JÖNKÖPING UNIVERSITY
School of Engineering

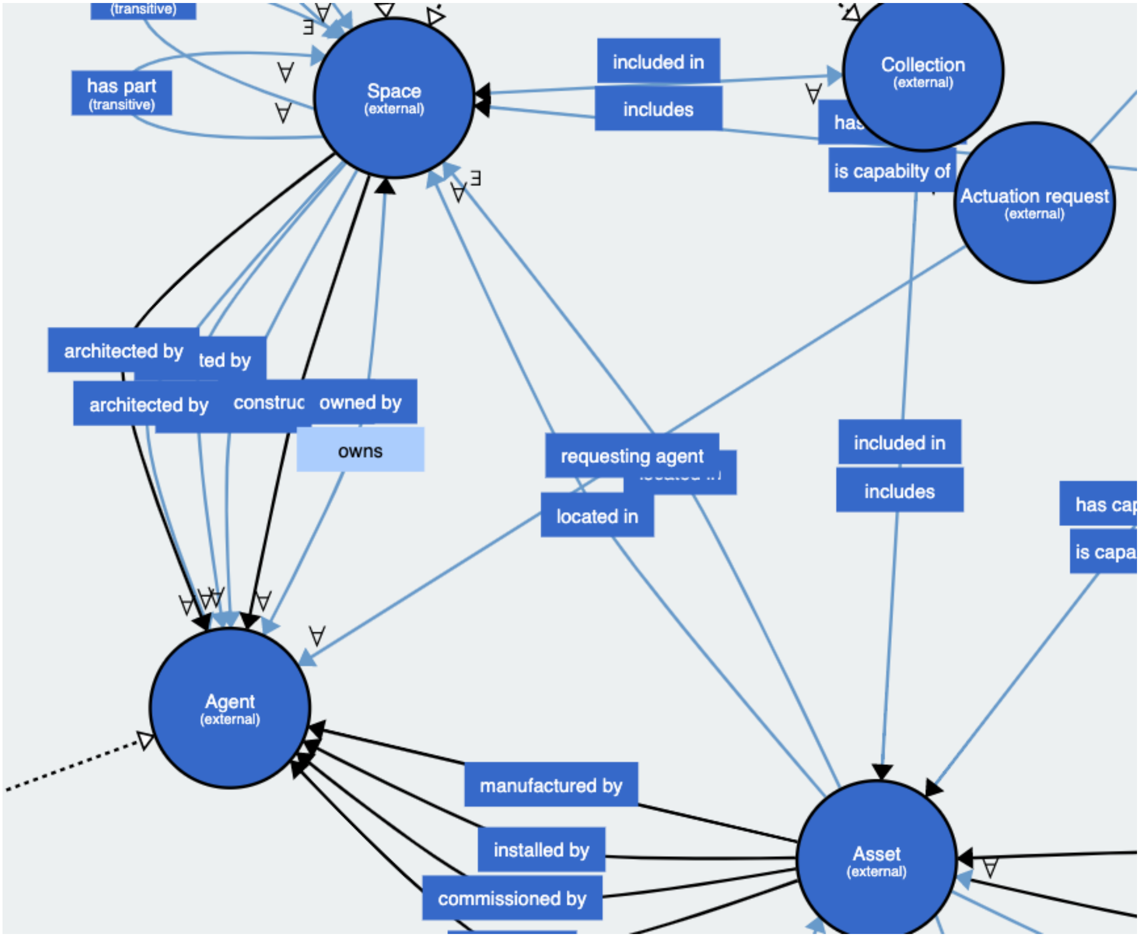
DTDL Ontology Use in Azure Digital Twins

Karl Hammar

2021-04-06

OVERVIEW

- The RealEstateCore Ontology Redux
- Microsoft Azure IoT Services (Digital Twins, IoT Hub, IoT Edge, etc.)
- DTDL – Digital Twin Definition Language
- OWL2DTDL
- JTH Smart Space Demonstrator

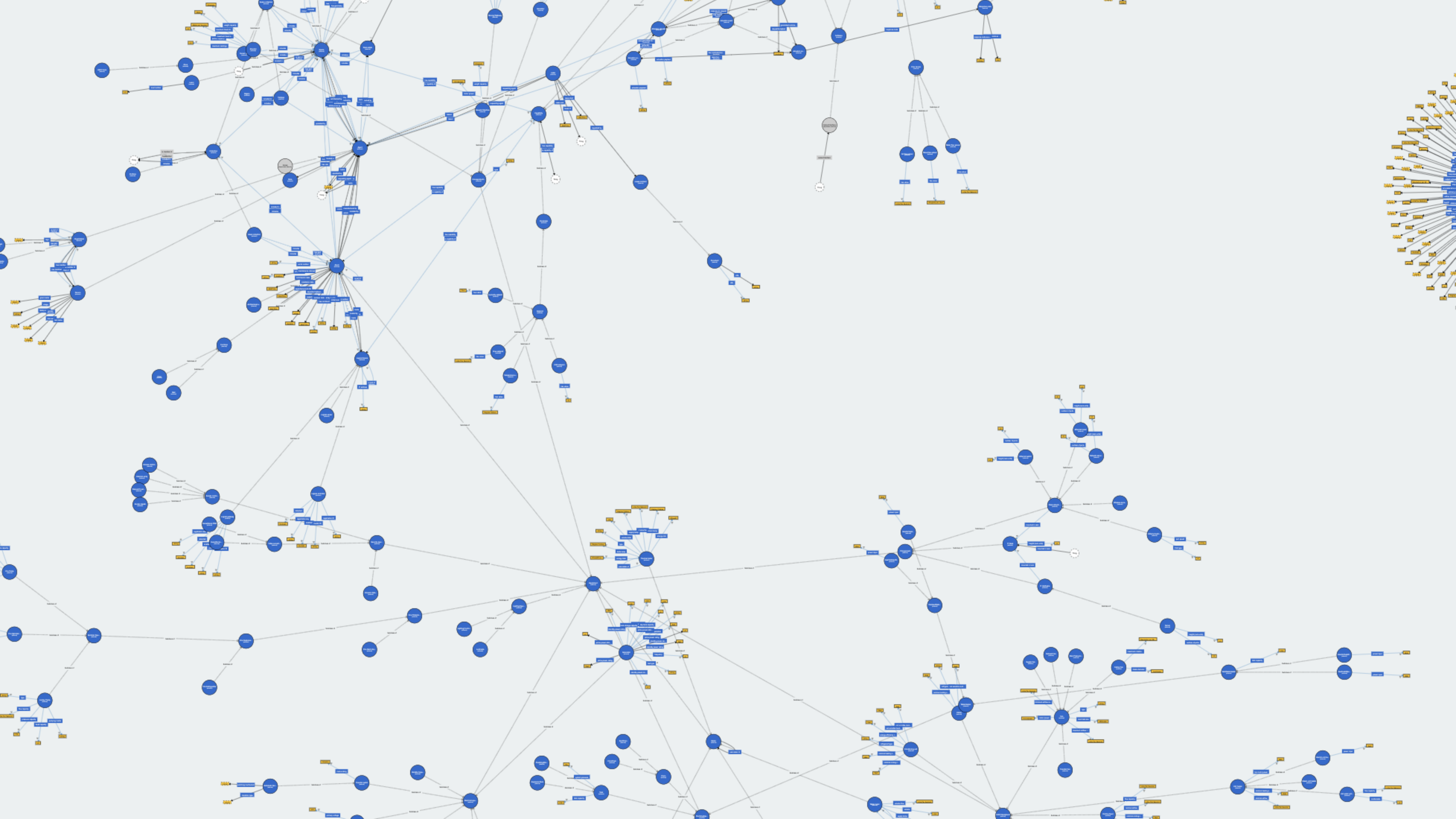


RealEstateCore

OWL ontology for the real estate domain

Describes concepts and relations in real estate:

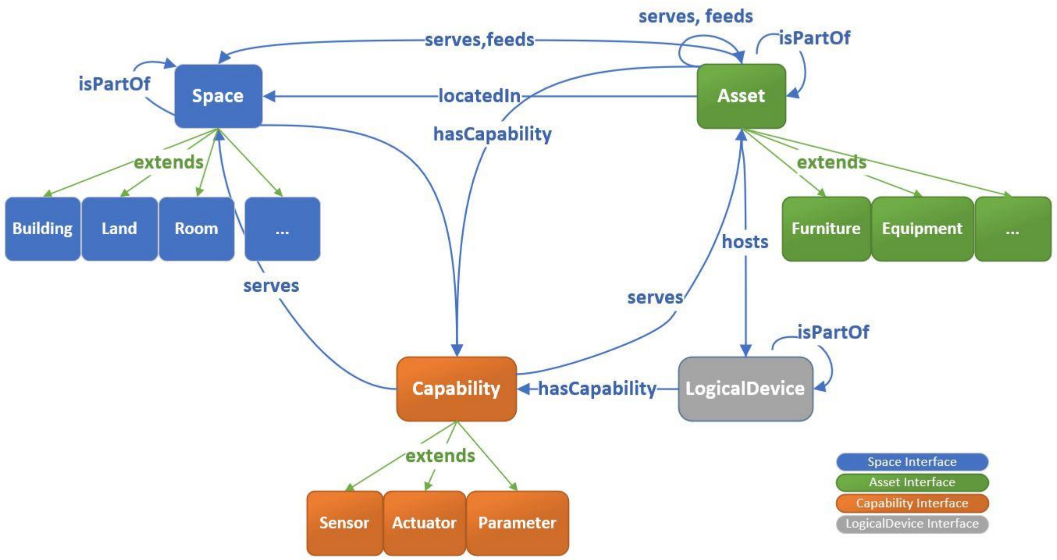
- BMS
- IoT
- BIM/CAD
- Admin/business



REC ALSO PROVIDES

- Light-weight edge message format
- OpenAPI spec for RESTful access to REC-compliant systems
- (Optional) certification process

REC REDUX (3.3?)

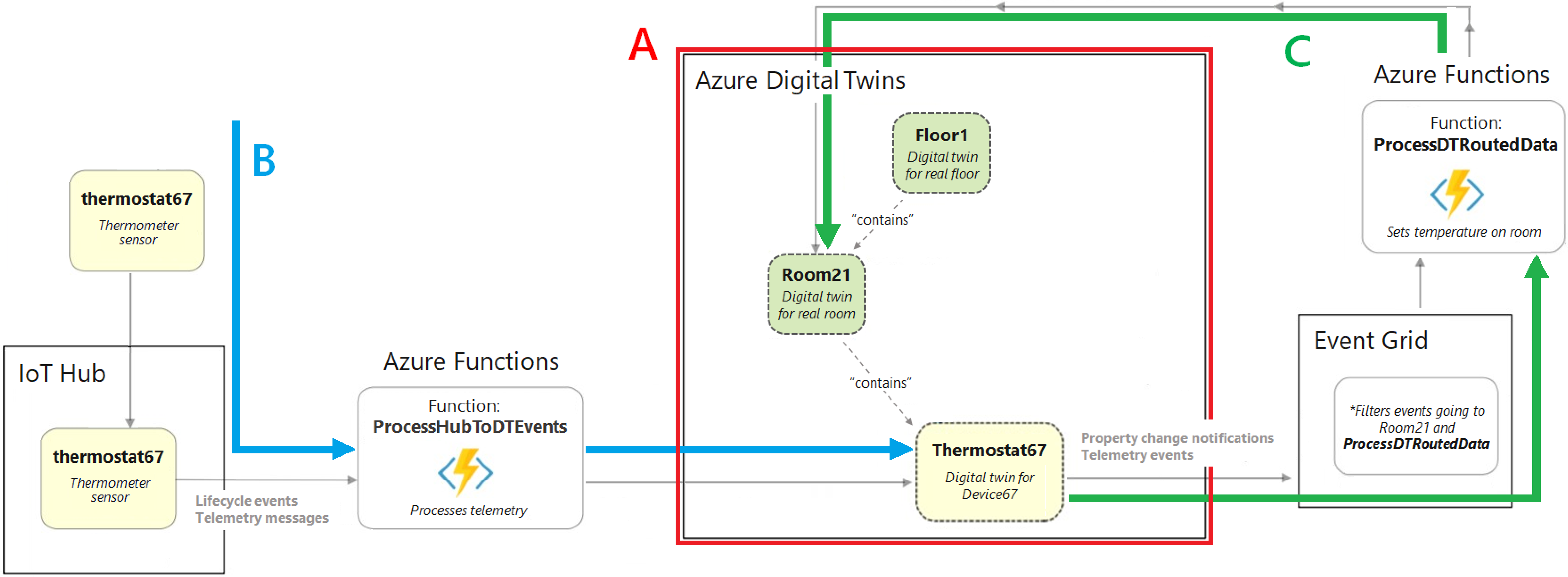


Adds:

- Asset hierarchy (influenced by Brick Schema)
- Capability hierarchy (influenced by Brick Schema)
- Improved spatial model (influenced by BOT)
- Improved Agent representation (companies and departments)

MICROSOFT AZURE IOT SERVICES

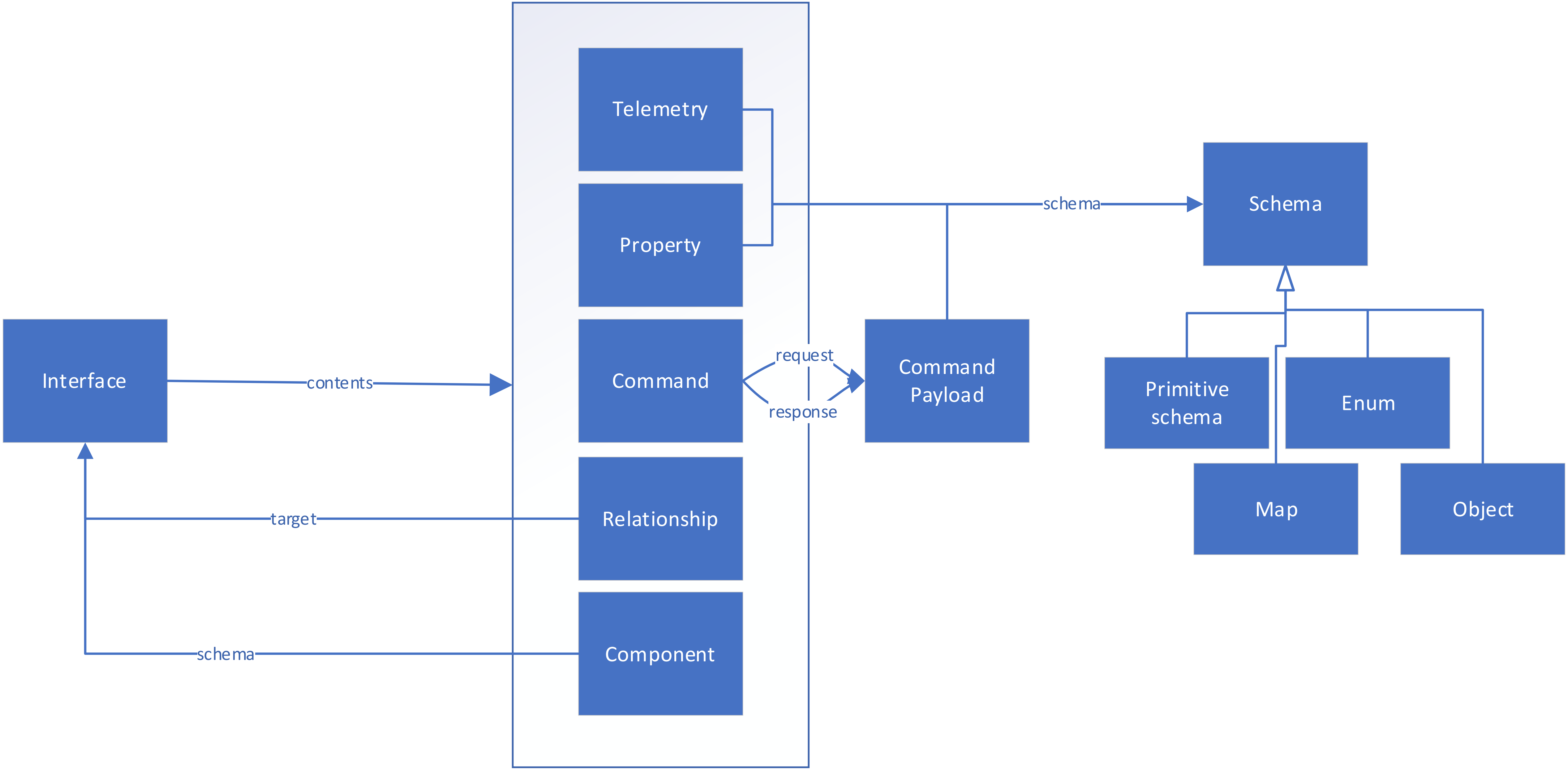
- IoT Edge
  - Runtime for deploying containerized components on edge devices.
  - Libraries for interop on-device, D2C, and C2D.
  - Runs on PCs, Raspberry Pis, in virtual machines, etc.
  - Certified IoT Edge hardware available
- IoT Hub
  - Cloud counterpart to IoT Edge
  - Handles:
    - Component deployment
    - Message routing
    - Software updates
- Digital Twins
  - Graph database holding digital twin representations
  - Represents current state of a space and its smart device portfolio
  - Can forward/trigger events when twins are updated or telemetry ingested
  - Uses DTDL modelling language as schema
- Time Series Insights
  - Analysis and visualization of time-indexed telemetry
- IoT Central
  - Drag’n’drop point-and-click, batteries included solution.


A
B
C
thermostat67
Thermometer sensor
IoT Hub
thermostat67
Thermometer sensor
Lifecycle events
Telemetry messages
Azure Functions
Function:
ProcessHubToDTEvents
Processes telemetry
Azure Digital Twins
Floor1
Digital twin for real floor
"contains"
Room21
Digital twin for real room
"contains"
Thermostat67
Digital twin for Device67
Property change notifications
Telemetry events
Event Grid
*Filters events going to Room21 and ProcessDTRoutedData
Azure Functions
Function:
ProcessDTRoutedData
Sets temperature on room

DTDL – DIGITAL TWIN DEFINITION LANGUAGE

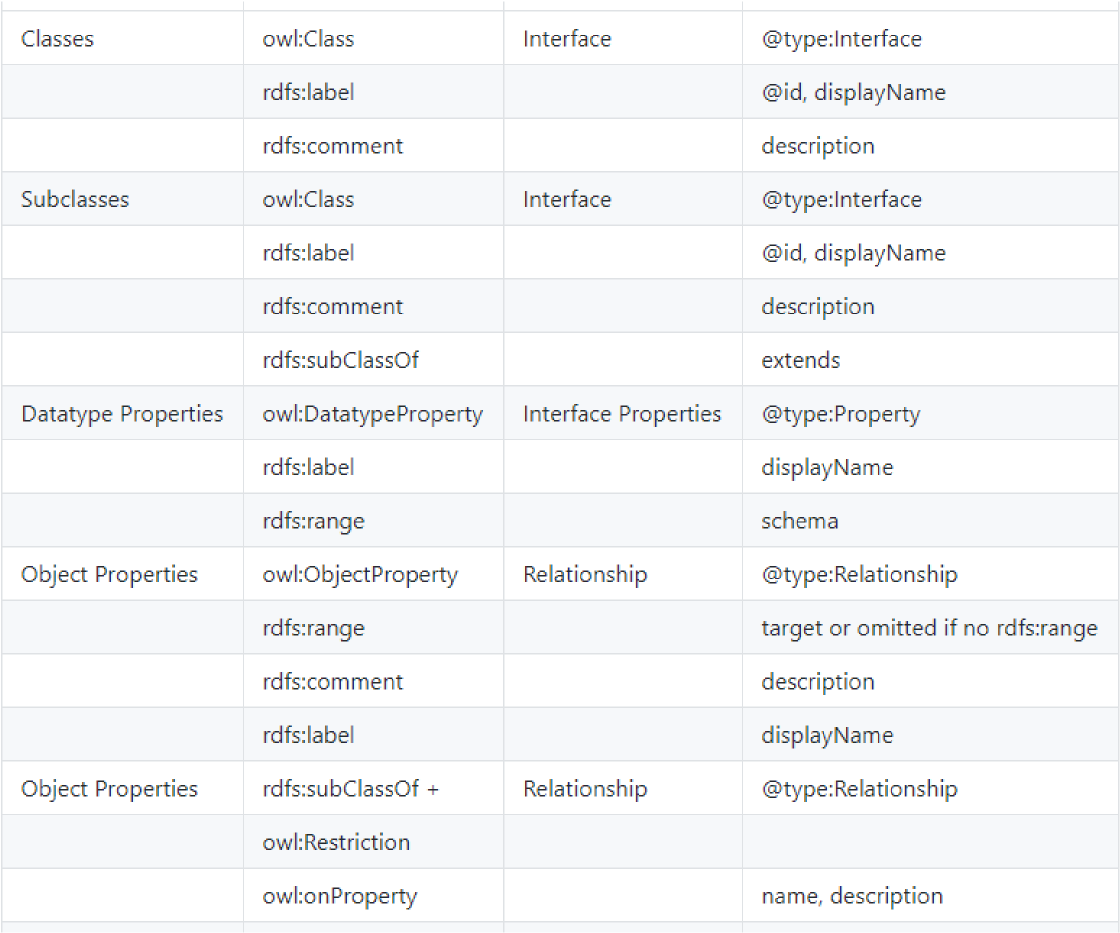
- https://aka.ms/dtdl
- JSON-LD-based
- Class concept: Interface
- Interfaces have contents:
  - Telemetry
  - Property
  - Command
  - Relationship
  - Component

- Interfaces extend one another
- Telemetry and Property objects have schemas
- Commands have request and response objects – which have schemas
- Relationship objects can have target interfaces and multiplicity constraints
- Semantic types for Telemetry and Property based on QUDT
- URNs are used, not IRIs, as identifiers


Interface
contents
Telemetry
Property
Command
Relationship
Component
request
response
Command Payload
schema
Schema
Primitive schema
Enum
Map
Object
target
schema

```
[{
        "@id": "dtmi:com:example:Room;1",
        "@type": "Interface",
        "contents": [
            {
                "@type": "Property",
                "name": "occupied",
                "schema": "boolean"
            }
        ],
        "@context": "dtmi:dtdl:context;2"
    },
    {
        "@id": "dtmi:com:example:ConferenceRoom;1",
        "@type": "Interface",
        "extends": "dtmi:com:example:Room;1",
        "contents": [
            {
                "@type": "Property",
                "name": "capacity",
                "schema": "integer"
            }
        ],
        "@context": "dtmi:dtdl:context;2"
}]
```

TRANSLATING OWL TO DTDL



Classes	owl:Class	Interface	@type:Interface
	rdfs:label		@id, displayName
	rdfs:comment		description
Subclasses	owl:Class	Interface	@type:Interface
	rdfs:label		@id, displayName
	rdfs:comment		description
	rdfs:subClassOf		extends
Datatype Properties	owl:DatatypeProperty	Interface Properties	@type:Property
	rdfs:label		displayName
	rdfs:range		schema
Object Properties	owl:ObjectProperty	Relationship	@type:Relationship
	rdfs:range		target or omitted if no rdfs:range
	rdfs:comment		description
	rdfs:label		displayName
Object Properties	rdfs:subClassOf +	Relationship	@type:Relationship
	owl:Restriction		
	owl:onProperty		name, description

- XSD datatypes translate into DTDL primitives (bool, float, int, etc) with string as fallback.
- RDF label/comment translate into DTDL displayName and description fields (preserving language tags)
- DTDL Properties on Relationships <-> OWL Annotation properties on Object properties
- DTDL Components <-> OWL Classes

THE OWL2DTDL TOOL

```
./OWL2DTDL -u https://w3id.org/rec/full/3.3/ -i ./RecIgnoredNames.csv -o
/Users/karl/Desktop/DTDL/
```

Options

```
- -n, --no-imports       Sets program to not follow owl:imports declarations.
- -f, --file-path        Required. The path to the on-disk root ontology file to translate.
- -u, --uri-path         Required. The URI of the root ontology file to translate.
- -o, --outputPath       Required. The directory in which to create DTDL models.
- -m, --merged-output    Sets program to output one merged JSON-LD file for batch import into ADT.
- -i, --ignorefile       Path to a CSV file, the first column of which lists (whole or partial) IRI:s that
  should be ignored by this tool and not translated into DTDL output.
- -s, --ontologySource   An identifier for the ontology source; will be used to generate DTMI:s per the
  following design, where interfaceName is the local name of a translated OWL class, and ontologyName is the
  last segment of the translated class's namespace:
  <dtmi:digitaltwins:{ontologySource}:{ontologyName}:{interfaceName};1>.
```

DTMI MINTING

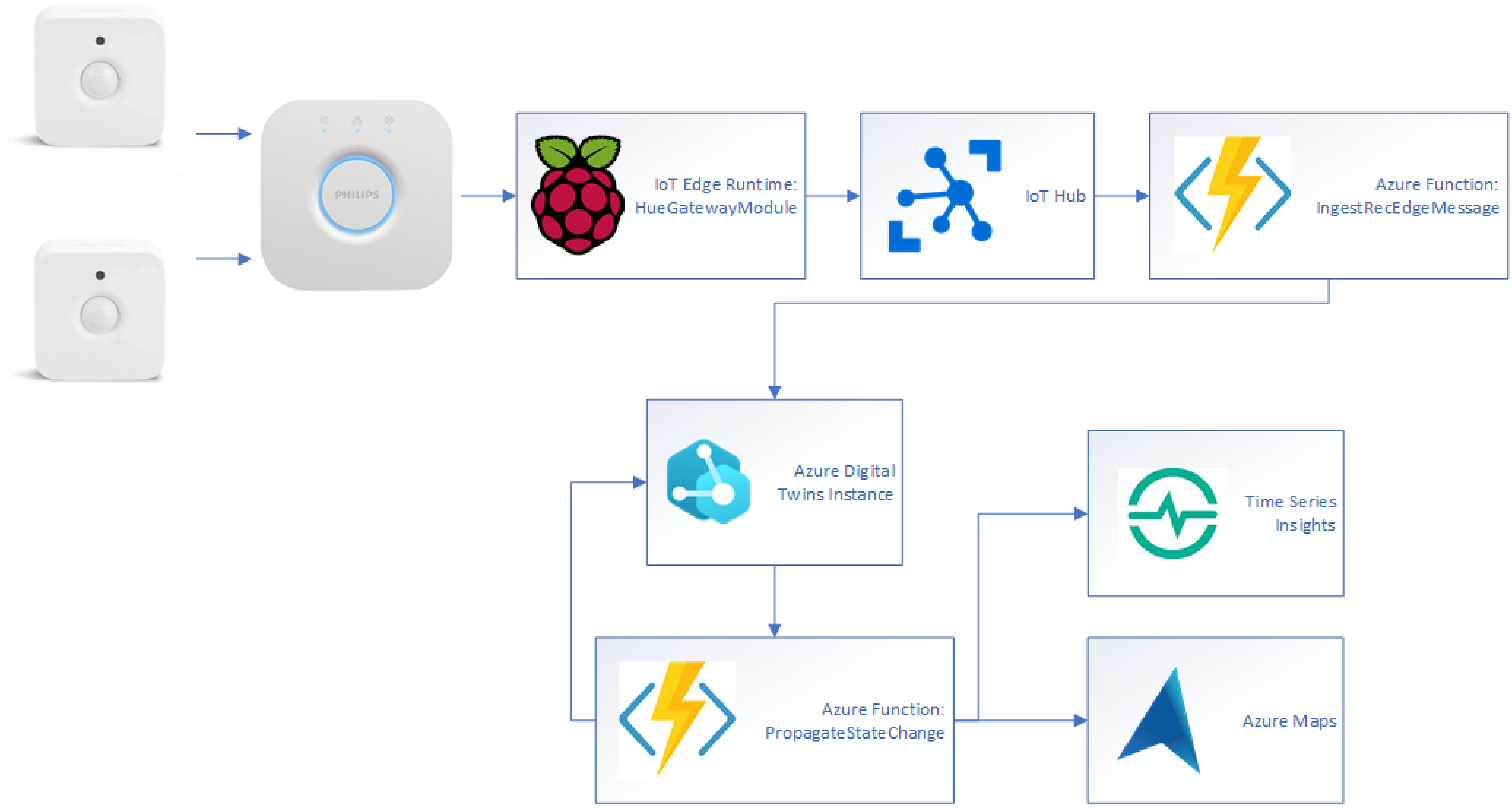
- DTMI:s for named classes are minted based on the classes' URIs by concatenating five components:
  - The "dtmi:digitaltwins:" prefix
  - The ontology source, either given by a CLI option (-s), or generated by reverting the hostname and concatenating with the path segments excluding those that go into the ontology name or local name, see below
  - The ontology name: the last fragment of the URI before the local name
  - The local name of the class
  - The DTMI version identifier; for now hardcoded as ";1“
- E.g., `https://w3id.org/rec/device/Actuator` becomes `dtmi:digitaltwins:org:w3id:rec:device:Actuator;1`. If the CLI option `-s rec_3_3` is given it becomes `dtmi:digitaltwins:rec_3_3:device:Actuator;1`

REC-DTDL ONTOLOGY

- https://github.com/Azure/opendigitaltwins-building
- https://www.youtube.com/watch?v=mN0pAvC2pAo

JTH SMART SPACE DEMONSTRATOR

- 2 sites, 12-15 rooms
- Telemetry gathered:
  - Temperature
  - Illuminance
  - Motion
  - Door opening/closing
  - Person detection
- Visualization: Time series and indoor maps
- Spatial anchors for Mixed Reality overlay
- Next steps: analysis for suggested operations/utilization?



PHILIPS
IoT Edge Runtime:
HueGatewayModule
IoT Hub
Azure Function:
IngestRecEdgeMessage
Azure Digital
Twins Instance
Time Series
Insights
Azure Function:
PropagateStateChange
Azure Maps

DEMONSTRATION

WHAT’S THE POINT?

- Putting these standard pieces together took a couple of days
- Majority of time was on understanding C# Hue interface and building the HueGatewayModule
- Only edge connectors need to be created and configured for specific hardware; once messages are in REC Edge message format, they’ll be ingested and processed
- The entire system could be replicated in a matter of hours
- Moving outside of our (OWL) comfort zone has forced us to make some tough tradeoffs (see WOP paper)
  - But definitely worth it for increased visibility and usability