```
{
    "deviceId": "GUID#3",
    "tags": {
        "PumpStationTagname": "ECML-101",
        "deploymentLocation": {
            "GPS": "N40° 44.9064', W073° 59.0735'"}
    },
    "properties": {
        "desired": {
            "$ResetFaults" : 1

        },
        "reported": {
            "telemetryConfig": {
                "sendFrequency": "10s",
                "status": "success"
            }
            "$Alarm" : "PMP 101 - High current"
            "$version": 4
        }
        "ChildTwins": {
            "$Tank" : "GUID#1"
            "$Pump" : "GUID#2"
        }
    }
}
```

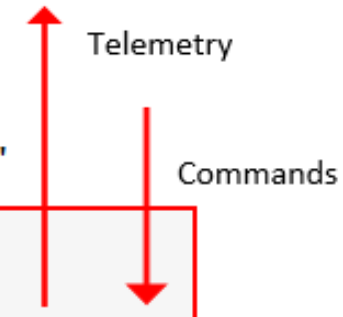

Figure 5 Pump Station JSON Digital Twin model

By using the unique identifiers of each child twin, we inserted "**child twins**" linking the corresponding tank and pump devices. This clearly models the physical relationships between these 3 devices. The Pump station representation is linked to any static or dynamic properties of the child devices and, conversely, the pump station device manages the commands to each child twin.

The above is the first basic step to model the complexity of the physical world through a digital twin representation. Next, we want to expand the model by adding "**logical rules"** that encompass the child twins. An example of this is shown in Figure 6.

```
{
        "deviceId": "GUID$3",
        "tags": {
            "PumpStationTagname": "ECML-101",
            "deploymentLocation": {
                "GPS": "N40° 44.9064', W073° 59.0735'"}
        },
        "properties": {
            "desired": {
                "$ResetFaults" : 1
            },
            "reported": {
                "telemetryConfig": {
                    "sendFrequency": "10s",
                    "status": "success"
                }
                "$Alarm" : "PMP 101 - High current"
                "$version": 4
            }
            "ChildTwins": {
                "$Tank" : "GUID$1"
                "$Pump" : "GUID$2"
            }
        }
        },
        "logicrules": {
            "checkrunningtime": {
                IF $Pump.AccRunningTime > 3000
                {
                    $Notification.Send "Pump" + $Pump.PumpTagname + "maintenance required"
                }
            "checktanklevel": {
                IF ($Tank.WaterLevel > 45 OR  $Tank.WaterLevel < 10 )
                {
                    $Notification.Send "Tank" + $Tank.TagTagname + "check tank level to avoid service"
                }

            }
        }
    }
```

Figure 6 Advanced Modeling of the pump station digital twin

The proposed model includes a digital twin representation that encapsulates the behavior of the physical components and their relationships. Typically this complex modeling requires proprietary cloud back end support beyond the basic IoT framework but we hope that in the near future cloud Platform as a Service (PaaS) infrastructure would support this natively.

Next we take a look at existing plant floor standards that already have semantics of the things/devices and how they could be leveraged to expedite the creation of digital twin representation for a factory.

PUTTING IT ALL TOGETHER

For Smart Factory applications, data comes from Programmable Logic Controllers (PLCs), drives, historians, etc. Historically, data collection took place using a list of 'atomic' data points or tags.

Standards such as OPC UA (OPC Connect)[2], SensorML[3], SSW (Semantic Sensor Web)[4], Sensor Grid[5], etc., were utilized, along with semantic information to capture the relationships between these devices on the plant floor. This is similar to the above example but much more complex – factories have thousands of data points with many hierarchical relationships between devices.

OPC UA's Information Modeling[6] capability is the closest standard that meets the requirement to model devices and their relationships and should be leveraged to generate the Semantic Ontology for the Digital Twin representation of Smart Connected Factories or Connected Devices.

In OPC UA, objects and relationships are represented as shown in Figure 7.

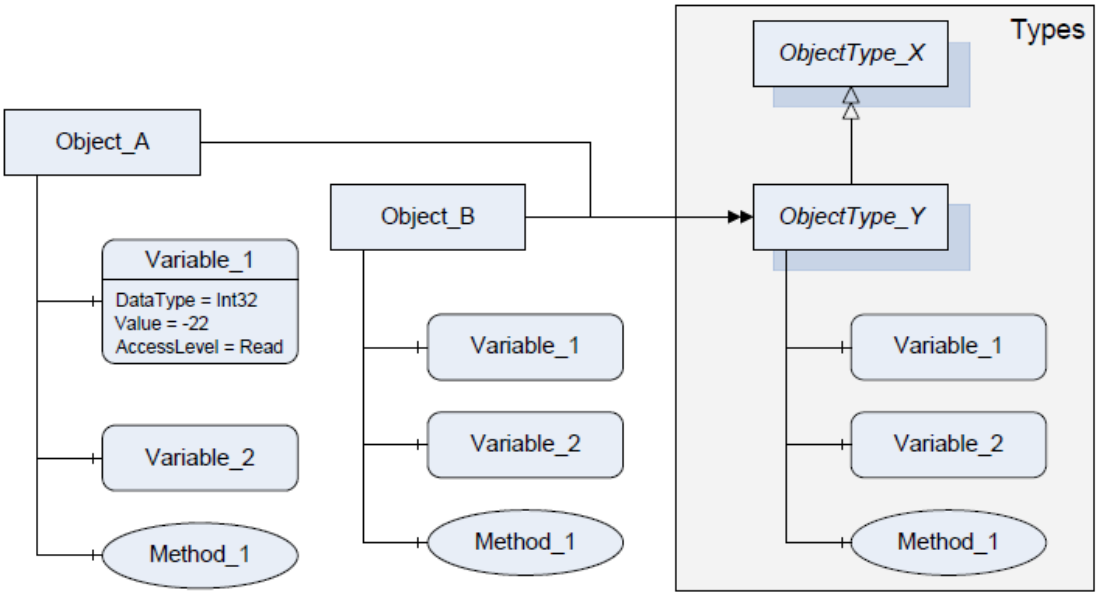

Figure 7 OPC UA representation of an object

This follows an Object Oriented approach: Objects can be composed of objects and properties; objects belong to types and you model the relationships between them.

Figure 8 shows an example of a temperature controller represented as *Device Object*. The component *ParameterSet* contains all *Variables* describing the *Device*. The component *Method Set* contains all *Methods* provide by the *Device*. Both components are inherited from the *TopologyElementType* which is the root *Object* type of the *Device Object* type hierarchy. Objects

[2] http://opcconnect.opcfoundation.org/2015/12/why-semantics-matter/

[3] http://www.sensorml.com/sensorML-2.0/examples/helloWorld.html

[4] https://en.wikipedia.org/wiki/Semantic_Sensor_Web

[5] https://en.wikipedia.org/wiki/Sensor_grid

[6] OPC UA Information Model https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model/

of the type *FunctionalGroupType* are used to group the *Parameters* and Methods of the Device into logical groups. The *FunctionalGroupType* and the grouping concept are defined in UA Part DI but the groups are Device type specific, i.e., the groups' ProcessData and Configuration are defined by the *TemperatureControllerType* in this example.

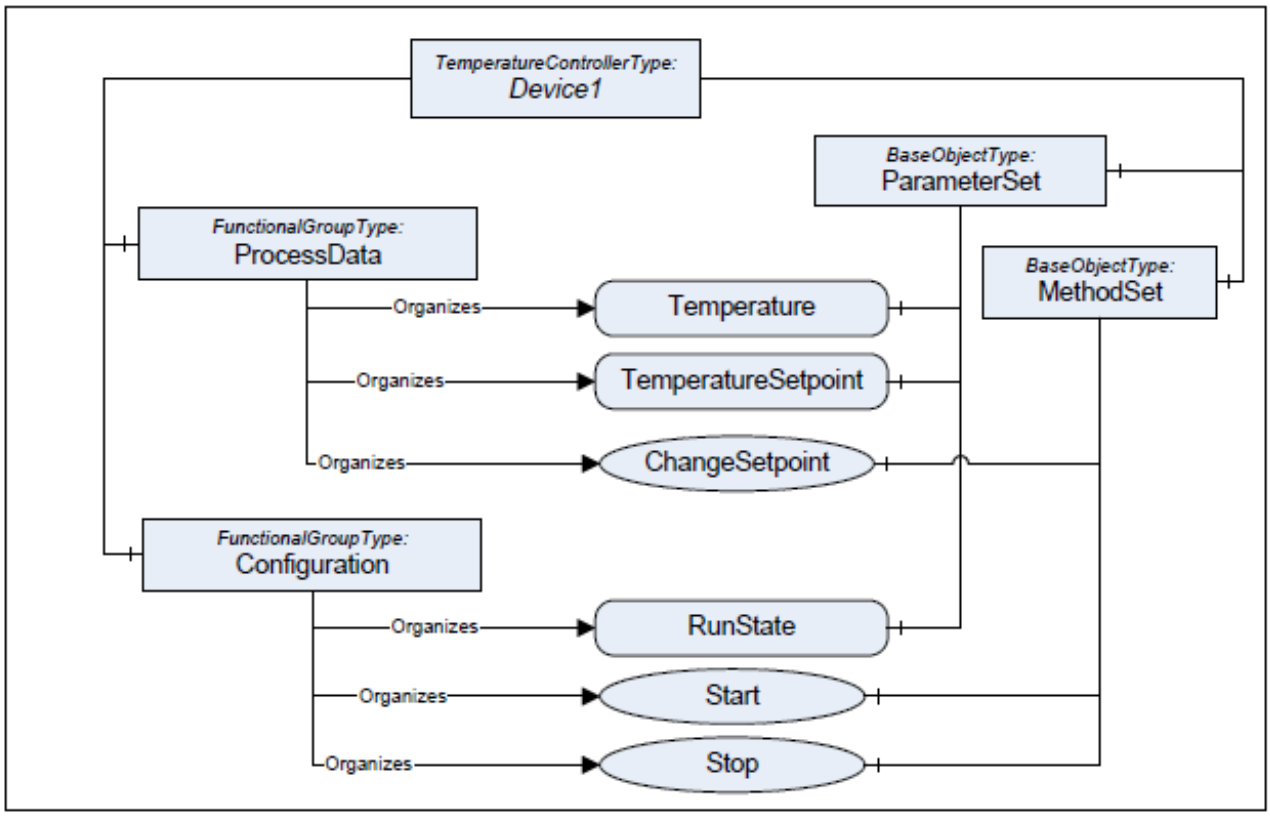

Figure 8 OPC UA Device Example

OPC UA is a well-established standard in the industry with a large installed base. This presents an opportunity to leverage the OPC UA Information Model to automatically generate the JSON representation of the corresponding IoT digital twins, avoiding tedious work for large scale systems.

Benefits include:

- Better fidelity of digital twin as is automatically generated from an existing model
- Lower level of effort; a translation tool will generate the cloud representation from the OPC UA model
- Faster implementation and time-to-operation, same as above
- Agility, flexibility, and reusability
- Scalability of model complexity, once supported by cloud IoT framework
- Wider market acceptance (technical and commercial)

To model large Smart Factory IoT implementations, new standards are needed to address:

- Extend the basic digital twin representations to include child twin hierarchies and a behavior modeled as logic rules. Cloud vendors would need to expand current frameworks to support these advanced models or provide comprehensive API to allow third party software companies to extend them.
- Leverage industry standards such as OPC UA Information Modeling to capture the complex relationship between devices and automatically generate the representation of the digital twin for existing installed base.

SUMMARY

- Cloud technologies have enabled a new era of possibilities for IoT solutions (low cost, high storage and compute, high throughput messaging, security, etc.).
- Digital Twin is the core of an IoT strategy for Connected Devices and Smart Factory.
- Most Cloud vendors already provide a platform for Digital Twin implementations (Amazon CoreGreenGrass[7], Microsoft Azure IoT Hub[8], Google IoT[9], etc.)
- Currently, cloud IoT framework only support atomic device representations relying on proprietary back end code to model complex systems.
- Plant floor standards like OPC UA, SensorML, S95, etc., already provide a logical model for large scale devices and their relationships.
- New standards are needed to port these plant floor models to the cloud IoT frameworks, thus accelerating the implementation of Smart Factory IoT solutions.
- Cloud frameworks should expand their IoT frameworks to expedite IoT solutions. Alternatively, third party software vendors can provide this extension of top of the cloud IoT frameworks.
- These new standards are not just to port existing plant floor standard but to embrace the enormous potential that public clouds bring to IoT.

[7] https://aws.amazon.com/greengrass/

[8] https://docs.microsoft.com/en-us/azure/architecture/guide/

[9] https://cloud.google.com/solutions/iot/

- Return to IIC Journal of Innovation landing page for more articles and past editions.