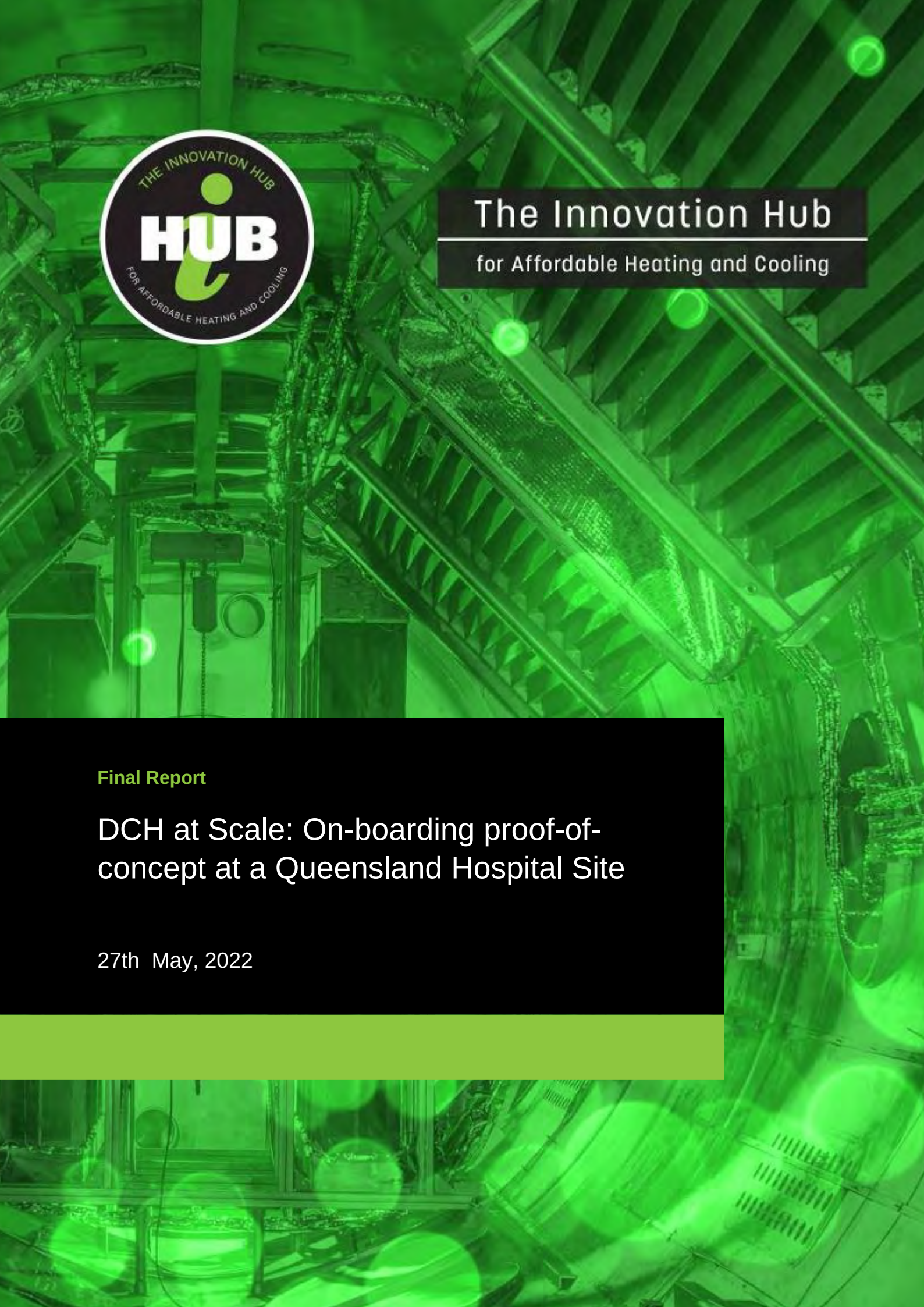

Final Report

DCH at Scale: On-boarding proof-of-concept at a Queensland Hospital Site

27th May, 2022

About i-Hub

The Innovation Hub for Affordable Heating and Cooling (i-Hub) is an initiative led by the Australian Institute of Refrigeration, Air Conditioning and Heating (AIRAH) in conjunction with CSIRO, Queensland University of Technology (QUT), the University of Melbourne and the University of Wollongong and supported by Australian Renewable Energy Agency (ARENA) to facilitate the heating, ventilation, air conditioning and refrigeration (HVAC&R) industry's transition to a low emissions future, stimulate jobs growth, and showcase HVAC&R innovation in buildings.

The objective of i-Hub is to support the broader HVAC&R industry with knowledge dissemination, skills-development and capacity-building. By facilitating a collaborative approach to innovation, i-Hub brings together leading universities, researchers, consultants, building owners and equipment manufacturers to create a connected research and development community in Australia.

This Project received funding from ARENA as part of ARENA's Advancing Renewables Program.
The views expressed herein are not necessarily the views of the Australian Government, and the Australian Government does not accept responsibility for any information or advice contained herein.

Primary Project Partner

DCH at Scale: On-boarding Proof-of-Concept at a Queensland Hospital Site

Metro North Health (MNH) has a diverse set of buildings with large variability in age and systems contained within. Like many hospital buildings, they are in a constant state of flux and receive continual and ongoing upgrades. While telemetry and sensors have the ability to generate large datasets, most of them are inaccessible due to on-premise data silos coupled with inconsistent naming conventions, while electricity and gas invoices arrive as PDF or paper documents. Modelling data becomes a per-building activity, making scalable application development an impossibility. Without a standard metadata schema, application development cannot happen across the entire sector. This project will allow MNH to trial the CSIRO cloud based platform the Smart Building Data Clearing House (DCH).

Lead organisation
Metro North Health, Queensland Government

Project commencement date
March, 2022

Completion date
27 May, 2022

Date published
27 May, 2022

Contact name
Christopher Aynsley-Hartwell

Email
Christopher.Aynsley-Hartwell@health.qld.gov.au

Project website
https://www.ihub.org.au/dhc9-dch-at-scale-on-boarding-proof-of-concept-at-a-qld-hospital-site/

Lead Author
Arne Hansen (Buildings Evolved)

Contributing Authors
Christopher Aynsley-Hartwell (Metro North Health), James Ward (Metro North Health), Chris Schneider (Bar-tech), Ariel Tobey (Buildings Evolved), and Lindsay Rex (Buildings Evolved).

Table of Contents

List of figures

1 SUMMARY

1.1 Background

The DCH 9 project is a proof of concept research and development project devised to onboard Heating, Ventilation and Air-Conditioning (HVAC) systems at a large complex of buildings at a Queensland hospital site – Royal Brisbane and Women's Hospital – (circa 100,000 m2) into the iHub Smart Building Data Clearing House (DCH). This will allow scalability testing of the DCH and improve the development of the Brick building data model (schema and ontology) to support a wider range of use cases. The rich data set collated will allow the development of advanced applications and services.

Bar-tech Automation is the current controls integrator at four hospital sites and has developed rich data sets for analysis and integration. The project posits that automation of on-boarding existing building management control systems (BMCS) is achievable at scale by developing automated commissioning tools and moving towards a real-time messaging bus for building plant and equipment. The project used a measurement and verification (M&V) application hosted in the DCH to begin to develop a baseline, and reports have been developed in Microsoft Power BI tool against the REETSEF KPIs for the beneficiary; Metro North Health's (MNH) Royal Brisbane and Women's Hospital site.

Hospitals have extensive flexible demand resources and an interest in reducing their carbon footprint under mandates from State Governments. Metro North Health is actively pursuing portfolio-wide, or enterprise BMS for integration of assets and energy optimisation to assist in this task and has an interest in developing their requirements further, informed by the outcomes of this project.

1.2 Executive summary

The DCH9 project achieved the following objectives:

Documentation: *reference design for DCH on-boarding, training and manuals completed (Buildings Evolved)*

On-boarding a building from a building metadata viewpoint is challenging due to a lack of standardisation between buildings. The project investigated several methods for on-boarding slow-changing asset data at the hospital site to expedite the process of integrating a building ontology to the DCH and its objective functions. The following options were evaluated:

1. Using python, to generate models programmatically allows transformation of existing datasets into a Brick model.
2. Using a UI driven tool (Protege/WebProtege) we found it did not have great utility in generating a brick model due to over 180 lines of code in a simple 1 zone local demonstrator test site; this would mean 180 or more interactions with the webpage to generate a single model.
3. Using the DCH web-based Brick generation wizard, The DCH has a unique web-based wizard specifically targeted toward generation of a Brick model. While Protege/WebProtege were found to be difficult to use, this could be attributed to the general nature of the tool for RDF, rather than an RDF based tool designed specifically for generating Brick/building models. Consequently, the wizard performs well compared to the general RDF tools, but would still likely be difficult to use if on-boarding a large or complex building.

Software: *Build a dashboard application for monitoring and reporting on REETSEF performance indicators by (Buildings Evolved)*

1. 10 KPIs outlined in the REETSEF report were developed covering energy intensity, electrical demand, load-shifting, air pollution, emissions and reporting, self consumption of renewables, cost and billing.
2. The range of KPIs are clear indicators that help owners and operators of renewable and related energy technologies understand and plan to manage building assets in a changing and challenging operational environment.
3. The KPIs have been demonstrated on the hospital site using a limited data set. Greater data availability and validation of assets and their operation would have improved the results. In future, more buildings and

accurate data will demonstrate the effectiveness and maturity of KPIs for validation, feedback and improvements. What has been delivered is proof that digital integration can be achieved by bringing the buildings in this project to a digital readiness state.

4. To improve upon energy efficiency or intensity analysis, additional contextual information is needed to improve the quality and effectiveness of the KPI indicators. For example, m2 is not sufficient to raise awareness of poor performance at a site compared to an energy benchmark without considering factors such as climate, building and/or equipment age, building orientation, degree of medical specialisation etc. It is only when these factors are taken into consideration that a benchmark is effective. Once poor performing sites are identified a deeper dive into operational technology, data from HVAC systems and/or the BMCS should be sought to determine causation.
5. Key takeaways from the KPIs at the hospital site are:
    a) KPI-1; m2 energy intensity was less than the benchmark, 177 kWh/m2 (although metered area was not confirmed) compared to 393 kWh/m2 benchmark average of Australian hospital sites.
    b) KPI-4; Peak demand has been declining over the past few years resulting in reduced charges and demand on the network.
    c) KPI-5; there is low co-incidence between peak electricity consumption and peak wholesale price data indicating that the site may benefit from a variable pricing contract with exposure to the wholesale price.
    d) KPI-8; maximum electricity demand occurs in the summer and autumn months and maximum demand events show the site would benefit from a peak-lopping strategy with the potential to reduce demand charges by 30-40%.
    e) Once owners and operators are aware of performance issues they can develop new business cases to assess energy data by gathering information from BMCS and telemetry data and begin to realise targeted efficiencies and benefits from operational and renewable energy technologies.

Software: *Build an integration tool set for fast on-boarding to the DCH (Bar-tech)*

1. The current implementation of DCH was found to be complex requiring bespoke knowledge for integration. Upgrades are required to enable development houses a reasonable time to interface with a database system, as a user permissions issues consumed more time than anticipated from the project budget.
2. In addition to the complexity of getting data to DCH was the complexity of getting it out of the existing system, the main requirement of this project. There were two main issues with this. The first is some complex naming conventions used across the hospital building portfolio to create “meta information.” These were overcome but required several testing iterations to generate custom code to handle ‘the edge cases’ for generating the brick model to ingest into the DCH.
3. The second was Niagara and JAVA’s lack of good memory management. This resulted in multiple stoppages and manipulation of data queries to slowly get the required data instead of getting large chunks as would be the standard approach. Time for integration of other sites would be greatly improved from the knowledge gained from this project.
4. The successful integration of these hospital buildings into the DCH will enable MNH to achieve strategic goals optimising energy performance against REETSEF KPIs. It will also inform requirements for future enterprise BMS implementations, and enable greater collaboration with research and industry partners. In addition to achieving these strategic goals, MNH would be better able to achieve operational goals related to energy performance optimisations, including balancing energy usage against achieving indoor air quality (IEQ) targets, maximising asset effective-life spans through targeted upgrades of infrastructure that are no longer performing efficiently, and determining improved HVAC load requirements for new projects.
5. For the wider iHub project, the onboarding and integration methodology, lessons learnt and documentation functions to support future channel partners such as Bar-tech in on-boarding buildings to DCH, and deployment of applications and services.

2 DEVELOPMENT OF ONBOARDING TOOLS AND DOCUMENTATION

2.1 Brick metadata schema, graph database & class-entity relationships

From the Brick Schema website[1]:

> *"Brick is an open-source effort to standardise semantic descriptions of the physical, logical and virtual assets in buildings and the relationships between them. Brick consists of an extensible dictionary of terms and concepts in and around buildings, a set of relationships for linking and composing concepts together, and a flexible data model permitting seamless integration of Brick with existing tools and databases. Through the use of powerful Semantic Web technology, Brick can describe the broad set of idiosyncratic and custom features, assets and subsystems found across the building stock in a consistent matter."*

Brick is an entity-relationship or class diagram of a building, and is utilised by the DCH and other applications to build a graph database that provides foreign keys for the unique datastream name within the DCH or other time series databases[2]. This allows querying of the graph database using SPARQL to find a datastream name, or multiple datastream names. With the datastream names as a variable, a further query of the time-series database yields data from the required datastreams. A metadata schema allows a standard set of classes and naming convention to reference arbitrary names sourced from a BMS or other form of operational technology. In this way, a metadata schema simply acts as middle-ware – allowing standard database queries to reference any known arbitrary datastream name. It is envisaged that the mechanical and electrical trades will provide the relationship between the standard Brick metadata schema and the unique names that reside on their client systems.

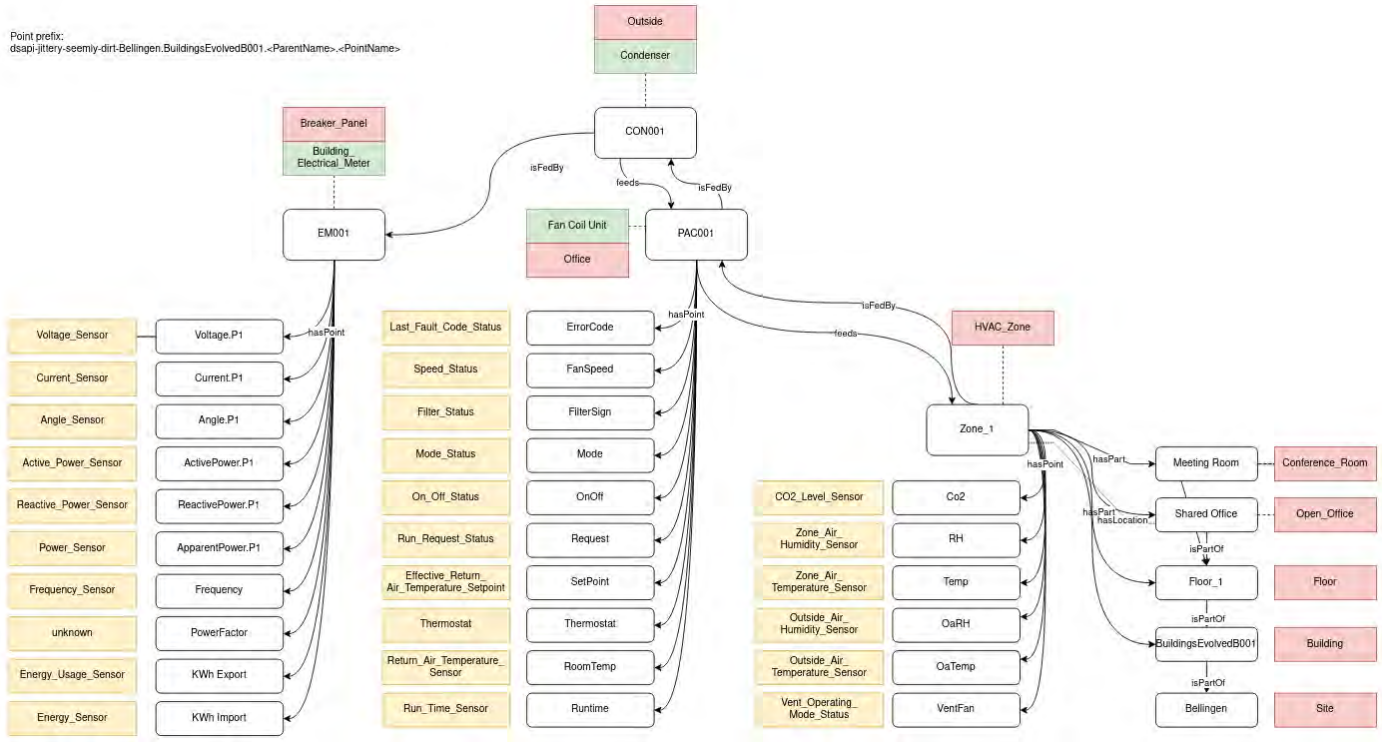

Figure 1: Brick – a diagram representing the metadata schema for the living lab

Brick uses the W3C standard "Resource Description Framework" (RDF) that describes the relationships between objects in the form subject, predicate and object. For example, a way to represent the idea that "zone 1 has point CO2 level sensor" would be expressed in RDF as "zone 1" (subject), "has point" (predicate), CO2 level sensor (object). Other examples from Figure 1 are "Fan coil 1 is fed by Condenser 1" or "Meeting room is part of Floor 1". Therefore,

[1] https://brickschema.org/

[2] Fierro, Prakash, Mosiman et al (2020). Shepherding Metadata Through the Building Lifecycle.

queries can rapidly generate a graph of the relationship between these objects so that an application developer using the metadata schema can easily find a list of what points zone 1 has, using the above example.

An excellent overview video is available which features the academics behind Brick Schema: Gabe Fierro and Jason Koh[3].

A reference schema is available at ref-schema.brickschema.org[4].

2.2 Brick: Building a metadata schema

There are a few methods available to generate Brick models, of which all were investigated during the research phase of the project.

Option 1: Using Python

Brick schema is built and maintained using Python[5]. Therefore it is a natural place to start generating a Brick model for the living lab.

Being able to generate models programmatically allows transformation of existing datasets into a Brick model. For example, a CSV export of BMS object catalogue can be manually manipulated to provide the subject predicate and object for each entity that will form the class diagram. Tools such as OpenRefine[6] provide a web-driven method to clean and manipulate exported object catalogues ready for ingestion through transformation scripts.

Several examples of python code are available on the Brick Schema GitHub site that perform CSV to Brick model conversions[7]. Other conversion tool code examples in this repository include, IFC (BIM) to Brick[8] & TSV to Brick[9]. In addition, readthedocs.io has a page for brickschema python[10].

The Brick schema authors and maintainers have built a Python package py-brickschema[11] that installs via the Linux command using “pip install brickschema”. The brickschema package allows:

- management and querying of Brick models;
- simple OWL-RL, SHACL and other inference;
- conversion of Haystack models to Brick; and
- adding VBIS tags to Brick model, or getting Brick types from VBIS tags[12]

Brickschema is also used to import various python libraries as shown in Figure 2 for writing applications to generate a graph, manipulate entities, or provide parsing and inference.

```
1  from brickschema.namespaces import RDF, RDFS, BRICK, TAG, OWL
2  from brickschema.graph import Graph
3  from rdflib import Namespace, URIRef, Literal
```

Figure 2: Brick - using the brickschema python package to import various python libraries

In our R&D, we validated the above tools, and used the Brickschema python package to generate valid Brick models. We further used Brickschema package coupled with SHACL to parse and validate the RDF output of the python script,

[3] https://www.youtube.com/watch?v=5w3uu_vevCA
[4] https://ref-schema.brickschema.org/#hasTimeseriesReference
[5] https://github.com/BrickSchema/Brick/tree/master/bricksrc
[6] https://openrefine.org
[7] https://github.com/BrickSchema/Brick/tree/master/tools/convert
[8] https://github.com/gtfierro/brick-ifc-convert
[9] https://github.com/BrickSchema/Brick/tree/master/examples
[10] https://brickschema.readthedocs.io/en/add-brickify/source/brickschema.html
[11] https://github.com/BrickSchema/py-brickschema
[12] https://brickschema.readthedocs.io/_/downloads/en/latest/pdf/

stored as a turtle/ttl file. An example is the BrickEntityShapeBase.ttl[13], and will likely be a method of application developers communicating their data model requirements to integrators..

To achieve a successful parse of the ttl file by SHACL, the full Brick schema ttl file needs to be parsed into the generated ttl. However, the integration of the two ttl files is not a prerequisite for loading the Brick ttl into the DCH. In this case, line 238 of the image shown in Figure 3 is commented out when writing a file with a target for the DCH, and as indicated when the target is a SHACL parser/interpreter.

```
235  Assuming Brick.ttl is in the root directory of this repo, you can load it with the following.
236  """
237
238  g.parse("Brick.ttl", format="ttl")
239
240  """
```

Figure 3: Brick - Python code to parse and add the base Brick TTL to the generated TTL

Another resource provided in the Brick Schema github repository provides a “quick start” as shown in Figure 4 to generate a brick model, and provides excellent documentation as comments in code[14].

[13] https://raw.githubusercontent.com/BrickSchema/Brick/master/shacl/BrickEntityShapeBase.ttl

[14] https://github.com/BrickSchema/Brick/blob/master/examples/example1/generate.py

```python

# (subject, predicate, object)
g.add((BLDG.AHU1A, RDF.type, BRICK.Air_Handler_Unit))
# you can use "quotes" to name entities as well
g.add((BLDG["VAV2-3"], RDF.type, BRICK.Variable_Air_Volume_Box))


"""
We can also add relationships between entities in our Brick model. The
BRICK.feeds relationship indicates a sequence between two pieces of equipment
"""

g.add((BLDG.AHU1A, BRICK.feeds, BLDG.VAV2_3))

"""
Let's add a few more entities so the graph is more interesting. We will
implement the Brick model for the "blue" entities in the sample graph at
brickschema.org
"""

# declare entities first
g.add((BLDG["VAV2-4"], RDF.type, BRICK.Variable_Air_Volume_Box))
g.add((BLDG["VAV2-4.DPR"], RDF.type, BRICK.Damper))
g.add((BLDG["VAV2-4.DPRPOS"], RDF.type, BRICK.Damper_Position_Setpoint))
g.add((BLDG["VAV2-4.ZN_T"], RDF.type, BRICK.Supply_Air_Temperature_Sensor))
g.add((BLDG["VAV2-4.SUPFLOW"], RDF.type, BRICK.Supply_Air_Flow_Sensor))
g.add((BLDG["VAV2-4.SUPFLSP"], RDF.type, BRICK.Supply_Air_Flow_Setpoint))
g.add((BLDG["VAV2-3Zone"], RDF.type, BRICK.HVAC_Zone))
g.add((BLDG["Room-410"], RDF.type, BRICK.Room))
g.add((BLDG["Room-411"], RDF.type, BRICK.Room))
g.add((BLDG["Room-412"], RDF.type, BRICK.Room))

# declare edges
g.add((BLDG["AHU1A"], BRICK.feeds, BLDG["VAV2-4"]))
g.add((BLDG["VAV2-4"], BRICK.hasPart, BLDG["VAV2-4.DPR"]))
g.add((BLDG["VAV2-4.DPR"], BRICK.hasPoint, BLDG["VAV2-4.DPRPOS"]))
g.add((BLDG["VAV2-4"], BRICK.hasPoint, BLDG["VAV2-4.SUPFLOW"]))
g.add((BLDG["VAV2-4"], BRICK.hasPoint, BLDG["VAV2-4.SUPFLSP"]))
g.add((BLDG["VAV2-3"], BRICK.feeds, BLDG["VAV2-3Zone"]))
g.add((BLDG["VAV2-3Zone"], BRICK.hasPart, BLDG["Room-410"]))
g.add((BLDG["VAV2-3Zone"], BRICK.hasPart, BLDG["Room-411"]))
g.add((BLDG["VAV2-3Zone"], BRICK.hasPart, BLDG["Room-412"]))

"""
We can "serialize" this model to a file if we want to load it into another program.
"""
with open("example.ttl", "wb") as f:
    # the Turtle format strikes a balance beteween being compact and easy to read
    f.write(g.serialize(format="ttl"))

```

Figure 4: Brick - Python code example to generate a Brick model

The graph represented in Figure 1 was converted into python using the template shown in Figure 4, and was then used to generate a valid Brick model for the DCH.

The main steps in creating a Brick model from raw Python are represented in the Figures 5, 6, 7 and 8.

```
# DEFINING THE BRICK MODEL

# add() inserts another triple into the graph as (subject, predicate, object)
# add site, building, floors, rooms, locations
g.add((EX.Bellingen, RDF.type, BRICK.Site))
g.add((EX.BuildingsEvolvedB001, RDF.type, BRICK.Building))
g.add((EX.Floor_1, RDF.type, BRICK.Floor))
g.add((EX.Meeting_Room, RDF.type, BRICK.Conference_Room))
g.add((EX.Shared_Office, RDF.type, BRICK.Open_Office))
g.add((EX.Outside, RDF.type, BRICK.Outside))
g.add((EX.Meter_Panel, RDF.type, BRICK.Breaker_Panel))

# add HVAC equipment and object properties
g.add((EX.CON001, RDF.type, BRICK.Condenser))
g.add((EX.PAC001, RDF.type, BRICK.Fan_Coil_Unit))
g.add((EX.ErrorCode, RDF.type, BRICK.Last_Fault_Error_Code))
g.add((EX.FanSpeed, RDF.type, BRICK.Speed_Status))
g.add((EX.FilterSign, RDF.type, BRICK.Filter_Status))
g.add((EX.Mode, RDF.type, BRICK.Mode_Status))
g.add((EX.OnOff, RDF.type, BRICK.On_Off_Status))
g.add((EX.Request, RDF.type, BRICK.Run_Request_Status))
# check this one, status not actual setpoint, perhaps a deletion?
g.add((EX.SetPoint, RDF.type, BRICK.Effective_Return_Air_Temperature_Setpoint))
g.add((EX.Thermostat, RDF.type, BRICK.Thermostat))
g.add((EX.RoomTemp, RDF.type, BRICK.Return_Air_Temperature_Sensor))
```

Figure 5: Brick - defining RDF types in Python to generate a Brick model

In Figure 6, if each triple in each line is read consecutively, it describes the subject, predicate & object. Both directions are declared: for example, the relationship ‘feeds’ is matched with ‘isFedBy’. This allows the query to travel in both directions, querying entities upstream or downstream of the target object.

```
############################################################
#  add the relationships between entities (declaring 'edges')
#### high level
g.add((EX.EM001, BRICK.feeds, EX.CON001))
g.add((EX.CON001, BRICK.feeds, EX.PAC001))
g.add((EX.PAC001, BRICK.feeds, EX.Zone_1))

g.add((EX.CON001, BRICK.isFedBy, EX.EM001))
g.add((EX.PAC001, BRICK.isFedBy, EX.CON001))
g.add((EX.Zone_1, BRICK.isFedBy, EX.PAC001))

#### Rooms
g.add((EX.Bellingen, BRICK.hasPart, EX.BuildingsEvolvedB001))
g.add((EX.BuildingsEvolvedB001, BRICK.hasPart, EX.Floor_1))
g.add((EX.BuildingsEvolvedB001, BRICK.hasPart, EX.Outside))
g.add((EX.Floor_1, BRICK.hasPart, EX.Shared_Office))
g.add((EX.Floor_1, BRICK.hasPart, EX.Meeting_Room))
g.add((EX.Outside, BRICK.hasPart, EX.Meter_Panel))

g.add((EX.Meter_Panel, BRICK.isPartOf, EX.Outside))
g.add((EX.Meeting_Room, BRICK.isPartOf, EX.Floor_1))
g.add((EX.Shared_Office, BRICK.isPartOf, EX.Floor_1))
g.add((EX.Outside, BRICK.isPartOf, EX.BuildingsEvolvedB001))
g.add((EX.Floor_1, BRICK.isPartOf, EX.BuildingsEvolvedB001))
g.add((EX.BuildingsEvolvedB001, BRICK.isPartOf, EX.Bellingen))
```

Figure 6: Brick - defining relationships between entities in python for a Brick model

In Figure 7, each data point is then attached to a unique id for the target time series database, in this case the DCH. Brick/Mortar, in contrast, use the relationship “hasTimeSeriesId” coupled with a UUID linking to an ID contained within

Postgres TimescaleDB. It has not been tested in this project, but it is logically deduced that the same approach would work with InfluxDB, Prometheus or other time series databases.

While the sting following the Literal is arbitrary, that that proceeds it is standardised around the Brick schema. Hence a standardised SPARQL query to an RDF graph will allow the return of a single or an array of unique IDs for the time series database.

```
143  # This is a fork from new Brick functionality, which uses BRICK.hasTimeseriesId rather than SENAPS.stream
144  # Have to store a string rather than UUID as intended with Brick/Mortar AFAIK
145  # g.add((EX.ErrorCode, BRICK.hasTimeseriesId, Literal("bd65600d-8669-4903-8a14-af88203add38")))
146  g.add((EX.ErrorCode, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.
147  g.add((EX.FanSpeed, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.P
148  g.add((EX.FilterSign, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001
149  g.add((EX.Mode, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.PAC00
150  g.add((EX.OnOff, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.PAC0
151  g.add((EX.Request, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.EM
152  g.add((EX.SetPoint, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.P
153  g.add((EX.Thermostat, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001
154  g.add((EX.RoomTemp, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.P
155  g.add((EX.RunTime, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.PA
156
```

Figure 7: Brick - providing external references to DCH datastream names in python for a Brick model

Brick therefore allows for discovery of entities in the class-relationship diagram/metadata graph, both upstream or downstream, and further provides a foreign key to a timeseries database. Application developers, knowing the target database, can then perform queries across the resultant datastream names returned from a query against the metadata graph.

```
262  # save the output to "name.ttl"
263  g.serialize(destination='test8.ttl', format='turtle')
```

Figure 8: Brick - Python serialising the output to a TTL file

In Figure 9, a simple Brick model is represented, generated by the above code examples. This ttl file was subsequently uploaded successfully to the DCH as shown in Figure 10.

Figure 9: Brick - simple Brick schema represented in TTL format. This example is not merged with the main Brick Schema TTL.

TTL files can then be uploaded to the DCH or used to power queries of local time series databases. It is worth noting that when using the **senaps:stream_id** or **brick:hasTimeseriesId**, the TTL file is then targeted towards a particular platform or environment, but could possess multiple external references – i.e. both **senaps:stream_id** and **brick:hasTimeseriesId** could co-exist.

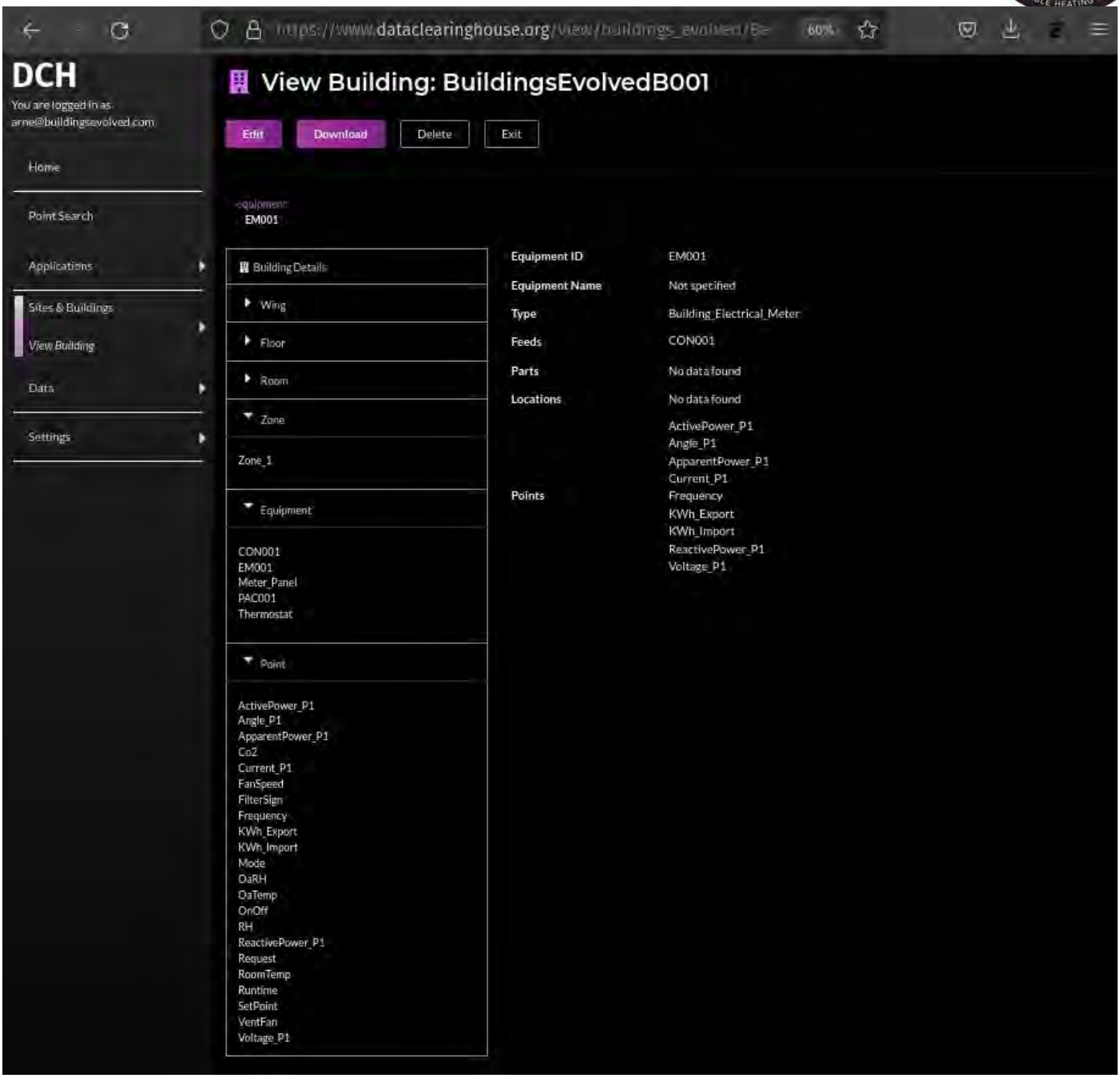

Figure 10: Brick - Python generated Brick TTL file, parsed and loaded successfully into DCH

Option 2: using a UI driven tool (Protege/WebProtege)

While it is possible to generate a brick model within a UI driven tool such as webprotege[15], as shown in Figure 11 or Protege and equivalent generalised RDF tools were found**Option 2: using a UI driven tool (Protege/WebProtege)** to be more useful to view and edit a generated brick model rather than generation itself.

[15] https://webprotege.stanford.edu/

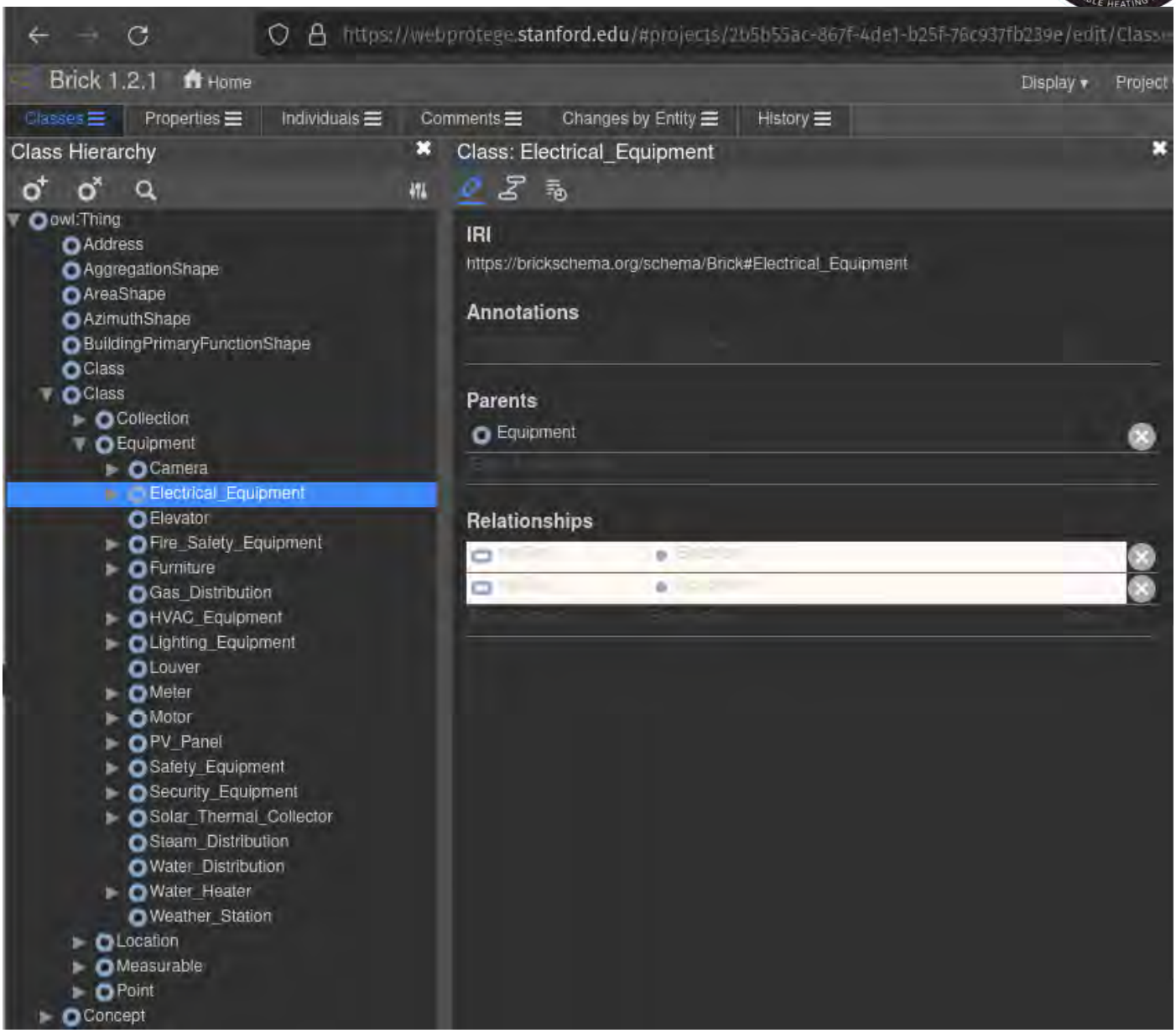

Figure 11: Brick - Stanford University WebProtege RDF viewing tool rendering the Brick 1.2.1 reference ttl graph database

Protege/Webprotege were found to not have great utility in generating a brick model. As there are over 180 lines of code in our simple 1 zone living lab site, this would mean 180 or more interactions with the webpage or user interface to generate a single model. The model generated from the tool could not be iterated, tested or improved upon without going back to the beginning each time. Throughout the development of our Python-based solution to Brick model generation, we had 8 major iterations, reinforcing that presently a Brick model is best generated in a declarative manner, or (more likely) converted from a flat file such as CSV or TSV using Python.

Option 3: Using the DCH web-based Brick generation wizard

The DCH has a unique web-based wizard specifically targeted towards generation of a Brick model. While protege/webprotege are difficult to use, this could be attributed to the general nature of the tool for RDF, rather than an RDF based tool designed specifically for generating Brick/building models. Consequently, the wizard performs well compared to the general RDF tools, but would still likely be difficult to use if onboarding a large or complex building.

First step is to create a site via the DCH website as shown in Figure 12.

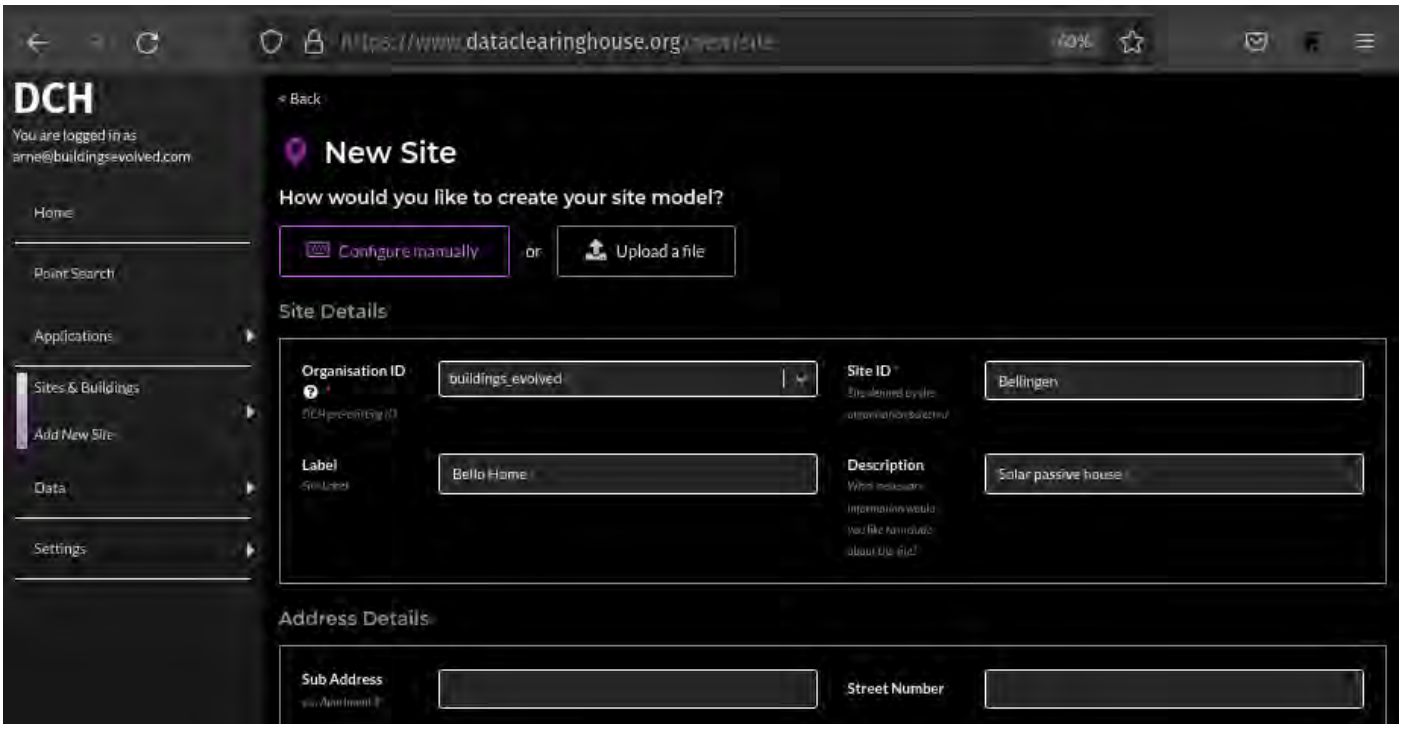

Figure 12: Brick - DCH wizard: creating a site

After a site is created, a building model generation wizard can be launched, or a programatically generated model can be uploaded (as a ttl file) as shown in Figure 13.

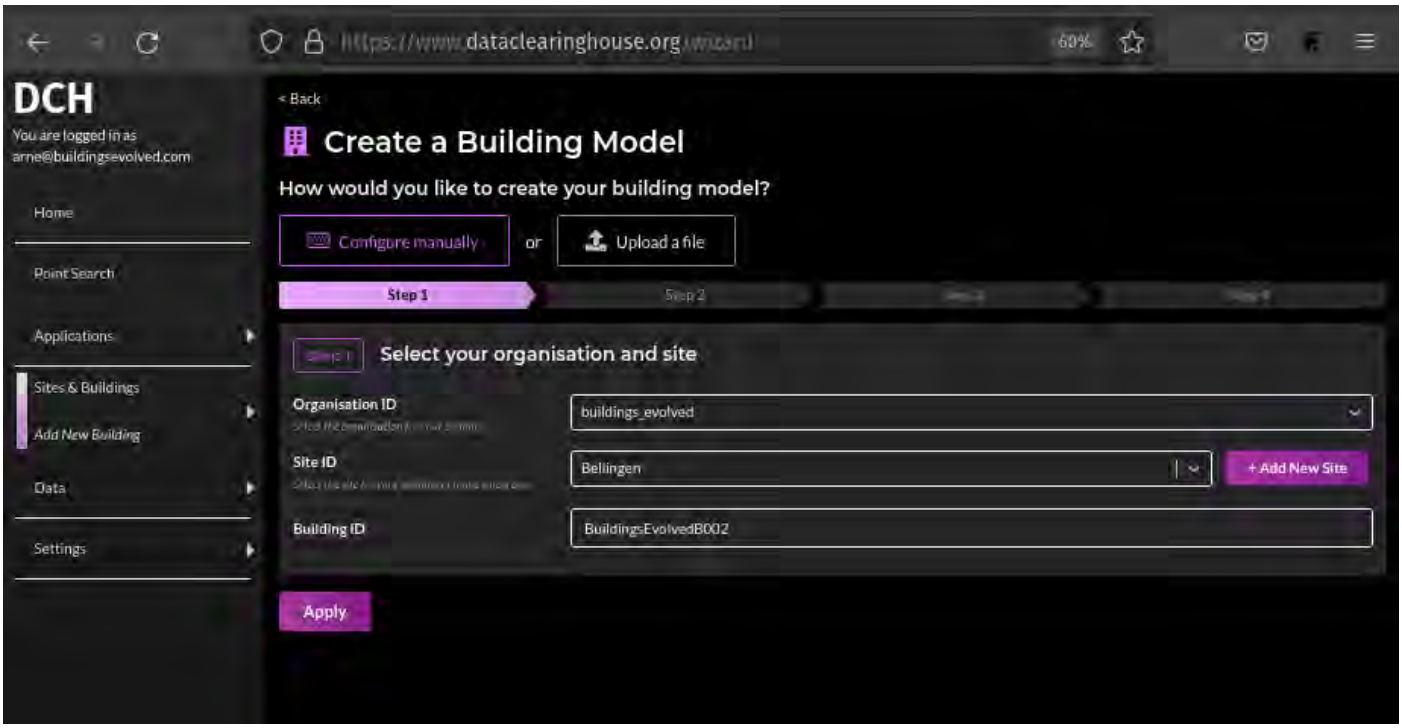

Figure 13: Brick – DCH wizard: step 1

Step 2-4 are dynamic, with fields expanding depending on number of wings/floors/rooms specified as shown in Figure 14

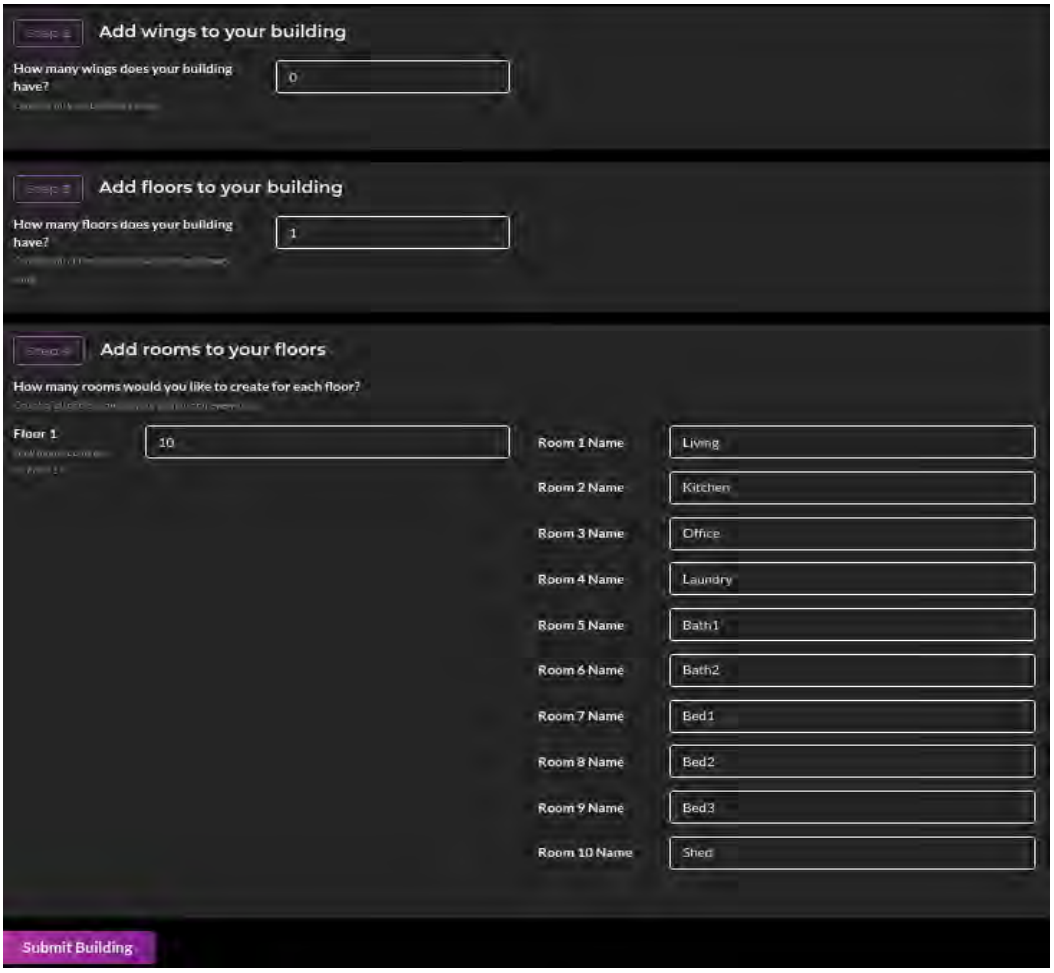

Figure 14: Brick - DCH wizard steps 2-4

Following the basic setup providing the structure of the building into wings/floors and rooms, the user is brought to an editing screen that allows further addition of zones, equipment and points. From there the user is prompted to create the relationships between objects, with the user interface using lookups on type to enforce consistency with the brick schema, as shown in Figure 15.

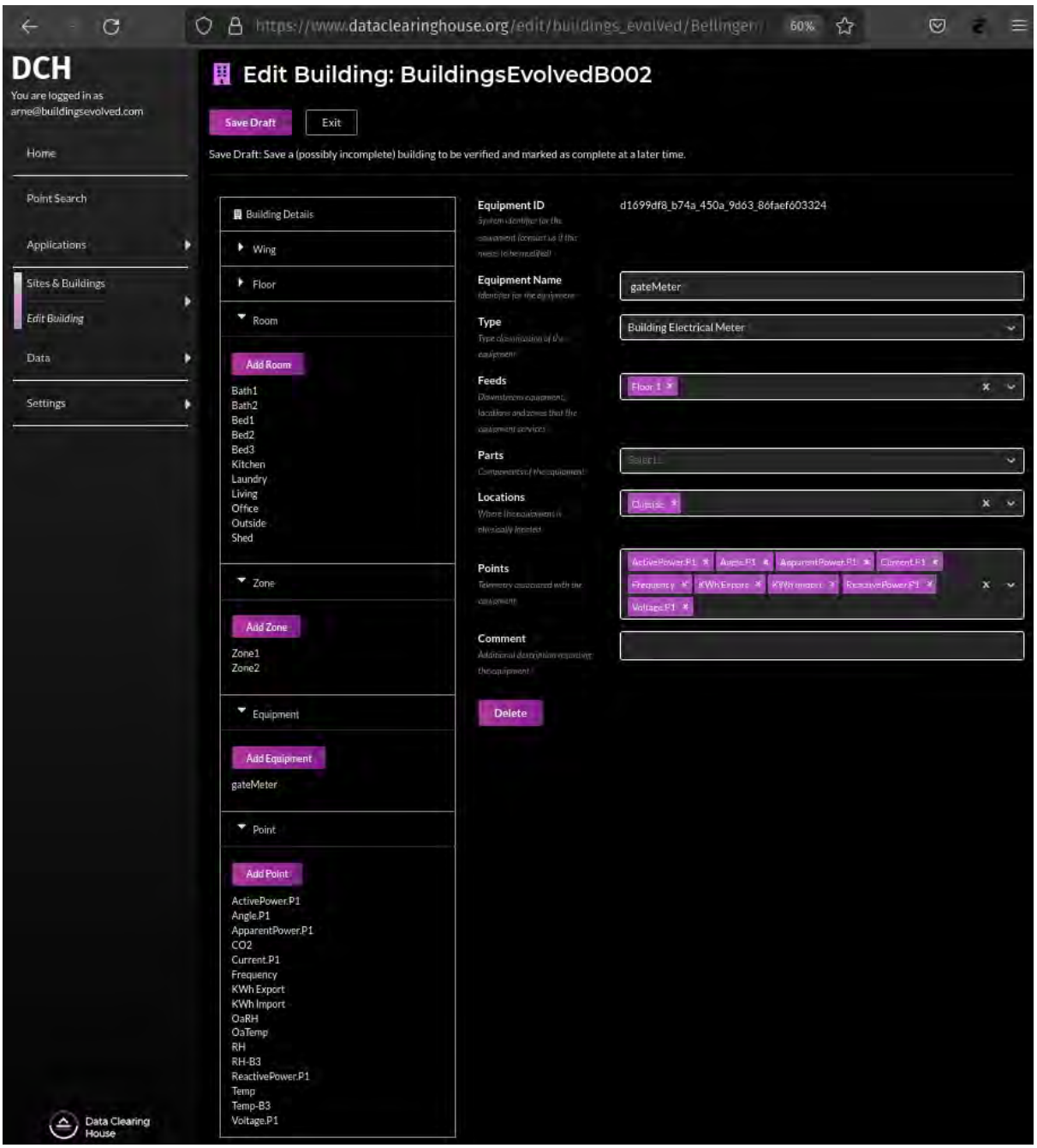

Figure 15: Brick - adding data points in the DCH wizard and creating relationships between entities. Type is a lookup to the Brick schema to enforce consistency.

2.3 Brick: DCH model variations

While the Brick model is intended to be used in a standard way, there were subtle differences with how the DCH implemented a Brick model compared with updated versions of the Brick schema. This is due to the rapidly developing

capability of the Brick schema: indication of this is that two versions of Brick have been released since the commencement of the iHub project.

The variations from the code example provided by the Brick schema github repository include:

Do not bind a URI to the building namespace, bind to a DCH reference comprised of **“dch:org/<orgName>/site/<siteName>/building/<buildingName>”** as shown in Figure 16.

```
12  ## Define the building - differences between vanilla Brick and DCH
13  ## Documented Brick method is to bind a URI to the graph. These are now commented out
14  # EX = Namespace("http://example.com/mybuilding#")
15  # g.bind("Bellingen001", EX)
16
17  ## Versus DCH method
18  EX = Namespace("dch:org/buildings_evolved/site/Bellingen/building/BuildingsEvolvedB001#")
19  # Notably, does not use a URI in the namespace, nor binds that namespace to the graph.
20  # This causes the output as provided in CSIRO example.
```

Figure 16: Brick - DCH variation: URI reference

Custom DCH extensions “Bricx” were not required but were noted. The Senaps (DCH) namespace was bound to the graph to support custom RDF type stream_id as shown in Figure 17.

```
22  # DCH - note that both links are broken or redirect to http with 404 error.
23  # Only using the senaps:stream_id namespace, so comment out bricx
24  # BRICX = Namespace("https://dataclearinghouse.org/schema/Brick/extension#")
25  # g.bind("bricx", BRICX)
26  SENAPS = Namespace("http://senaps.io/schema/1.0/senaps#")
27  g.bind("senaps", SENAPS)
```

Figure 17: Brick - DCH variation: SENAPS namespace, unused BRICX namespace

SENAPS (DCH) had already defined their own predicate “stream_id” to provide a foreign key store in the metadata schema prior to the adoption of BRICK.hasTimeseriesId. As such, we took note of the variation and made our code compatible with DCH. There was the option to add both, but as one would be unused, this becomes a stub, as shown in Figure 18,

```
143 # This is a fork from new Brick functionality, which uses BRICK.hasTimeseriesId rather than SENAPS.stream
144 # Have to store a string rather than UUID as intended with Brick/Mortar AFAIK
145 # g.add((EX.ErrorCode, BRICK.hasTimeseriesId, Literal("bd65600d-8669-4903-8a14-af88203add38")))
146 g.add((EX.ErrorCode, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.
147 g.add((EX.FanSpeed, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.P
148 g.add((EX.FilterSign, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001
149 g.add((EX.Mode, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.PAC00
150 g.add((EX.OnOff, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.PAC0
151 g.add((EX.Request, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.EM
152 g.add((EX.SetPoint, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.P
153 g.add((EX.Thermostat, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001
154 g.add((EX.RoomTemp, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.P
155 g.add((EX.RunTime, SENAPS.stream_id, Literal("dsapi-jittery-seemly-dirt-Bellingen.BuildingsEvolvedB001.PA
156
```

Figure 18: Brick - DCH variation: SENAPS.stream_id vs BRICK.hasTimeseriesId

The Brick model does not provide a method on how to interpret data. As such, CSIRO have authored their own specification dealing with energy metering which incorporates a few extensions to the brick model (many of which have subsequently been backported into the Brick schema). This documentation is available upon request from the DCH team at CSIRO, but should form part of iHub DCH 1 reporting.

3 REETSEF REPORTS FOR HOSPITAL BUILDINGS

3.1 Overview

REETSEF reporting functional requirements are derived from the relevant iHub report by Queensland University of Technology (QUT)[16], referred to as 'the REETSEF report'. 10 KPIs from this report were produced using sample energy and asset related data sets for the beneficiary, Queensland Health. The resultant reports were produced in Microsoft Power BI demonstrating how the KPIs can be useful in helping to understand, manage, and improve the energy productivity of healthcare facilities and the value of renewable energy utilisation in these facilities.

Energy Modelling Technology

Buildings Evolved have developed a sophisticated modelling tool that provides users with an interface or web-form enabling them to upload, transform and write data from the following sources to a database:

- electricity meter data agent (NEM12);
- electricity retailer invoices (ERM & Origin);
- Bureau of Meteorology (BoM);
- Australian Energy Market Operator (AEMO);
- solar irradiance data (Solcast);
- GHG gas emissions factors;
- battery management system; and
- solar PV inverters.

The technical overview of the data architecture from ingestion via the web-form, to processing and transformation, to producing KPIs and reports for analysis is shown in Figure 19.

[16] Ihub, Healthcare Sector: Renewable Energy and Enabling Technology and Services Framework (REETSEF), 2020, p. 12

3.2 System architecture

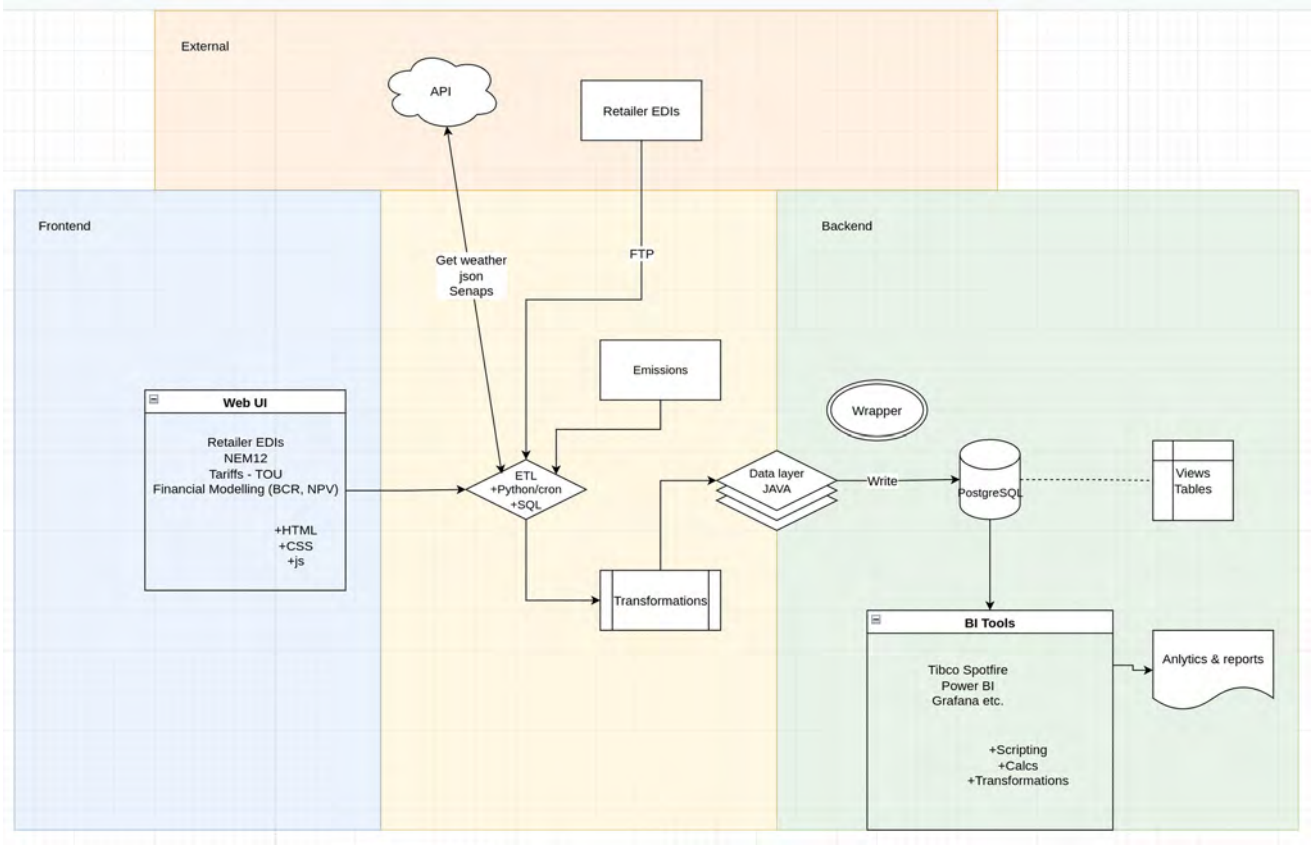

Figure 19: modelling tool process architecture

The basic architecture involves the following processes:

- ingest data via ETL;
- lay down data into a database;
- allow scripts to run across datasets to calculate required outputs;
- allow an array of scenarios and assumptions to represent different what-if scenarios for modelling;
- automatically calculate costs on an interval-by-interval basis;
- produce report outputs; and
- easily allow database querying to be performed by any standardised BI reporting tools, such as Tibco Spotfire, Tableau, Power BI, Grafana et. al.

Various open-source coding languages and software frameworks were used to develop the solution including:

- Python,
- JavaScript,
- PostgreSQL, and
- React.JS, amongst others.

This architecture has user access control, logs and validation processes to ensure data completeness and accuracy. The dashboards presented below, designed from KPIs in the REETSEF report[17] used data generated from the modelling tool.

Users can access the modelling tool via a web-form built on JavaScript and React JS framework. An example electricity (NEM12) extract, transform and load workflow is outlined below:

[17] iHub, Healthcare Sector: Renewable Energy and Enabling Technology and Services Framework (REETSEF), 2020, p12

Time-series electricity consumption file from a metering provider in the appropriate NEM12 format is procured and validated

300	20220522	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
200	QB0367415	B7E7K7Q7B8E8K8	E7	E7	N7	9836012	kWh	30								
300	20171201	766.415	759.102	754.409	753.596	753.918	747.873	748.503	750.628	753.984	756.171	805.847	825.717	853.559	887.673	92
300	20171202	746.368	737.534	734.676	732.394	730.968	721.584	720.92	720.22	717.456	719.911	741.154	759.927	788.968	802.561	802
300	20171203	766.213	765.259	765.244	759.955	759.674	757.447	752.665	745.111	732.358	716.56	741.695	747.87	762.116	754.039	73
300	20171204	741.193	727.511	723.017	721.793	720.273	720.656	718.672	719.425	722.638	733.688	784.095	804.334	853.098	913.049	953
300	20171205	679.427	676.859	672.266	668.958	669.412	669.586	670.735	665.526	681.202	688.654	730.805	748.51	778.963	816.538	856
300	20171206	697.029	689.375	685.853	683.387	674.872	666.75	663.986	660.706	668.508	666.362	701.582	717.753	746.774	768.283	799
300	20171207	670.201	665.553	661.568	662.016	657.274	655.985	644.31	641.42	664.791	666.478	702.686	712.952	743.576	774.304	815
300	20171208	775.245	766.808	760.834	748.112	746.963	742.712	739.516	734.67	730.108	735.701	786.766	808.219	844.349	887.253	943
300	20171209	667.836	672.762	671.783	665.474	666.807	666.9	665.912	664.04	656.697	664.893	676.249	686.613	698.68	707.343	712
300	20171210	669.94	670.191	667.3	664.101	661.81	661.138	663.411	652.507	654.556	651.941	671.221	678.791	688.082	690.806	699
300	20171211	668.623	660.227	654.314	655.905	649.073	647.763	646.544	646.882	650.352	660.801	696.37	721.287	756.507	789.223	830
300	20171212	669.979	666.215	661.722	658.406	658.937	658.971	658.41	657.689	667.177	672.354	707.747	726.452	761.954	785.718	828
300	20171213	700.701	695.479	692.184	684.991	682.373	677.026	677.843	671.511	673.385	681.901	722.568	741.09	772.097	797.489	843
300	20171214	695.788	692.623	686.986	684.043	681.795	682.081	677.787	679.36	689.161	695.141	736.862	750.608	766.294	829.537	873
300	20171215	704.398	706.906	697.198	694.132	690.805	693.26	695.323	692.839	695.027	704.168	751.605	767.676	812.11	843.931	894
300	20171216	769.993	766.93	731.914	741.698	740.287	731.277	735.447	734.688	733.21	738.404	763.8	766.655	781.412	788.672	795
300	20171217	710.627	704.566	699.404	699.526	700.777	696.694	699.478	692.255	687.846	689.725	709.857	724.914	743.97	762.605	769
300	20171218	719.997	724.932	716.165	718.422	706.232	707.902	703.626	703.703	702.584	712.575	758.484	785.658	823.072	851.717	904
300	20171219	706.045	706.348	698.292	695.912	693.994	692.386	688.755	689.13	700.077	710.187	754.796	770.942	822.893	854.046	920
300	20171220	754.211	755.374	740.197	739.307	738.303	739.199	728.218	728.606	732.454	735.16	783.537	808.002	854.204	942.333	966
300	20171221	780.057	775.692	758.478	762.698	748.752	733.427	727.12	715.117	715.508	731.96	780.929	807.505	846.028	927.576	950
300	20171222	781.128	784.548	779.124	781.877	783.855	785.675	775.789	777.085	763.579	772.759	830.608	853.206	894.834	927.747	951
300	20171223	775.745	775.986	767.139	765.221	760.438	757.558	751.944	747.12	723.012	742.09	762.299	769.641	784.444	786.693	794
300	20171224	723.415	721.353	719.361	717.296	717.644	710.476	703.785	700.408	696.512	703.009	716.749	731	748.642	759.376	766
300	20171225	765.625	771.923	761.279	755.003	751.832	742.995	729.489	710.996	712.269	728.394	775.491	786.956	825.256	870.786	877

Figure 20: NEM12 raw data example

User logs into the web-form and selects the: file ETL from the options tab, selects the validated NEM12 file and executes the ETL job as shown in Figure 21.

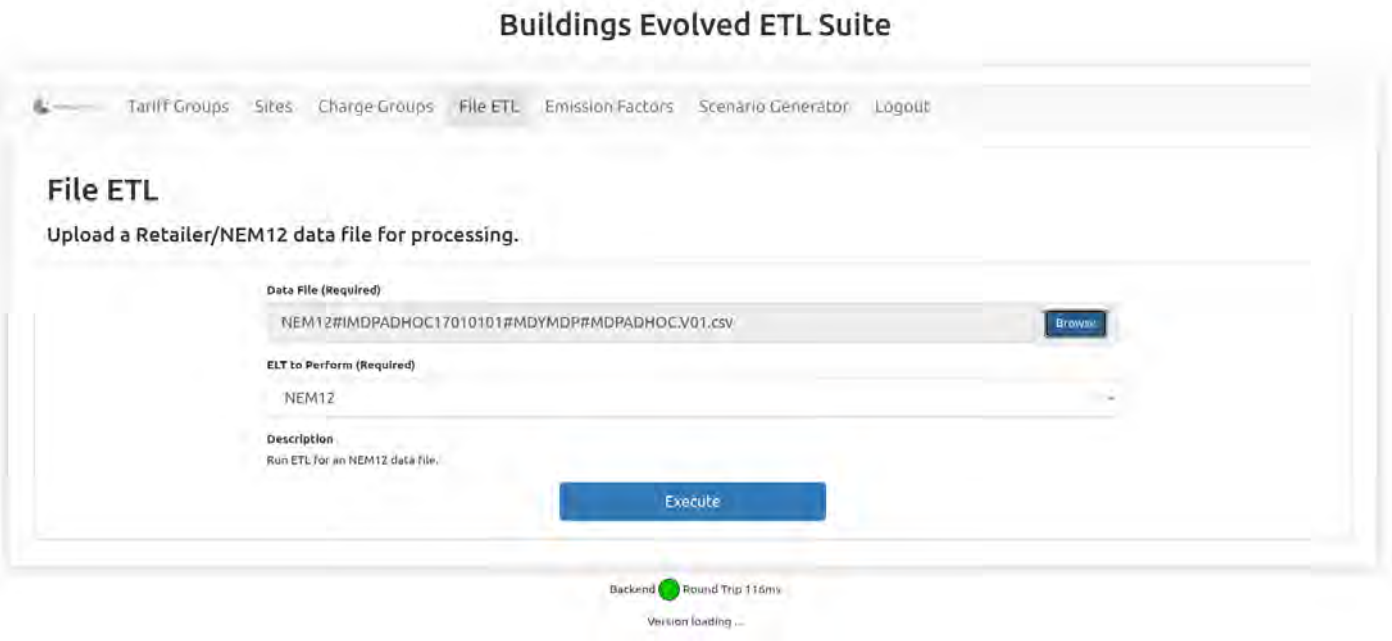

Figure 21: web form to upload and process NEM12 data

The cron operation picks up the NEM12 file in the queue and a Python script performs the ETL procedure to transform the file into a typical time-series format in the PostgreSQL database table.

Data Output | Explain | Messages | Notifications

	nmi character (10)	meter_serial_number character varying (12)	reading_date_time date	interval integer	power_factor numeric (4,3)	kva numeric (10,3)	net_kw_h double precision	net_var_h double precision	data_type text	interval_time character (8)
1	QB03674151	208002690	2018-05-13	0	0.994	1369.578	681.016	71.786	NEM12	00:30:00
2	QB03674151	208002690	2018-05-14	0	0.994	1128.893	560.905	63.13	NEM12	00:30:00
3	QB03674151	208002690	2018-05-15	0	0.993	1156.220	574.012	68.711	NEM12	00:30:00
4	QB03674151	208002690	2018-05-16	0	0.993	1156.965	574.361	68.931	NEM12	00:30:00
5	QB03674151	208002690	2018-05-17	0	0.991	1150.831	570.114	77.931	NEM12	00:30:00
6	QB03674151	208002690	2018-05-18	0	0.991	1176.601	583.258	76.861	NEM12	00:30:00
7	QB03674151	208002690	2018-05-19	0	0.993	1178.309	584.785	71.619	NEM12	00:30:00
8	QB03674151	208002690	2018-05-20	0	0.992	1173.775	582.183	74.162	NEM12	00:30:00
9	QB03674151	208002690	2018-05-21	0	0.992	1171.109	580.85	74.076	NEM12	00:30:00
10	QB03674151	208002690	2018-05-22	0	0.992	1191.439	590.79	76.479	NEM12	00:30:00
11	QB03674151	208002690	2018-05-23	0	0.993	1164.024	577.657	71.065	NEM12	00:30:00
12	QB03674151	208002690	2018-05-24	0	0.992	1167.564	579.011	74.484	NEM12	00:30:00
13	QB03674151	208002690	2018-05-25	0	0.992	1173.849	582.135	74.829	NEM12	00:30:00
14	QB03674151	208002690	2018-05-26	0	0.992	1164.897	577.662	74.517	NEM12	00:30:00
15	QB03674151	208002690	2018-05-27	0	0.992	1186.114	588.543	73.033	NEM12	00:30:00
16	QB03674151	208002690	2018-05-28	0	0.993	1142.180	566.942	68.705	NEM12	00:30:00
17	QB03674151	208002690	2018-05-29	0	0.993	1189.076	590.124	72.313	NEM12	00:30:00
18	QB03674151	208002690	2018-05-30	0	0.993	1159.719	576.038	66.463	NEM12	00:30:00
19	QB03674151	208002690	2018-05-31	0	0.994	1157.302	575.125	63.781	NEM12	00:30:00
20	QB03674151	208002690	2018-06-01	0	0.994	1173.523	582.974	66.56	NEM12	00:30:00

Figure 22: NEM12 after transformation, laid down in the psql database

The NEM12 ETL also calculates the daily totals for maximum demand, and tonnes of C02 for carbon reporting via a lookup into emissions factors under the National Greenhouse and Energy Reporting Scheme. Emissions factors can easily be added via the web-form as shown in Figure 23.

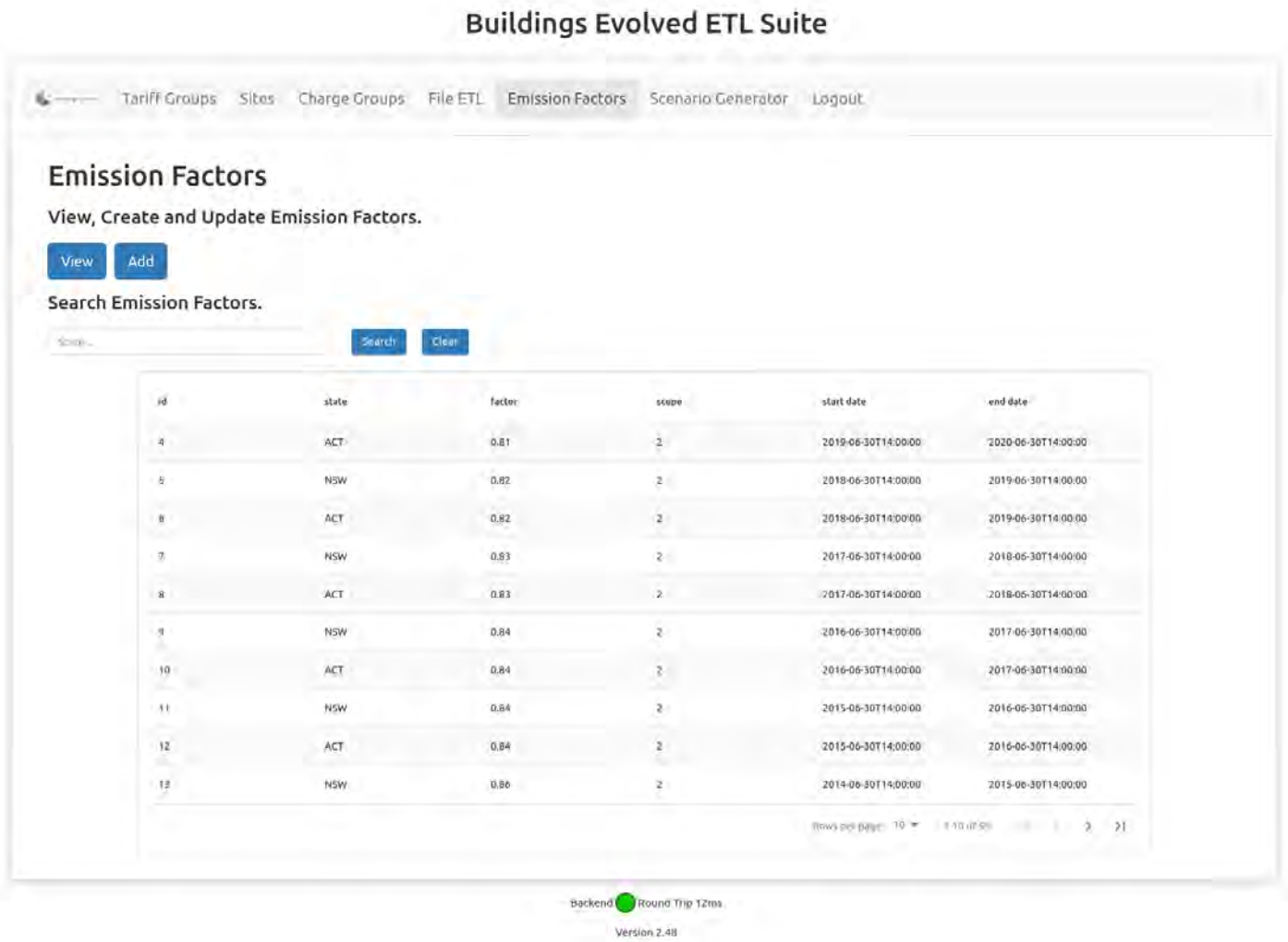

Figure 23: emissions factors set and reviewed via web page

The web-form designed by Buildings Evolved provides users with ‘one-click’ data ingestion of multiple data file formats relating to building energy management to produce the following outcomes:

- Validate data quality, format and type.
- Transform and normalise data in an appropriate schema for report outputs, and for energy management applications, such as:
  - Billing, and financial modelling of energy systems.
  - Time-series analysis of energy, and
  - Carbon reporting.

Being able to save time on energy analysis whilst ensuring information quality and completeness is crucial for organisations seeking to take ownership of building energy management initiatives, to save money, increase efficiency and reduce emissions.

4 ENERGY USE INTENSITY & PRODUCTIVITY KPIs

4.1 KPI 1: Energy Intensity

Key outcomes: Enable facility, operation managers and other stakeholders to understand electricity performance, otherwise known as energy intensity by comparing consumption per area, building and/or site to benchmarks specific to hospital facilities.

Key metrics and normalisation factors used: Kilowatt hour (kWh) – historical electricity consumption at the electrical meter (NMI) level, per area, building and/or site. And the following normalisation factors relevant to the electrical meter:

- Meter squared (m2) – floor area.
- Bed day – the occupancy rate of beds.
- Bed – the number of beds.

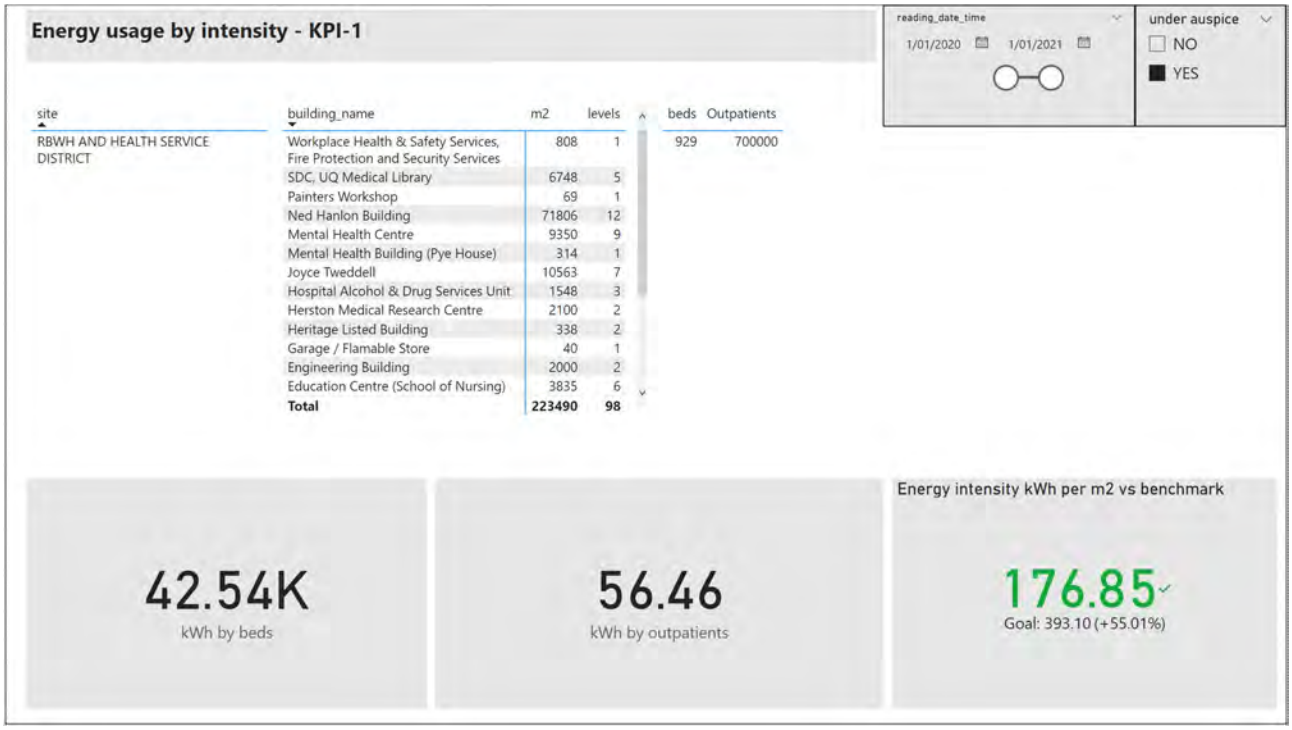

Figure 24: KPI 1 - energy usage by intensity

Figure 24 shows energy intensity of a NMI and site by m2, bed and outpatient normalisation factors against benchmark values. The top table shows breakdown of asset data and total consumption at an electricity meter (NMI) and/or site level. The tiles from left to right show the kWh total per m^2, outpatients, and difference to benchmark respectively per filtered period. The user can filter the data by electricity meter, site and/or building by date.

Technical method: Time series data in NEM12 format is procured and transformed into a relational database table whereby each row of data is an interval of time allowing for accurate historical analysis of energy consumption.

Energy consumption is then normalised by factors to evaluate energy intensity per m^2, bed and bed days (total bed volume – outpatients) simply by dividing consumption by each factor.

These normalisation factors are then compared against the benchmark numbers outlined in the 'Living Labs Healthcare Sector Energy Baseline and Key Performance Indicators' (baseline report)[18] set as a static value within the dashboard.

[18] iHub, Living Labs Healthcare Sector Energy Baseline and Key Performance Indicators, p15.

Comments on effectiveness: Energy intensity is a measure of the amount of energy used to produce an output, such as units produced, or in the case of a hospital, m2, beds and people serviced. Intensity factors function to normalise consumption figures by giving context. For example, one site or building may have a greater level of consumption than another yet be much more efficient because comparatively it uses less energy to provide a product or service.

The m2 benchmark goal used in this KPI gives an indication on how the site is performing compared to Australian capital city public hospitals, which have an energy usage intensity (EUI) of 393 kWh/m2. However, as outlined in the baseline report (cited above), the energy intensity normalisation does not indicate causation of variation, merely that a variation exists. For example, possible reasons for a variation in energy efficiency could be, climate, building and/or equipment age, building orientation, degree of medical specialisation etc.

Where to from here? More contextual data at hand would enable the comparison of energy performance between sites. Data from building management control systems, weather and telemetry systems as well as slow changing asset data will provide more context and allow further understanding on building performance. For example, HVAC accounts for 47% of electricity in Australian hospitals, understanding the effect of weather or temperature on building is indicative of performance:

- Buildings less effected by temperature may have more thermal mass and be suitable for pre-heating or cooling scheduling for efficiency.
- Or conversely, buildings more effected by temperature may be more suitable for a conventional control strategy.
- In another example, a HVAC system that functions near capacity constantly may be ageing and due for replacement.

In short, energy intensity and normalisation factors are useful in identifying poor performers, yet access to contextual data is crucial in verifying and understanding what causes poor energy performance and how it can be managed for improvement.

5 ENVIRONMENTAL AND SOCIETAL KPIs

5.1 KPI 2: Avoided GHG emission (tCO2-e and $)

Key outcomes: Ability to put a dollar value on tonnes of C02 avoided from energy efficiency measures per electrical meter (NMI), area, building and/or site.

Key metrics and normalisation factors used: Avoided GHG emission = (Baseline CO2-e – Reporting CO2-e) ± Adjustments, and Conversion from emission (tCO2-e) to societal benefit ($) uses an estimation of social cost of carbon from the baseline report literature (e.g. US$35/tCO2) as a conversion factor.

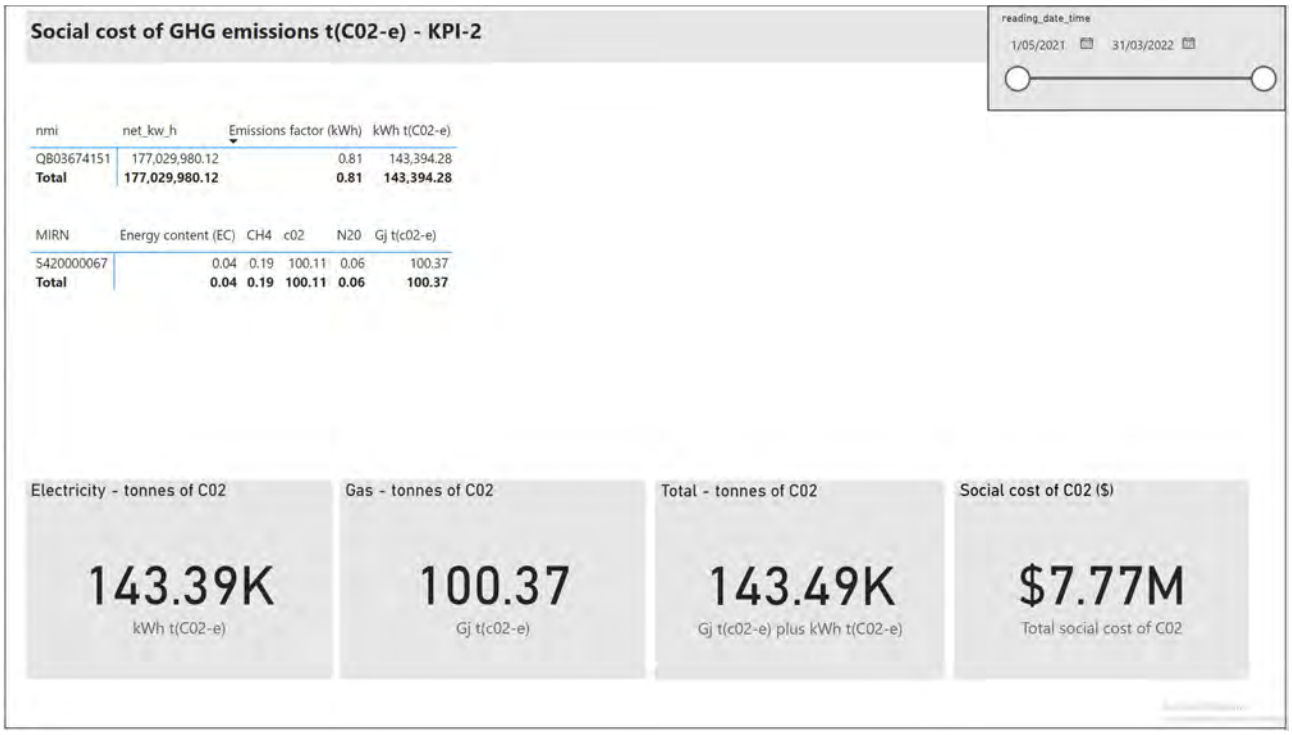

Figure 25: avoided greenhouse gas emissions (social cost of t-C)2-e)

Figure 25 shows the breakdown of tonnes of C02 at the electricity meter level, total tonnes, tonnes of C02, and the social cost of C02 from the baseline report. The user can filter the data by electrical meter (NMI), area, building and/or site per date.

Technical method: Time series data for electricity (NEM12 format) is procured in the same manner as KPI-1.

Interval gas data was procured in a daily summary format where each row of data contained total consumption per gas meter (MIRN) in gigajoules (Gj).

The emissions are calculated from total net kWh and Gj figures over the period of data transformed using the following calculations as per the REETSEF report[19]:

[19] iHub, Healthcare Sector: Renewable Energy and Enabling Technology and Services Framework (REETSEF), 2020, p13

Gas	Electricity
$GHG\ emission_{gas} = \frac{E_{gas,gross}\ x\ EC\ x\ EF}{1000}$ *Where:* *Q = quantity of fuel consumed (m^3);* *EF = the emission factor (kg CO2-e/kWh);* *EC = the energy content factor (kWh/m^3);*	$GHG\ emission_{elec} = \frac{E_{elec,net}\ x\ EF}{1000}$ *Where:* *Q = the quantity of electricity purchased (kilowatt hours);* *EF = the scope 2 emission factor (kg CO_2-e/kWh)*

Figure 26: emissions factor calculations

Emissions factors from the relevant National Greenhouse and Energy Reporting (NGER) were used in calculating tonnes of C02-e from each interval period for both gas and electricity data sources.

The social cost of energy was calculated using the US$35/tCO2 value derived from the Climate Institute report[20] converted to AUD.

Comments: Reporting of greenhouse gas emissions for medium to large organisations is mandatory under the NGER scheme. Many organisations are choosing to report on non-mandatory emissions sources that contribute to wider sustainability initiatives such as carbon neutral and net zero. Positive results above scheme thresholds is often used as a highlight topic in sustainability, quarterly and annual reporting.

Where to from here? Being able to account for carbon in a digital system without waiting for conventional methods such as billing cycles enables organisations to set targets and burn-down metrics in a dashboard. For example, Figure 27 below shows how sustainability initiatives could be tracked, for example, +- % of goal reached. This could be referred to at regular intervals to assist with managing initiatives relative to the goal.

[20] TCI, Social Cost of Carbon, 2014, https://www.yumpu.com/en/document/view/46810898/tci-socialcostofcarbon-policybrief-september2014

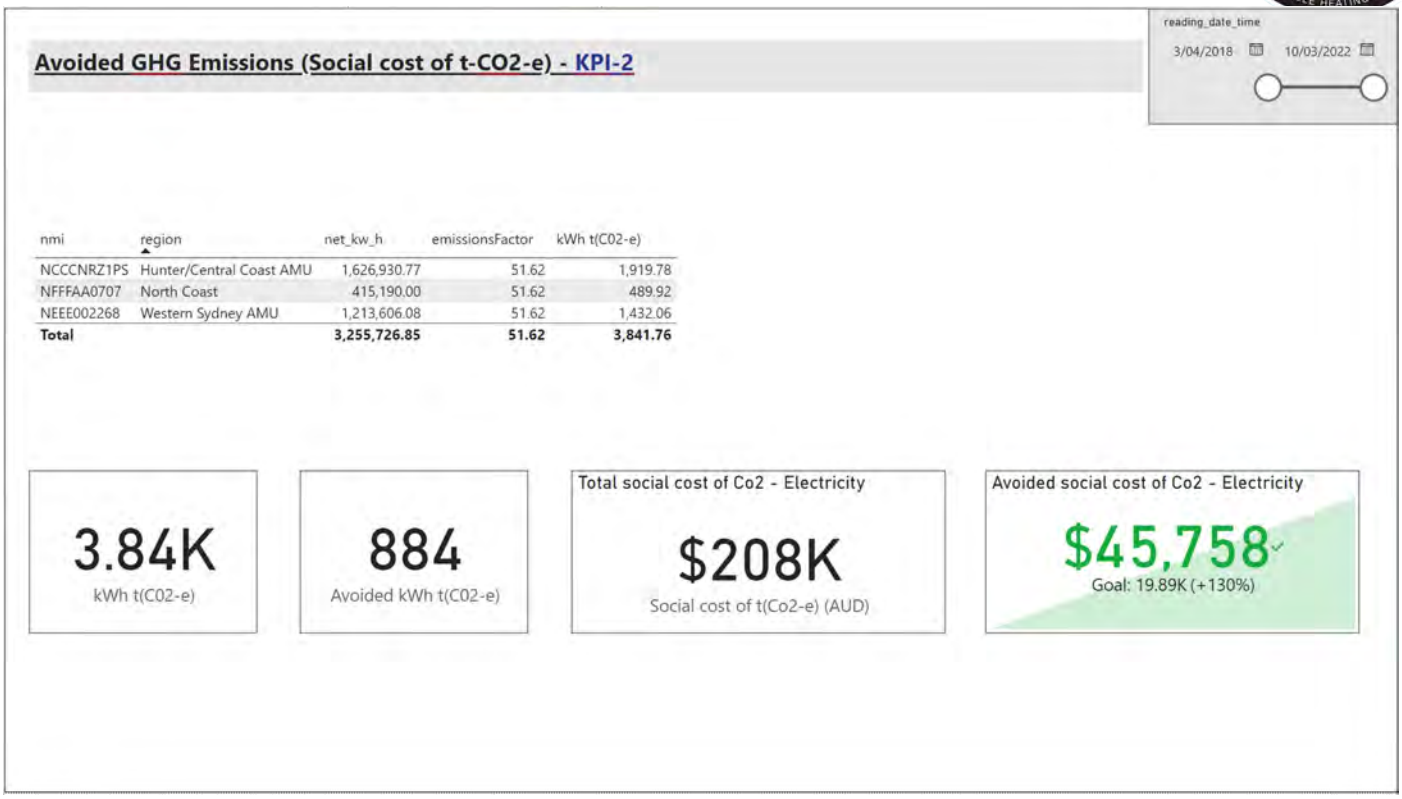

Figure 27: KPI 2 example with goals

Being able to account for carbon and emissions at the touch of a button, at any point in the reporting cycle is standard practice for leading property portfolio managers and owners.

5.2 KPI 3: Avoided air pollution

Key outcomes: Puts a cost value on air pollution (PM10, NOx, and SO2 gases) impacting populations close to power stations so that social benefit due to avoided air pollution can be quantified.

Key metrics and normalisation factors used: The calculation applies a damage benefit to each MWh (1000 kWhs) of energy saved per gas and as a total damage value. A m2 energy intensity benchmark is used to normalise and gauge performance.

Damage benefit factors[21]

Table 10 Damage costs for atmospheric emissions from the NSW grid

Pollutant	Cost $A/MWh
PM_{10}	0.173
SO_2	2.58
NO_x	2.76
CO_2	51.6
TOTAL	**57.1**

Figure 28: damage cost for atmospheric emissions from the NSW grid

Baseline energy use (energy intensity benchmark) from the baseline report[22] of 393.1 kWh per m2 p.a. was used.

Reported energy usage MWh from source.

[21] The Hidden Costs of Electricity: Externalities of Power Generation in Australia, 2009, p28

[22] iHub, Living Labs Healthcare Sector Energy Baseline and Key Performance Indicators, p15.

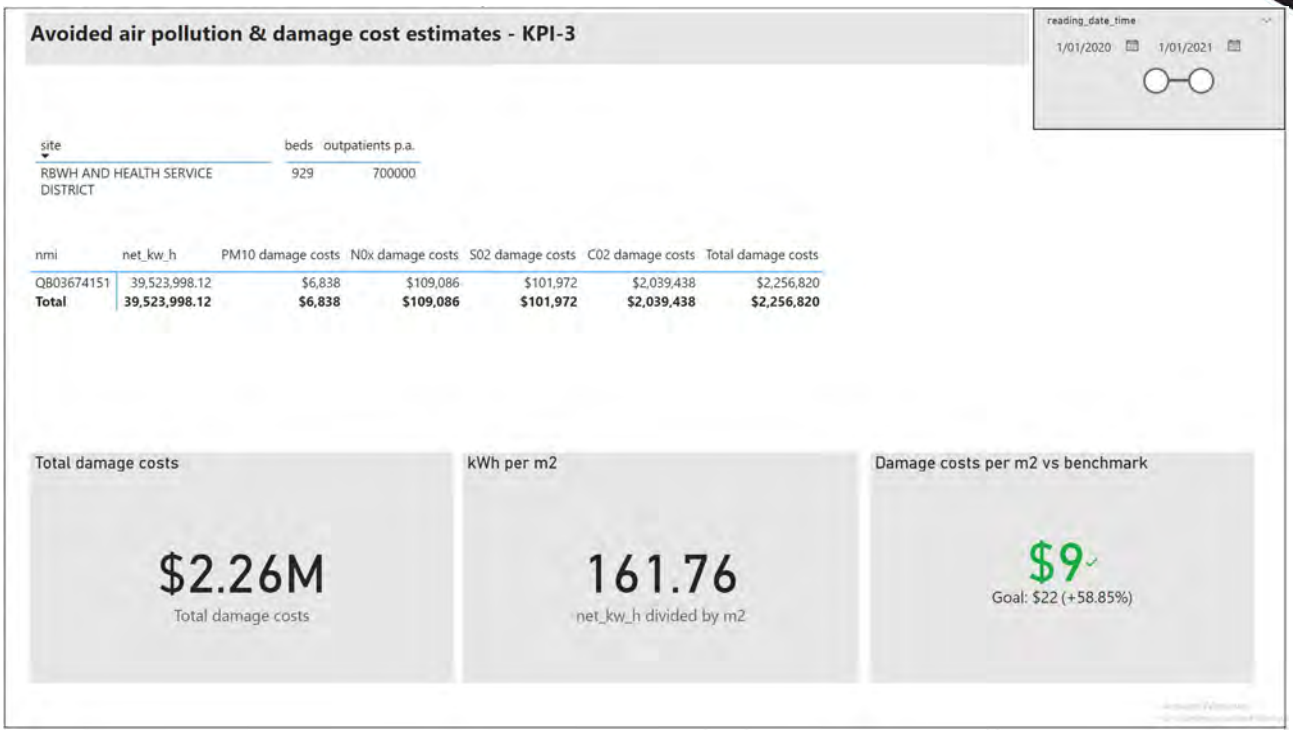

Figure 29: KPI 3 - avoided air pollution and damage estimates

Figure 29 shows the intensity of damage costs of electricity generation per gas type at the electricity meter level. The total damage cost is evaluated against a m2 benchmark for hospitals referenced in the REETSEF report.

Technical method: Time series data for electricity (NEM12 format) is procured in the same manner as previous KPIs.

kWh totals are multiplied by the cost of atmospheric emissions from the grid as:

Value of avoided air pollution = Damage Benefit Factor x Reporting Energy Use.

kWh totals are normalised by dividing kWh totals by total m2 giving a damage cost of gasses per metered area.

The difference in baseline and reported cost of gases is the value of avoided air pollution:

Damage Cost of Baseline (m2) - Reporting Energy Use (m2) x total m2 = Value of Avoided Air Pollution.

6 ENERGY NETWORK KPIs

6.1 KPI 4: Peak demand by period

Key outcomes: To demonstrate the ability of technology and renewable energy systems to support the electricity network by reducing demand at critical times.

Key metrics and normalisation factors used: kilovolt amperes (KVA) per period (measure of active power intervals).

1. Peak demand derived from KVA calculations at a daily resolution.
2. Date and time matched weather data from nearest weather station.

Figure 30: KPI 4 - peak demand by period

Peak demand (or maximum demand) is applicable to larger consumers of electricity appearing as a charge (usually in a $-dollar rate per KVA). Electricity consumers who can significantly reduce maximum demand can also reduce costs and demand on the electricity grid and network.

Figure 30 communicates the peak demand (or maximum demand) for a site per NMI. Maximum demand is generally calculated as the average between the two highest kilo volt ampere (KVA) intervals in a 12-month rolling period.

The line chart (top-left) shows the trend of maximum demand over time (in years) per electricity meter (NMI) and/or site. The box and whiskers chart (top right) shows the individual KVA readings that are outside a normal distribution and thus the significance of efficiency, such as peak lopping activities that would need to occur to reduce maximum demand. The closer the outliers are to normal distribution indicates less potential for peak-lopping and/or load-shifting benefits. The above chart shows several outliers that would indicate that this site would benefit from peak-lopping.

Average weather info for max demand periods is shown on the bottom right table and KVA normalised by m^2 is shown to the left. This enables easy comparison between sites or electricity meters.

Technical method:

1. Time series data for electricity (NEM12 format) is procured in the same manner as previous KPIs.

2. KVA is derived from processing electricity data files (NEM12 format) in the same manner as kWh.
3. The daily maximum demand (average of two highest KVA intervals) is calculated and written to a separate daily summaries table.
4. The max KVA figure is realised by setting the aggregation method within the BI tool to max, giving the max of the max demand table.
5. Temperature data is joined to electricity consumption data at the interval, date and time level.

Comments: Being able to manage demand at a site level requires monitoring and control of energy consuming equipment in real-time. This capability is achieved by bridging the operational technology to information technology divide. A modern real-time energy management solution will take sensor data from equipment, controllers and edge servers into the cloud for real-time cloud-to-device control whereby change of value readings are stored, alerts are triggered in the cloud environment and messages sent back to the edge server and controller to control devices in <1-second.

Effective demand management is not possible to be achieved manually, via human intervention. For example, by the time a message is sent, an individual will likely not have time to act before a peak demand event occurs. For an organisation, this may mean that their network demand charge is set for the entire year, or on the network, more base-load provisioning is required to ensure energy security.

Where to from here? Organisations that can demonstrate load flexibility, that is, curtail consumption or provide rapid generation in response to internal, network and market signals can reduce demand charges, participate in demand response programs, and FCAS (Frequency Control Ancillary Services) markets and receive benefit and income from these functions and services whilst improving energy security on the electricity network.

6.2 KPI 5: Wholesale cost of peak 30 minute electricity demand

Key outcomes: To determine whether the healthcare facility peak demand is co-incident with periods of network peak demand.

Key metrics and normalisation factors used: As price is representative of demand on the electricity network, KVA and time-matched (co-incident) wholesale spot price data is used to determine a relationship.

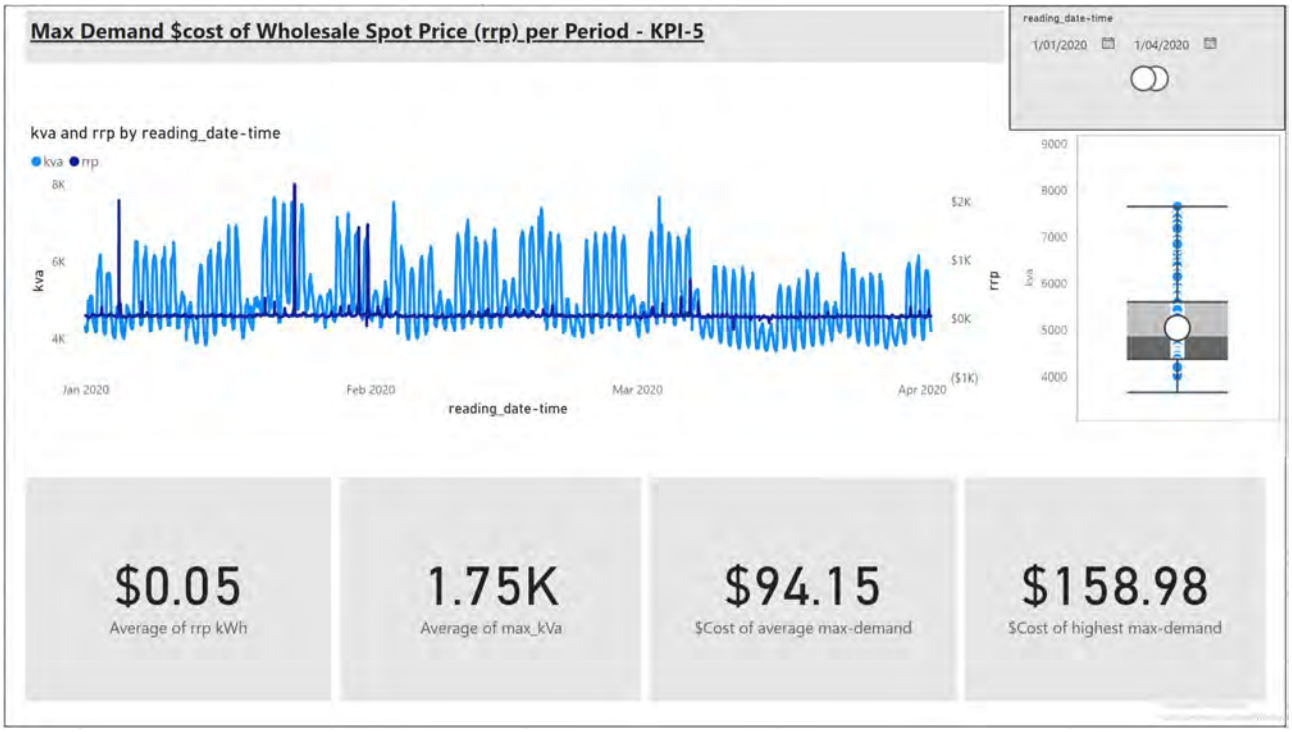

Figure 31: KPI 5 - max demand and coincident wholesale spot price

Figure 31 shows the relationship visually between active power (KVA) and the co-incident wholesale spot price by date and time by matching electricity KVA with AEMO data.

The line chart shows the date and time matched active power (KVA) and the AEMO spot price (rrp). The user can see where spikes in KVA and rrp occur.

The box and whiskers chart on the left shows the outliers from a standard distribution which enables a user to drill into peak consumption events and see where the coincidence price is at that point in time for comparison.

The tiles, or values down below show the average in terms of KVA and costs enabling the user to get a sense of the monetary effects and monetary risk associated with wholesale price exposure.

Technical method:

1. KVA is derived from processing electricity data files (NEM12 format) in the same manner as kWh.
2. Wholesale market price data (rrp) is collected from AEMO website via an application programming interface (API) get function to download data and transform it in a time-series format for date and time matching with electricity data.
3. The wholesale cost of peak demand at the time of occurrence is calculated using actual co-incidental wholesale spot prices. This is represented as an average figure using the aggregation function within the BI tool.

Comments: Other methods could be used to further establish the strength of the relationship between active power consumption or demand and the wholesale spot price. Such as a simple percentage difference calculation whereby values closer to 0 would indicate coincidence and reason to stick with traditional retail contracts. Whereas values above 100% variance would indicate that a site may benefit from variable pricing. Or alternatively, a regression model could be used.

Where to from here? Like KPI-4, benefits derived from active power management, such as peak demand and wholesale market participation requires real-time control. Smart controlled devices that are capable of rapid response to market conditions and pricing can replace the need for the retailer hedge, the protection against extreme price events on the wholesale electricity market.

On a financial level, coincidence between consumption at a site or electricity meter with wholesale price data might indicate that a site would benefit from a typical retail to customer scenario whereby the retailer protects the customer from extreme pricing events on the market. Whereas conversely, a lack of coincidence between consumption at a site with wholesale data indicates that a site may benefit from a variable pricing contract with exposure to the wholesale price[23].

From a network perspective, more load that is shed during peak and critical periods on the network means less base load provisioning, increased energy security and lower prices for consumers.

6.3 KPI 6: Total Self Consumption Rate

Key outcomes: To determine if electrical load could be shifted to times where solar self consumption is <1. In other words, could solar generation be utilised onsite before exporting to the electricity grid?

Key metrics and normalisation factors used: Solar export time series figures per NMI and site.

[23] Amber Electric, 2022, https://www.amber.com.au/how-it-works

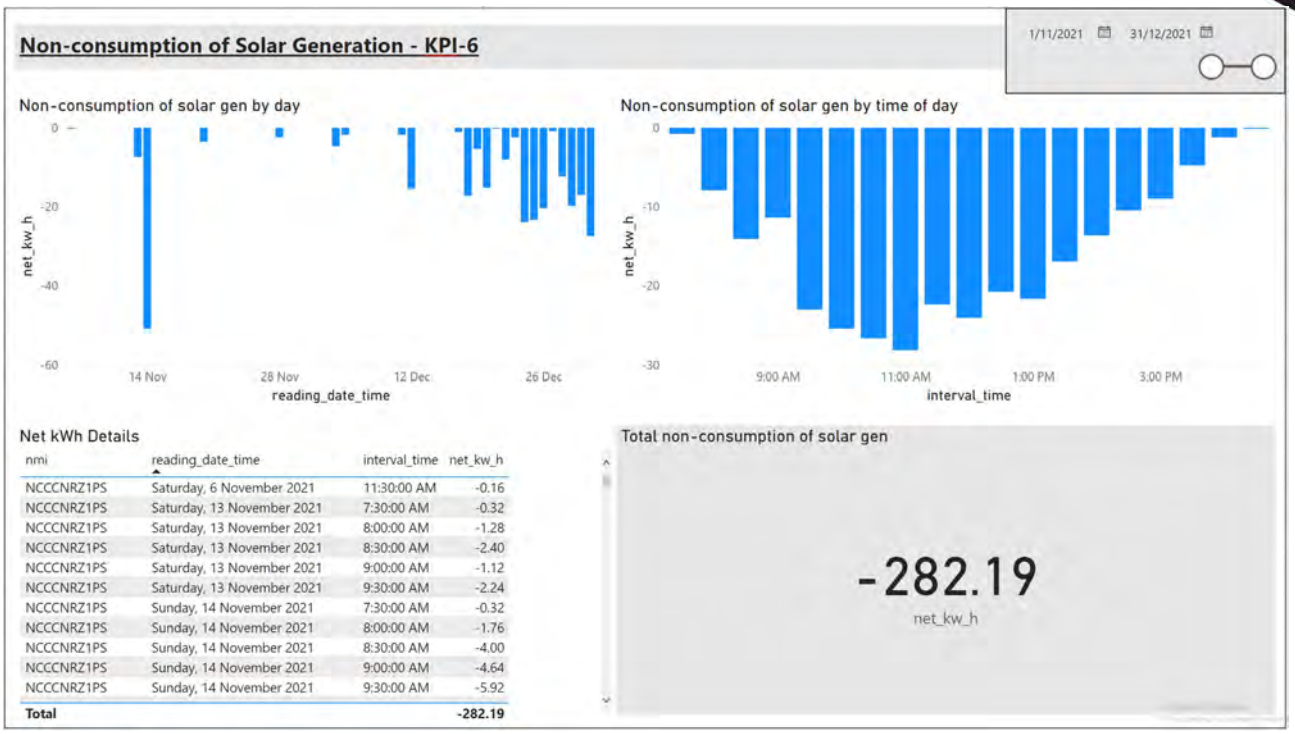

Figure 32: KPI 6 - non-consumption of solar generation

Figure 32 shows the non self-consumption (export) of renewable generation at the NMI and site. The chart on the top left shows the daily total non self-consumption periods. The chart on the top right shows the time-of-day, or intraday profile where non-consumption occurs. The table on the bottom left shows metadata, more details at the interval level. Whereas the tile on the bottom right shows the total non self-consumption in kWh over the filtered period.

Technical method: In lieu of solar generation data, negative kWh data was used from NEM12 electricity data files, procured in the same manner as previous KPIs. Binning time-series data by time allows for the different views of the same data, by day to intraday periods. Total non-self consumption is the aggregation of non self-consumption values by sum within the BI tool.

Comments: There was a lack of real solar data from inverters at the hospital site. Sample data was used to generate the dashboard in Figure 35.

Where to from here? Excess solar generation is generally wasted energy in terms of financial benefit due to declining feed in tariff revenues from traditional retail arrangements. Additionally, excess solar can produce frequency and voltage issues on the network. Modern inverters have Volt-Watt functions to reduce power output when grid voltage is high which limits any remuneration for excess energy generated onsite. However, being able to schedule and control loads with smart controlled HVAC for example, loads can be shifted to periods of self consumption and avoid the useless effects of the export constraint.

In another example, smart controlled networks can function as distributed energy resources, able to be co-ordinated to aggregate controlled loads, to dispatch electricity in MW units as a wholesale market participant or on contract in a variable agreement with an aggregation service provider.

6.4 KPI-7 HVAC Self Consumption Rate

Key outcomes: To determine if HVAC load could be shifted to times where solar self consumption is <1. In other words, could solar generation be utilised onsite before exporting to the electricity grid?

Key metrics and normalisation factors used: Kilowatt hour (kWh), historical electricity loads at the electrical meter (NMI) level, per area, building and/or site, specifically:

- Solar export;
- Disaggregated HVAC; and
- A percentage tolerance from total self-consumption is used as a goal.

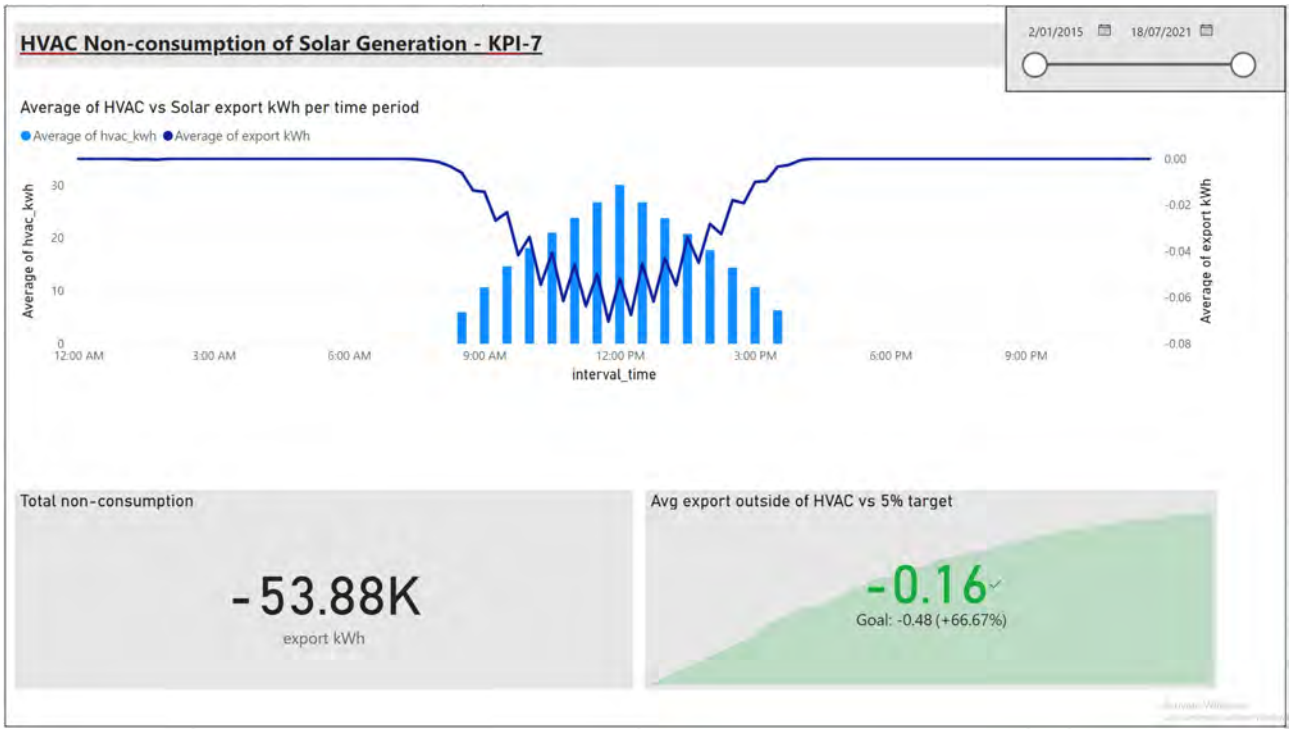

Figure 33: KPI 7 - HVAC non-consumption of solar PV generation

Figure 32 shows the time of use characteristics of average solar export and HVAC consumption, that is the time of day and average amount of excess generation being exported to the electricity grid, and the average HVAC load (derived from disaggregation) over the filtered period.

The bottom left card shows the total non-consumption value of excess solar generation being exported to the grid. The KPI tile on the bottom right shows the amount of solar export occurring outside of typical HVAC operation. A 5-percent tolerance from total self-consumption is used as a goal. A negative figure in this case is a positive result, indicating that a minimal amount of non-self-consumption occurs outside of HVAC operation. Therefore, little benefit would likely result from shifting HVAC loads to other periods of the day to maximise the consumption of solar generation onsite.

Conversely, if there was >5-percent export variance from HVAC then it might be beneficial to shift HVAC loads to periods where excess generation is occurring, if possible.

Technical method: Two intraday profiles are used to derive HVAC non-consumption of solar generation:

- average solar export, and
- disaggregated HVAC load.

Aggregation of time series data is required to produce the intraday load profiles.

A lack of solar data required a deviation from the REETSEF report method, involving, negative consumption, or export figures from NEM12 (kWh) time-series data used as a substitution for the conditional logic of:

> *If solar generation >0, then HVAC (kWh) / solar PV (kWh).*

An average of export time-of-use was binned by time within the BI tool to show the time-of-day (intraday period) that export is generally occurring.

HVAC loads are disaggregated from the base-load electricity using a fuzzy logic algorithm to understand the amount of flexible load at any given point in time. The algorithm uses a similar implementation to that of HVAC system controls, deciding when to heat or cool based on typical operation of buildings, such as occupancy, internal and external temperature conditions. These variables are used to input a rule structure of conditional statements and rule matrix that produce an output or conclusion to a rule[24].

Fuzzy logic implementation produced load profiles for baseload, heating and cooling (HVAC) time series kWh data at a 30-min granularity per NMI and/or site. Similar to the solar export average, an average HVAC usage was binned by time within the BI tool to show the time-of-day (intraday period) that HVAC is generally occurring.

The KPI tile functions to apply >5-percent export variance from HVAC via a time-of-use filter to yield solar export averages. The filter functions to quantify how much solar export (as a percentage of total) occurs outside of normal HVAC operation. A goal value of 5-percent was set as an acceptable tolerance of solar non-consumption considering HVAC operation.

Comments: As solar data was not available, a sample data set was used. As such the solar and disaggregated load profile is not derived from hospital data.

Where to from here? Commonly, HVAC data is metered along with other electrical systems. This makes deriving the HVAC load, or controllable load problematic. The Fuzzy logic method and others such as non-linear regression[25] can be used to derive and predict HVAC consumption into the future. These methods function to quantify the available controllable load used for energy modelling, that is, benefit analysis and scheduling in a sophisticated building energy management system capable of matching demand with supply and/or achieving other efficiency goals.

6.5 KPI 8: Net facility load factor KWh (avg) / peak KVA monthly, seasonally and annually

Key outcomes: Understand the load characteristics at a building and/or site at different time frames so that strategic plans can be made to manage electricity consumption.

Key metrics and normalisation factors used: Net Facility Load Factor requires kilowatt hour (kWh), historical electricity loads at the electrical meter (NMI) level, per area, building and/or site. The load factor calculation is simply the average load divided by the peak load during a specified time period.

[24] Encoder, Fuzzy Logic Tutorial, http://www.cassinis.it/Siti%20ex%20Uni/Sito%20webmail/Dida/2003/robb/Documentazione/Tutorial%20fuzzy%20logic.pdf

[25] Energy and Buildings, Cooling load prediction and optimal operation of HVAC systems using a multiple nonlinear regression model, https://www.sciencedirect.com/science/article/abs/pii/S0378778818337861

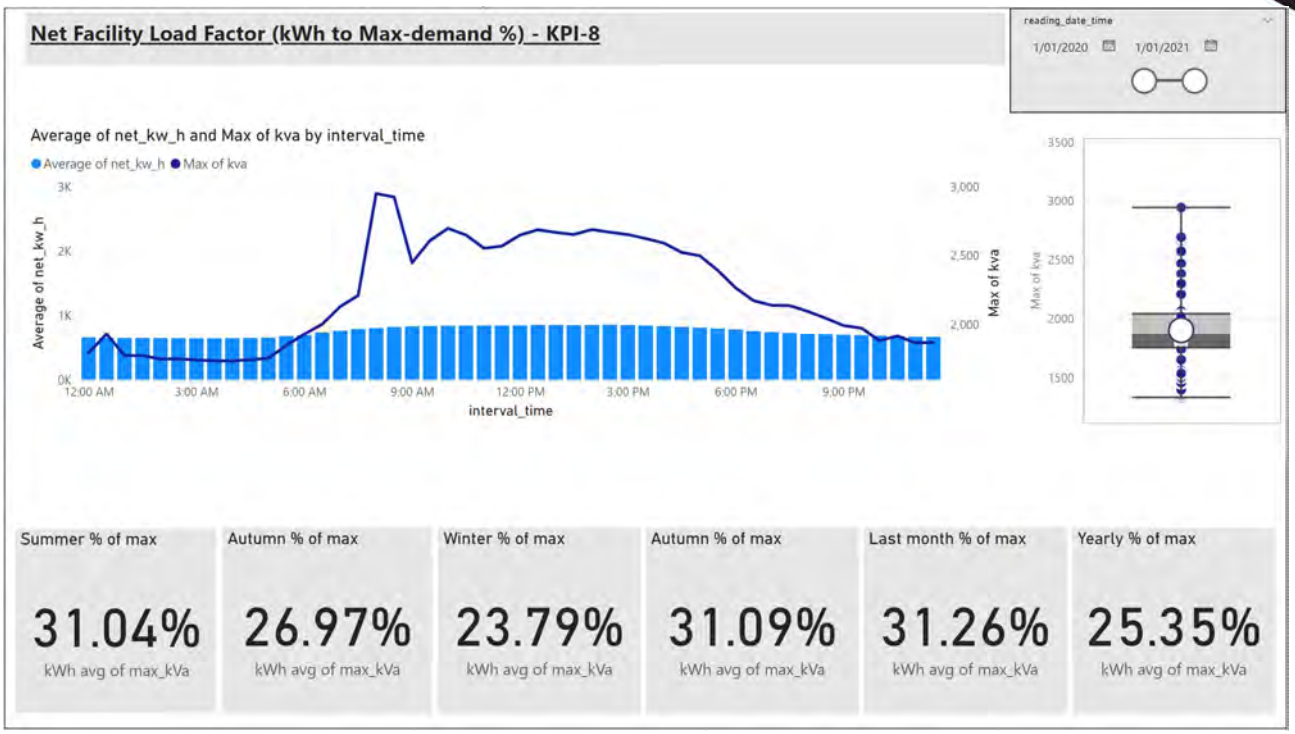

Figure 34: KPI 8 - net facility load factor (KWh to max demand %)

Figure 34 shows the average load divided by the peak load during a specified time period and is a measure of how 'peaky' an energy use profile is. The top bar chart shows average intraday energy consumption (kWh) in bars and the dark blue line is the daily maximum demand (KVA) per interval relative to historical consumption.

The bottom tiles show the percentage of average kWh compared to the max KVA value over the filtered period, seasonally and annually. A 'peaky' energy use profile might show higher percentages compared to the maximum demand.

Technical method: The net kWh time-series values are derived from daily kWh totals from ingested NEM12 electricity data. The daily max demand figures are calculated by taking an average between the highest intervals in a day. The yearly max demand, that is the figure used commonly by retailers to calculate demand charges, is simply the highest daily figure across a 12-month rolling window.

The load factor calculation is simply the average load divided by the peak load during a specified time period calculated as:

kWh (avg) / KVA (max)

Each tile shows this calculation as a percentage that is filtered; monthly, seasonally and annually per NMI.

Comments: It is of note that kWh and KVA are not 1:1. If for example a site has poor power factor (power quality issues), then the variance between kWh (historical) and KVA (active) power would increase. For this reason, it is suggested that an average KVA / max KVA, or a power factor conversion be used instead of kWh for this KPI.

Also of note is that the axis on the above chart are on different scales, thus they function to visually show a relationship rather than give precise scale of the two measures. This is due to the fact that the average kWh figures would appear very small on the chart if the axis were set to scale.

Where to from here? Like KPI-4, benefits derived from active power management, such as peak demand and wholesale market participation requires real-time control. Smart controlled HVAC systems alerted to spikes in KVA could ramp down and avoid setting a new high peak network price. In most network areas, peak or demand charges are charged at a $/max-KVA over a 12-month period. Typically, sites with smart controlled HVAC can reduce demand charges by up to 40%.

From the data in the above chart, it is evident that this hospital site would benefit from a peak lopping strategy. Max demand for the Year was set at around 8am (dark blue line), and the outliers in the box and whiskers chart on the right shows a significant variance from a standard distribution indicating that the event was not close to a normal or typical operation. In this case it is estimated that 30-40% of demand charge reduction could be achieved.

6.6 KPI 9: demand response capacity

Key outcomes: Determine load flexibility at any point in time. E.g. how much electrical load is able to be curtailed during different peak network periods summer, winter and at different time scales (e.g 6 sec, 1 minute, 10 minutes, 1 hour, 4 hour) whilst maintaining suitable operating conditions in line with occupational health and safety requirements, such as, ASHRAE standards and Indoor Environmental Quality (IEQ).

Key metrics and normalisation factors used: Kilowatt hour (kWh) – historical electricity loads at the electrical meter (NMI) level, per area, building and/or site, specifically disaggregated HVAC loads and a percentage reduction based on model predictive controls (MPC) from literature.

Price data is derived from daily cost of electricity as charged from the retailer.

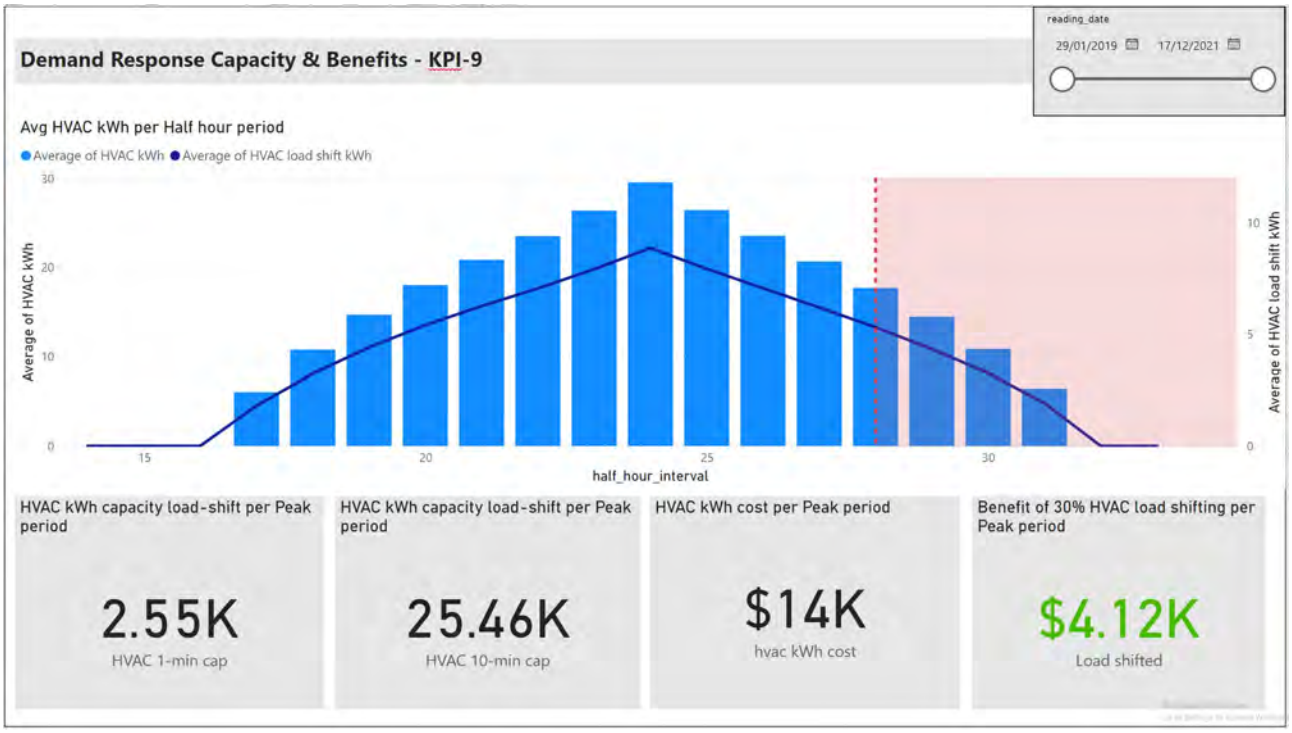

Figure 35: KPI 9 - demand response capacity & benefits

Figure 35 shows the disaggregated HVAC load as an average in an intraday period. The blue dotted line shows the potential for 30% demand response based on MPC algorithm operation[26]. Whereas the pink area shows the peak network time-of-use period relative to HVAC usage. The 30% reduction in the peak network period is quantified in kWh capacity and total $-dollar value. the KPI cards from left to right, respectively quantify:

- 1-min capacity (kWh),
- 10-min capacity (kWh),
- total cost of HVAC in the period, and
- benefits in dollars (avg) derived from delivering total capacity.

Technical method: Time series data for electricity (NEM12 format) is procured in the same manner as previous KPIs.

[26] Godina, Rodrigues, Pouresmaeil et. al., 2018, Model Predictive Control Home Energy Management and Optimization Strategy with Demand Response.

The technical method to derive disaggregation of HVAC is the same as KPI-7.

A simple 30% reduction in consumption is applied based off the literature in lieu of site specific data. With site specific data, an MPC algorithm can schedule optimised control of HVAC by:

- sampling the baseline consumption figures;
- understanding the nameplate electrical HVAC load; and
- obtaining interval temperature and humidity data from the BoM.

Steps in the algorithm involve:

- Classification of weather and temperature conditions,
- using inputs to produce regression a multivariate regression analysis of the demand flexibility inherent in the HVAC systems;
- identify high humidity but moderate temperature conditions (setting dry mode) and adjust consumption accordingly;
- load shift forecast HVAC peak demand for extreme temperature days using various strategies including:
  - pre-cool and pre-heat;
  - night purge; and
  - altering set points based on a thermal comfort model such (ASHRAE).

Comments: The resulting savings estimation is just an example and not representative of any specific results relating to hospitals, as:

- Sample disaggregated HVAC data was used, subsequently the average HVAC load is not typical of hospital load.
- Network time-of-use periods are different for each network area, thus the network peak period (in red) are used as a guide.

A simple 30% efficiency from building energy management, model predictive control algorithm (MPC) is used to quantify benefits.

Where to from here? MPC, in its application, controls HVAC via a smart controller onsite (programmed with the algorithm) in real-time, in a continual process of iteration. Each daily cycle, feedback is received and processed for the next iteration. Typically, buildings with more thermal mass, that are less effected by weather fluctuations are more suited to MPC whereby spaces can be pre-heated and pre-cooled based on forecast data and other site specific variables.

Modelling work conducted by Buildings Evolved in the DCH6.1 project[27] functioned to quantify the benefits of MPC for the project beneficiary, the Department of Education NSW.

6.7 KPI 10: Energy Cost

Key outcomes: To be able to quantify the cost of energy usage from various sources. In addition, to quantify the benefit of wholesale market pricing.

Key metrics and normalisation factors used: Kilowatt hour (kWh) – historical electricity loads at the electrical meter (NMI) level, per area, building and/or site. Daily gas consumption in gigajoules (Gj).

Price data is derived from daily cost of electricity and gas as charged from the retailer.

Wholesale data is derived from average wholesale price per period in analysis.

[27] hub, DCH6.1 Project, https://www.ihub.org.au/dch6-1-data-clearing-house/

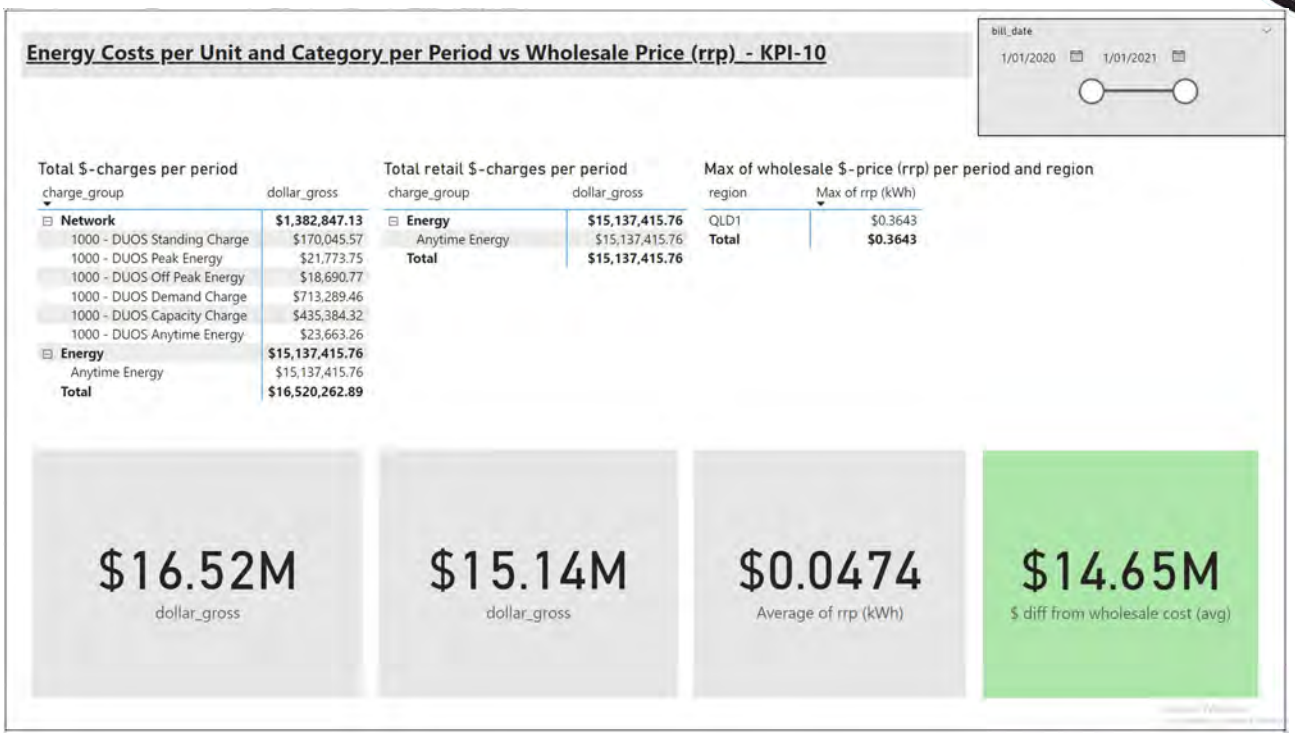

Figure 36: KPI 10 - energy costs per unit and category per period, vs wholesale spot price

The top table in the dashboard above shows the breakdown of electricity costs per type and transaction. The average wholesale price of electricity (rrp) is also provided.

The bottom cards show the breakdown of:

- total electricity charges,
- the retail charge component total, and
- the comparative cost of electricity on the wholesale market.
- The far right tile shows potential savings on a hypothetical wholesale pass-through contract that effectively replaces the retail charge component. Note that benefits are not indicative of actual benefits for this site.

Technical method: Electronic files from a retailer are loaded into a web-form. Files are parsed and transformed on a row by row basis according to their type and transaction variables. Results are written to a relational database table. This table is queried by the BI tool to produce the table and charges. Note that dollar amounts are derived from demo data and are not indicative of any specific site.

Wholesale market price data (rrp) is collected from AEMO website via an application programming interface (API) get function to download data and transform it in a time-series format for date and time matching with electricity data.

The wholesale benefit KPI is simply the difference between the retail component of costs – the average spot price (rrp) on the wholesale market over the filtered time period.

Comments: Organisations who can analyse bills in an electronic format from retailers, suppliers and from time-series data can reconcile costs on-the-fly, and manage budgets without waiting for billing cycles.

Where to from here? A digital billing engine can take billing files from a retailer, to display data on-demand in dashboards (as above) that shows the breakdown of consumption and costs. This data can be compared to a digital tariff engine model whereby energy tariffs are programmed via an ETL to perform one-click tariff modelling from NEM12 data files. An example of Buildings Evolved tariff engine model in Figure 37.

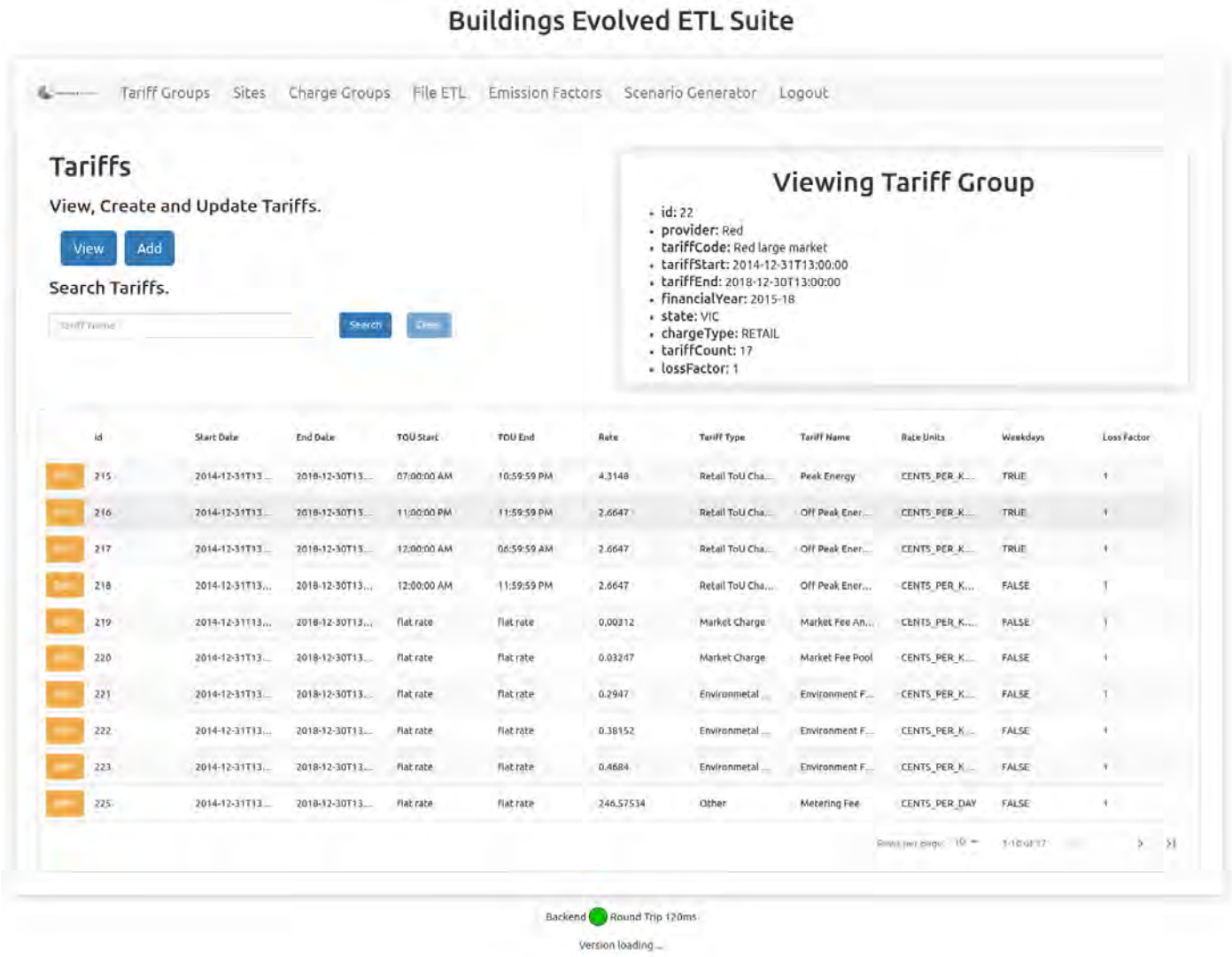

Figure 37: buildings evolved tariff engine software

An organisation equipped with such as solution can:

- Keep retailers honest by reconciling bills against consumption at any point in the billing cycle.
- Perform tariff audits easily, comparing offers from other retailers without manual spreadsheets and time consuming reconciliation and aggregation tasks.

Electricity retailers employ complex billing structures and mistakes are common. The ability to reconcile bills from NEM12 data as a source of truth enables organisations to reconcile charges against actually consumption and prove where and when mistakes arise.

7 RAPID ASSESSMENT: REETSEF KPI

7.1 Breadth

The REETSEF reports are a valuable method of assessing the renewable energy integration capability of hospitals. The relative ease of acquiring utility data allows a rapid assessment of building/campus performance, allowing generation league tables of good and poorly performing buildings, thereby allowing resources to be focused on the worst performing buildings. This is where the REETSEF report provides a huge amount of value.

By itself, or in isolation, the REETSEF report does not add a large amount of value. Normalised figures such as energy intensity per bed are meaningless without context from similar building topologies. The definition of ‘good’ is statistically defined by being above the mean.

7.2 Depth

The high-level REETSEF reports completed, and the poorly performing buildings/sites identified, the next steps are to delve into machine data for each building, derived from the Building Management (& Control) System (BMCS). Traditionally, this has been enormously complicated due to differing BMCS, different naming conventions, data formats, resolutions. Everything becomes bespoke per site, making this analysis impossible for all but the service agent of the HVAC systems themselves.

The Data Clearing House (DCH), coupled with a metadata schema such as Brick, will allow consistent models to be queried for the underlying datasteam name. The DCH 9 project shows that data from a BMCS can be rapidly onboarded to the DCH allowing analysis at depth to be undertaken, and faults and issues identified at a granular level.

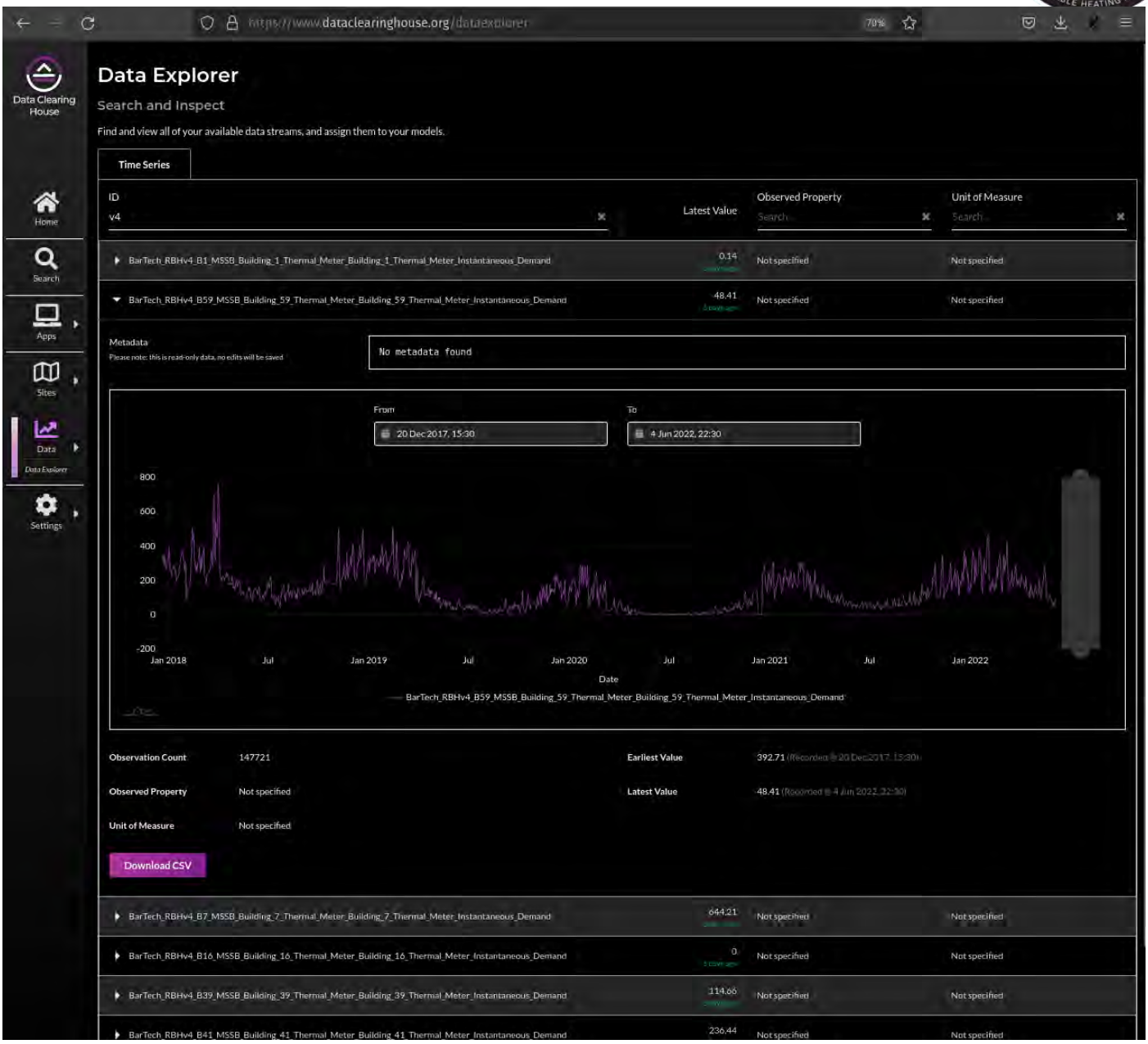

Figure 39: thermal meter data from MNH in the DCH

7.3 Summary

The range of KPIs are simple to use indicators and normalisations that help owners and operators of renewable and related energy technologies better understand and plan to manage assets in a changing and challenging operational environment.

The KPIs have been demonstrated on the Royal Brisbane and Women's Hospital using a limited data set. Slow changing asset data from buildings such as electrical meter (NMI) to building relationship was unverified. Information on HVAC systems from the building management control system was unavailable. Likewise, weather and telemetry data at the hospital site was unavailable. Sample data sets and were used to substitute for accurate contextual information. Thus, proper analysis at the hospital level and confidence in the KPI output was limited. Greater data

availability and validation of assets and their operation will improve the results. More buildings and accurate data will demonstrate the effectiveness and maturity of KPIs for validation, feedback and improvements.

Regarding energy intensity, more contextual information will improve the quality and effectiveness of indicators. For example, m^2 is not sufficient to raise awareness of poor performance at a site compared to a benchmark without considering factors such as; climate, building and/or equipment age, building orientation, degree of medical specialisation etc. It is only when these factors are taken into consideration that a benchmark can be made and buildings can be effectively normalised for comparison against the benchmark. Therefore, energy intensity using a single normalisation factor is a shallow dive or precursor for further investigation. However, in many instances, as was the case in this project, more information is not readily available.

Once effective normalisation occurs, poor performance can be understood, benchmarks established and a more meaningful comparison between sites can occur. Once poor performing sites are identified a deeper dive into operational technology, data from HVAC systems and/or the BMCS should be sought to determine causation.

Regarding KPI metrics, aggregations of time-series data, such as consumption divided by max consumption are simple measures to implement and useful to understand as they align with network charges and indicate potential for improvement. Likewise for renewable energy generation, however the building typology of hospitals has less potential for generation as they are generally multi-level buildings with less roof space.

In summary, KPIs are a measure of performance, what happens once owners and operators are aware of performance issues? It is likely they face the following challenges:

- Having access to timely, accurate data; from billing, time-of-use (time-series), integration of BMCS, to slow changing asset data.
- Digitalising and automating systems pro-grammatically, removing the need for manual tasks and eliminating the potential for error.
- Realising operational to information technology integration, real-time management and programmatic control from device-to-cloud and cloud-to-device signals, alerts and alarms, fault detection, algorithms such as MPC to match demand with supply for a balanced network with improved security at reduced cost.

Once organisations have realised capability and integrated technology they can begin to improve energy productivity and realise more value from renewable energy technologies, reduce demand on the network, reduce emissions and cost to all energy consumers.

8 DCH ONBOARDING WORKFLOW AUTOMATION ASSESSMENT

8.1 Perspectives on DCH workflow integration

The current implementation of DCH is very complex requiring an understanding of a lot of internal processes. There should only be one point of truth for an end user, no matter the complexity behind the scenes. Bar-Tech is of the belief this is planned in coming upgrades as part of the DCH and this would be needed to enable development houses a reasonable time to interface with the system. Privilege management is a complex web of three separate systems all requiring their own Users and privileges to work. At each step, many hours were used up finding issues on both sides of the system to enable the other side to troubleshoot user access issues. This was the most time-consuming part of the process and was a major contributor to not being able to get the data in time for BE to complete their work in a timely fashion.

```
@prefix ac:    <ac:> .
@prefix iam:   <iam:> .
@prefix bricx: <https://dataclearinghouse.org/schema/Brick/extension#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dch:   <dch:> .
@prefix senaps: <http://senaps.io/schema/1.0/senaps#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tag:   <https://brickschema.org/schema/BrickTag#> .
@prefix brick: <https://brickschema.org/schema/Brick#> .

<dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A_Zone_1_Face_Bypass>
        a               brick:Damper ;
        rdfs:label      "RBHv4/B40/Ground/MSSB/AHU-G-1A/Zone 1/Face/Bypass" ;
        brick:hasPoint  <dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A_Zone_1_Face_By
        brick:isPartOf  <dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A_Zone_1> .

<dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1B_F_B_Zone_2_Heater_Demand>
        a               brick:Heating_Command ;
        rdfs:label      "RBHv4/B40/Ground/MSSB/AHU-G-1B/F/B Zone 2/Heater/Demand" ;
        senaps:stream_id  "BarTech_RBHv4_B40_Ground_MSSB_AHU_G_1B_F_B_Zone_2_Heater_Demand" ;
        brick:isPointOf   <dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1B_F_B_Zone_2_H

<dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Basement_MSSB_AHU_B_2_Filter_Dirty>
        a               brick:Change_Filter_Alarm ;
        rdfs:label      "RBHv4/B40/Basement/MSSB/AHU-B-2/Filter/Dirty" ;
        senaps:stream_id  "BarTech_RBHv4_B40_Basement_MSSB_AHU_B_2_Filter_Dirty" ;
        brick:isPointOf   <dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Basement_MSSB_AHU_B_2_Filter> .

<dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A_Filter>
        a               brick:Filter ;
        rdfs:label      "RBHv4/B40/Ground/MSSB/AHU-G-1A/Filter" ;
        brick:hasPoint  <dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A_Filter_Dirty>
        brick:isPartOf  <dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A> .

<dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A_Zone_1_Room_Temperature_1>
        a               brick:Zone_Air_Temperature_Sensor ;
        rdfs:label      "RBHv4/B40/Ground/MSSB/AHU-G-1A/Zone 1/Room Temperature 1" ;
        senaps:stream_id  "BarTech_RBHv4_B40_Ground_MSSB_AHU_G_1A_Zone_1_Room_Temperature_1" ;
        brick:isPointOf   <dch:org/buildings_evolved/site/RBWH/building/RBH#RBHv4_B40_Ground_MSSB_AHU_G_1A_Zone_1> .
```

Figure 39: builidng metadata schema - a turtle (ttl) file generated programmatically to ingest data to the DCH

In addition to the complexity of getting data to DCH was the complexity of getting it out of the existing onsite system, the main requirement of this project. There were two main issues with this. The first is some complex naming conventions used to create “meta information” for the technicians complicated the ingestion process adding addition time to come up with solutions, around what the naming conventions meant to the brick model. These were overcome but required several testing iterations given rules changed between buildings.

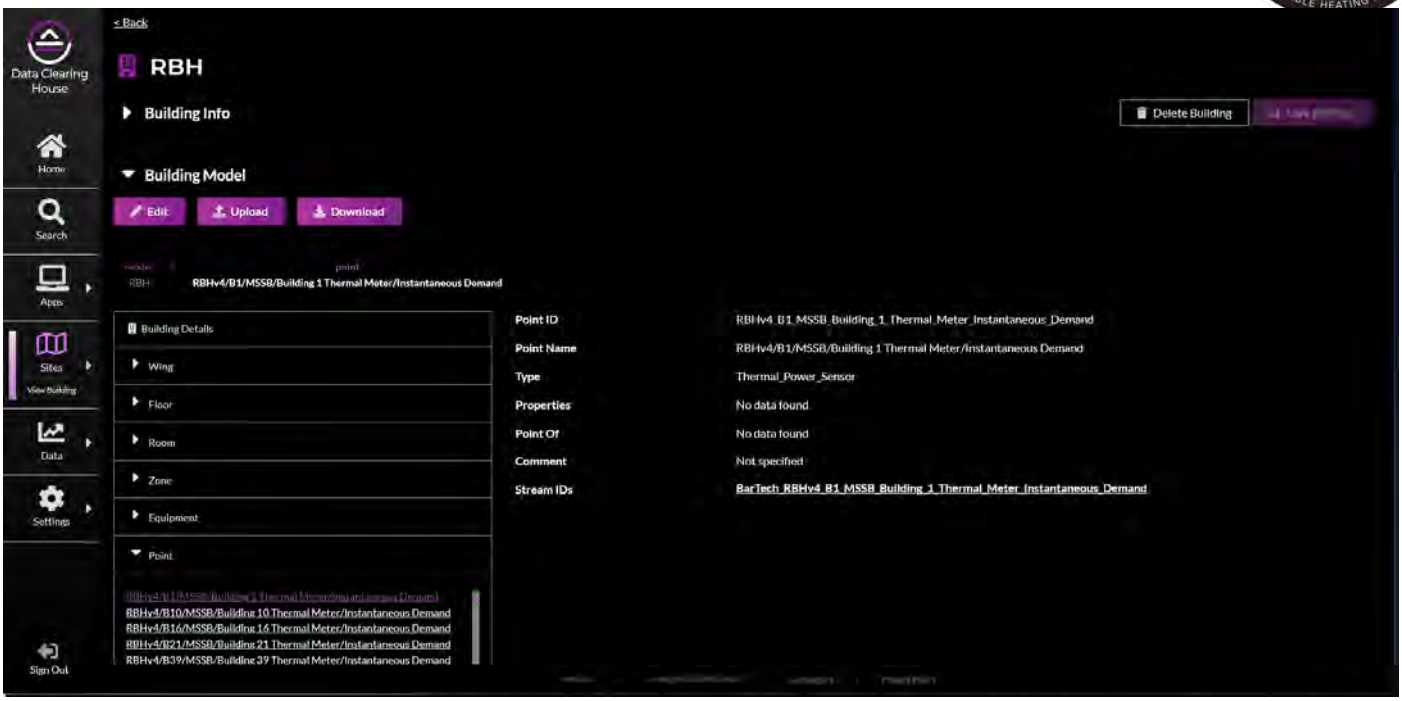

Figure 40: hospital building metadata schema rendered in the DCH

The second was Niagara and JAVA's lack of good memory management. This resulted in multiple stoppages and manipulation of data queries to slowly get the required data instead of getting large chunks as would be the standard approach. We required getting only one month of data at a time per history else risk instability of the overall platform. Refining this automation took several days to complete and make reliable. This is now working reliably without any impact on speed.

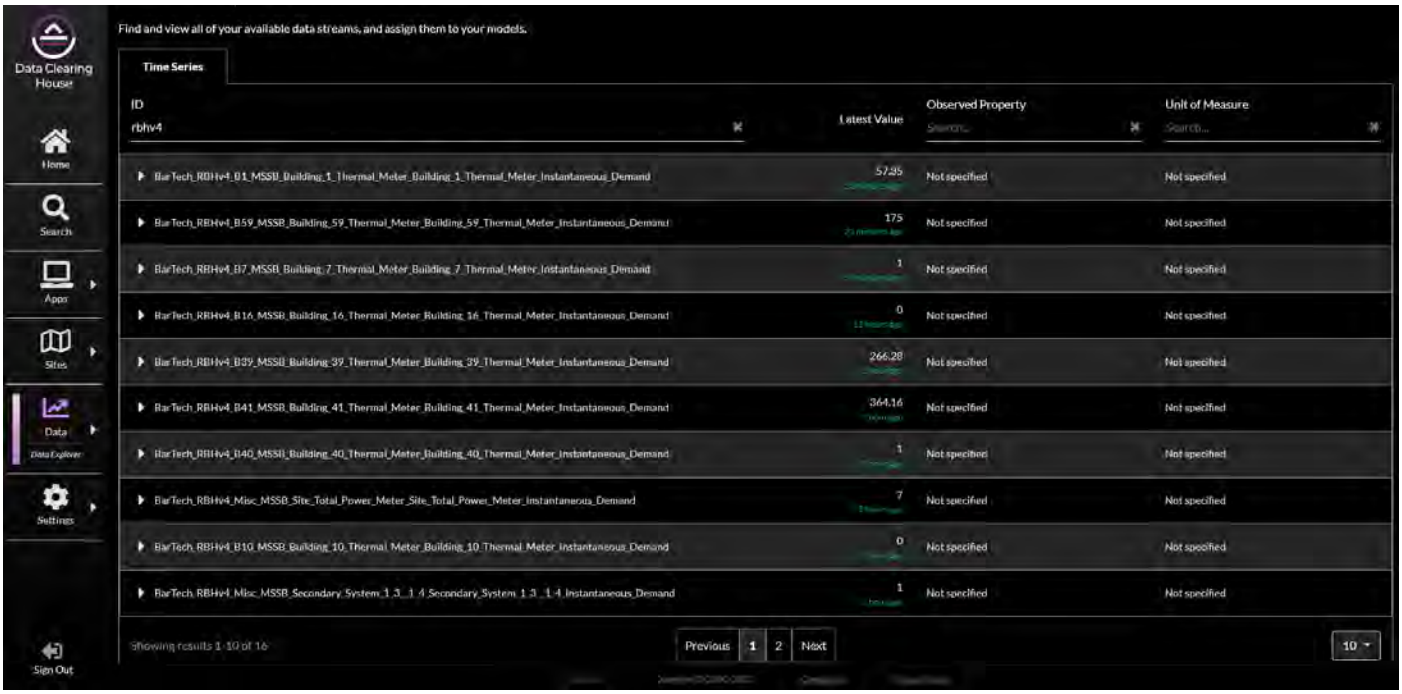

Figure 41: list of data points in the DCH for MNH

.

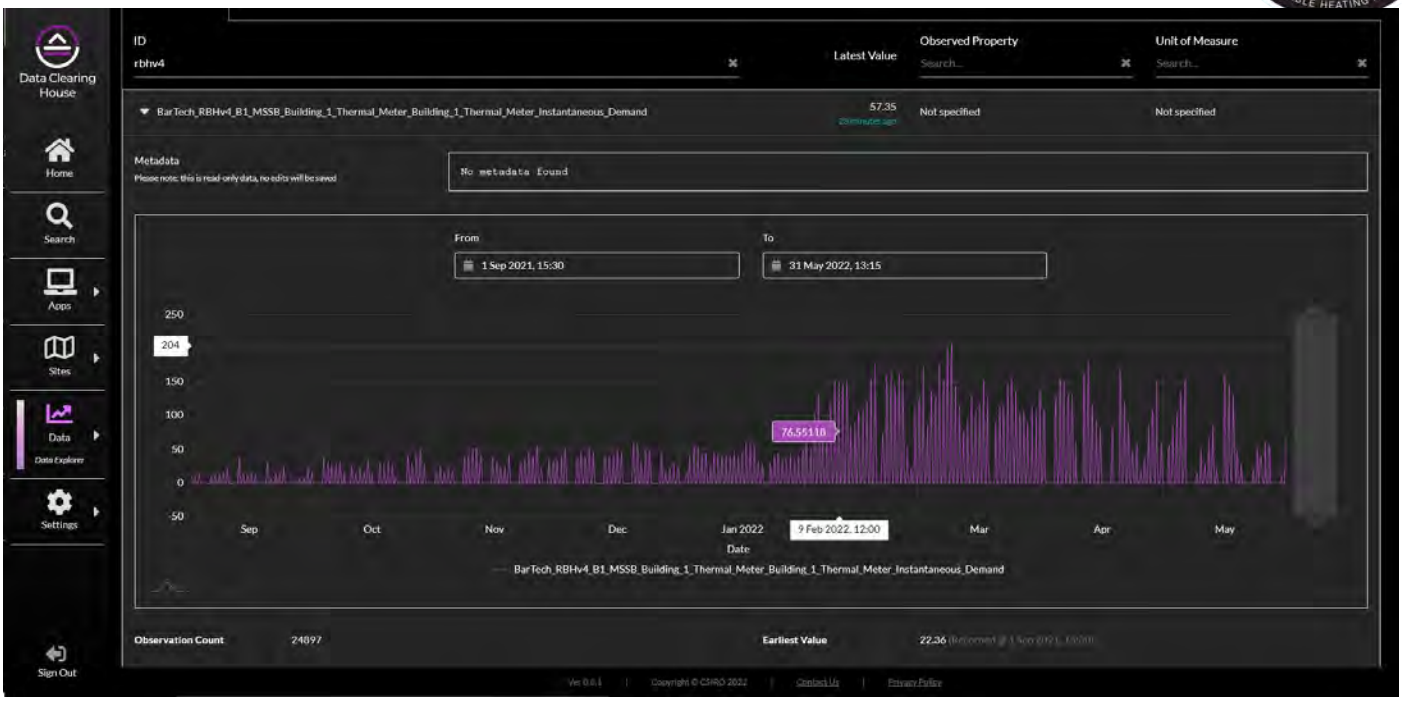

Figure 42: individual builidng monitor time series data, rendered in the DCH

8.2 Further DCH development and requirements

Overall the project from our perspective was a success and subsequent projects both at RBWH and on other sites would be greatly improved with all the learnings of this project. We believe ingestion of most sites can now be done very quickly. We have only added limited units in RBWH to test out the process given the limited time. We have ingested an additional test site which has a number of buildings but only completed 2 of those buildings resulting in over 1200 points quickly ingested. All these points are updating to the DCH and are a maximum10 minutes out, as they are monitored every 5 minutes this means that the data DCH is getting might only be 5 minutes since the recording period. Although this is not up to the minute this can be improved. Given the short time frame of the project many obstacles were created by that meri fact yet most of Bar-Tech's final requirements were met. Unfortunately, they were not met in time for reporting by Buildings Evolved (BE) or any meaningful evaluation of the OpenBMCS product from outside of Bar-Tech. Bar-Tech does though have numerous insights that will place future projects in good stead.

Bar-Tech believes the process of data ingestion is now complete. The next projects for Metro North Health should focus on being outcomes-based. These projects should focus on what the DCH can deliver from the data collated from the BMS's and what additional equipment should be monitored to get more informed results from the DCH and external operators.

For example, having energy monitoring for all the hospital buildings would allow the DCH and BE's Business Intelligence dashboard to widen the scope of and depth of reporting the current energy profile for the MNH building estate in its entirety. This will require a small investment in the meters for each building as Bar-Tech have controllers that can monitor these meters close by. The limited kWr is not accurate enough to fully validate the iHub report or give actionable data to MNH but will give them insights. Bar-Tech would assume there will be a lot more information that will be available given the separation of the buildings with their energy data. This would also be critical going forward into an environment focused on greenhouse gas reductions.

Other algorithms that could be used now are Automatic Loop tuning and energy validation of A/C Equipment, Optimum Scheduling, load shifting and many more. All these algorithms are not standard practice in BMCS but are invaluable for energy efficiency in this new data enabled environment.

9 CONCLUSION & FUTURE OF DCH AT METRO NORTH HEALTH

9.1 Building maintenance engineering & facilities

Metro North Health can achieve strategic goals of tracking and optimising energy performance against REETSEF KPIs to inform the requirements for future enterprise BMS implementations, enable greater collaboration with research and industry partners by providing data to third parties using role-based permissions and evaluate the overall value and security of the DCH. In addition to achieving these strategic goals, Metro North Health would be better able to achieve operational goals related to energy performance optimisations, including balancing energy usage efficiency against achieving Indoor Air Quality targets, maximising asset effective-life spans through targeted upgrades of infrastructure that are no longer performing efficiently, and determining actual HVAC loadings and requirements (ie, for ensuring supporting infrastructure for Business Continuity is adequate based on an understanding of “real-world” HVAC usage.)

9.2 Sustainability, assets & infrastructure

Metro North Health is currently delivering the Green Metro North – Sustainability Strategy 2021-2026[28], which includes a commitment to take action towards environmental sustainability and deliver high quality health services for our community and future generations. This i-Hub project supports the delivery of the Sustainability Strategy through the strategic elements of:

- **Green facilities**: Build and maintain all facilities, plant, and infrastructure to enhance environmental sustainability and resilience
- **Green monitoring**: Measure, monitor and report on key sustainability metrics to track progress and identify opportunities for improvement
- **Green partnerships**: Collaborate with other organisations to improve sustainability performance and innovation within the healthcare sector

The i-Hub project will also provide valuable analysis in the development of our Energy Implementation Plan. The Energy Implementation Plan will explore initiatives designed to reduce emissions, improve resilience and transition to renewable energy. Strategies to investigate include on-site solar panels, energy contract reviews, decarbonising relevant equipment and infrastructure, and additional opportunities to reduce energy wastage and enhance energy efficiency.

Metro North Health is proud to be the trial location of the i-Hub project, and we are particularly excited at the potential expansion of this technology across other health services to improve emission reductions nationally.

[28] https://metronorth.health.qld.gov.au/wp-content/uploads/2021/04/green-mn-sustainability-strategy-21-26.pdf