IoTwins
BIG DATA OPTIMIZE INDUSTRY AND SERVICES

Grant Agreement N°857191

Distributed Digital Twins for industrial SMEs: a big-data platform

DELIVERABLE 5.2 – FIRST DIGITAL TWIN VERSION DELIVERY FOR THE FACILITY MANAGEMENT TEST-BEDS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement № 857191

Document Identification

Project	IoTwins
Project Full Title	Distributed Digital Twins for industrial SMEs: a big-data platform
Project Number	857191
Starting Date	September 1st, 2019
Duration	3 years
H2020 Programme	H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)
Topic	ICT-11-2018-2019 - HPC and Big Data enabled Large-scale Test-beds and Applications
Call for proposal	H2020-ICT-2018-3
Type of Action	IA-Innovation Action
Website	iotwins.eu
Work Package	WP5
WP Leader	FCB
Responsible Partner(s)	FCB
File Name	DELIVERABLE 5.2 – FIRST DIGITAL TWIN VERSION DELIVERY FOR THE FACILITY MANAGEMENT TEST-BEDS
Contractual delivery date	April 30th, 2021
Actual delivery date	May 10th 2021
Version	1
Status	Final
Dissemination level	Public
Author	Alex Gil Julian (FCB, alex.gil@fcbarcelona.cat), Imanol Eguskiza (FCB, imanol.eguskiza@fcbarcelona.cat), Florian Kintzler (SAGOE, florian.kintzer@siemens.com), Tobias Schüle (SAG, tobias.schuele@siemens.com), Fernando Cucchietti (BSC, fernando.cucchietti@bsc.es), Eduardo Graells (BSC, eduardo.graells@bsc.es), Feliu Serra (BSC, feliu.serra@bsc.es), Carlos Calatrava (BSC, carlos.calatrava@bsc.es),

	Claudio Domeino Arlandini (CINECA, c.arlandini@cineca.it), Eric Pascolo (CINECA, eric.pascolo@cineca.it), Roberto da Vià (CINECA, r.davia@cineca.it), Paolo Bellavista (UNIBO, paolo.bellavista@unibo.it), Guiseppe di Modica (UNIBO, giuseppe.dimodica@unibo.it).
Contact details of the coordinator	Francesco Millo, francesco.millo@bonfiglioli.com

Executive summary

Here we report on the first version of the facility management digital twins of the IoTwins project. We describe their inner workings, their components, interactions, models used, intermediate processes, and preliminary results of this first version.

The three testbeds included in this work package share similarities in their overall goals, so even though the system and the goals are quite different for each one, the methods and technology share some commonalities, such as the architecture, Machine Learning algorithms, or data systems.

While adapted to their particularities, the three digital twins follow a general pattern of a layer of IoT or Edge devices that collect data from the facility, a Cloud layer with Machine Learning algorithms that process the sensor data, an HPC or compute layer that performs simulations and optimizations, which go back to the Edge layer for actuation.

These first versions of the digital twin will now start being used and evaluated towards a second version where functionalities will be expanded, issues will be addressed, and further packetization will be achieved.

Table of Contents

1 Introduction

This report documents the first version of the facility management digital twins developed in work package 5 of the IoTwins project.

The digital twins follow a general pattern of a layer of IoT or Edge devices that collect data from the facility, a Cloud layer with Machine Learning algorithms that process the sensor data, an HPC or compute layer that performs simulations and optimizations, which go back to the Edge layer for actuation.

The systems reported here are a product of the tasks T5.1.2, T5.1.3, T5.2.2, T5.2.3, T5.3.2, T5.3.3, and T5.3.4. On the first stage of the project, the three testbeds focused on gathering, cleaning, and organizing the input of data from the dge/IoT devices (see deliverable 5.1). In this second stage, work was done on two fronts: one, exploring and selecting proper Machine Learning models to analyse, process, and extract relevant information from the input data to feed the simulations at the core of the digital twin, which were the second line of work. As far as simulators go, each testbed has widely different needs, from parallel HPC code to statistical predictors, but they all contribute to the design by evaluating scenarios that seek to optimize the target cost functions.

The rest of this document is devoted to providing, for each test bed, their general architecture, a description of their components, and the first results obtained with the system.

2 Testbed N5

This section provides an overview of the current state of facility management Test bed #5: The Camp Nou stadium. The testbed consists mainly of a digital twin of the crowd behaviour during use-case scenarios, e.g., an evacuation of the venue, fed by data internal and external to the stadium, and processed by suitable Machine Learning algorithms that provide input to the simulator. Here, we describe the implementation of this testbed as well as the three main business outcomes from the developed technology: a demand prediction model, the ability to perform arrival management optimization, and the generation of evacuation reports for several future scenarios at the stadium. We also report here the working methodology and conceptual framework behind the implementation of the pilot, as well as the current outcomes of the project.

To achieve these outcomes there is a common workflow in the implementation, which starts from Fútbol Club Barcelona (FCB) data (both, IoT and non-IoT) and a pre-processing stage at the Edge. The resulting data is made available to the Cloud through the FCB API. The Cloud is then responsible for performing data analytics, Machine Learning, and simulation, and to process the results of these stages into actionable outcomes.

The outcomes from the three business cases emerge from the architecture in the following way. FCB has several data sources coming from their daily operations, such as ticket selling, customer entrance, anonymized WiFi usage logs, among others. This data coming from IoT and non-IoT sources is pre-processed in Edge Computing at the FCB premises. The processed data is made available to the Barcelona Supercomputing Center (BSC) Cloud through an Application Programming Interface (API). The Cloud analysis has three main steps, first, a Machine Learning stage where FCB and third-party data where the output is a set of spectators (visitor) profiles, which are used directly in the first outcome of the project (demand prediction model), as well as input for the next stage, the Digital Twin simulation. In the simulation, the digital twins have realistic traits according to the ML models and visit the venue in several configurations and scenarios. For instance, a typical soccer match at the venue may have 60K visitors, then, we run many simulations with that volume to identify patterns in how people behave while arriving at the stadium, and while performing an evacuation; we also use the models to predict demand for these simulations, which may be used as input for each scenario, but also for actual planning of FCB resources, as having a forecast of the number of visitors at soccer matches informs club's decisions. We assess the outcomes of these simulations, for instance, by identifying common jamming points or the time needed to reach destination from different points of origin at the venue. These results are reported in the last stage through specific KPIs and metrics defined in the project.

2.1 General description of the system and current status

We have finished the architectural design that integrates the several components of the project, following the main IoTwins schema. Figure 2.1 shows a detailed view of framework components (IoT, FCB Resources, FCB Edge, BSC Cloud, Outcomes/Use Cases) with their inner implementation components (such as API, Simulator, etc.), as well as the inputs and outputs of each node. The current status of the system is partial completion with the full pipeline having an already instanced implementation, which allows to collect data from FCB, train models, perform simulations, and visualize the results as business insights for FCB.

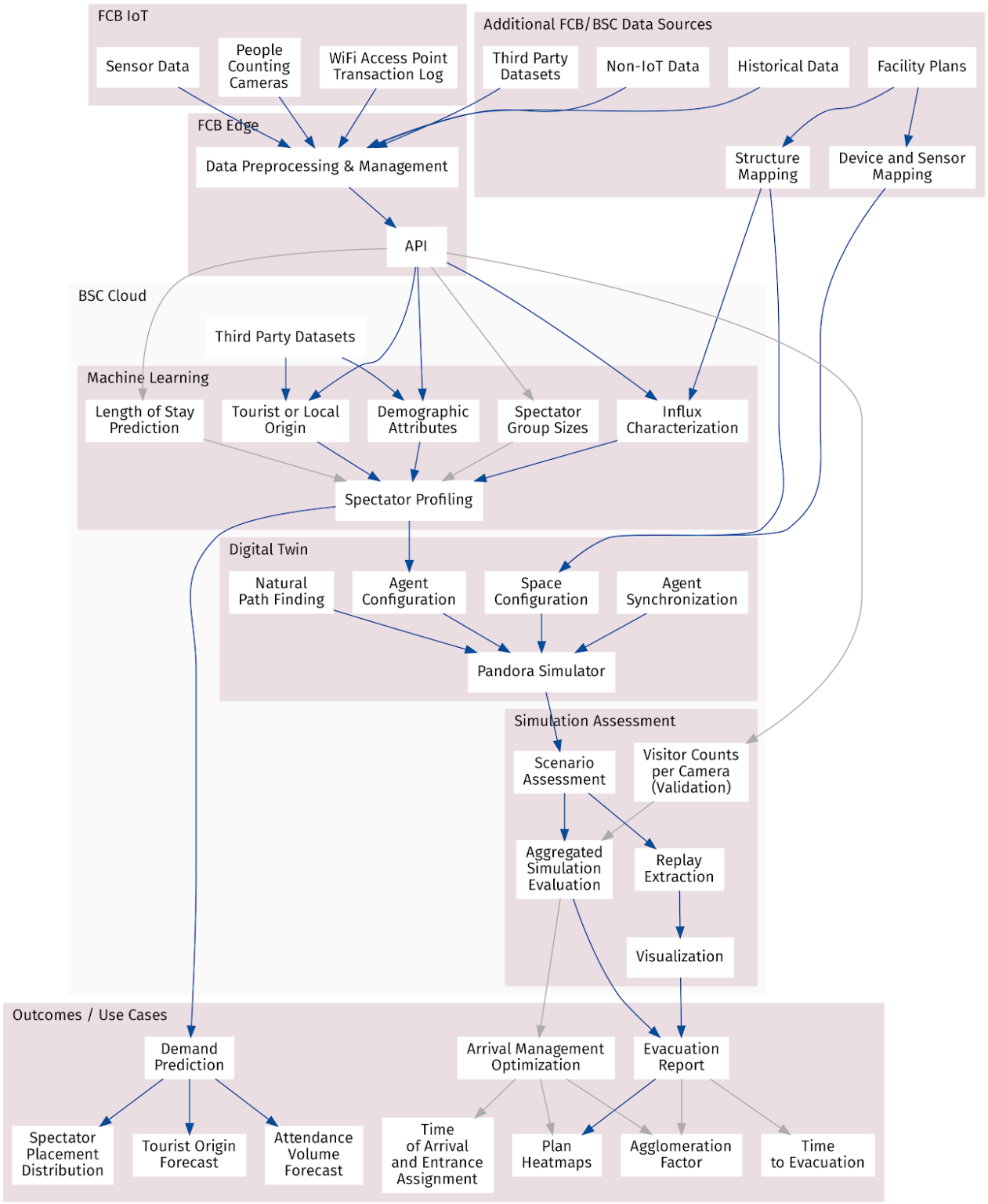


Figure 2.1 Implementation flow. Arrows represent flow of information. Blue arrows are implemented, grey arrows are planned or work-in-progress

Table 2.1 describes the status of the implementation of each of the components.

Table 2.1 Summary the testbed status

Component	Implementation Nodes	Status	Description
FCB IoT	*Sensor Data* *People Counting Cameras* *WiFi Access Point* *Transaction Logs*	Ongoing	IoT data from FCB. This includes static and dynamic data. Current status is ongoing, as cameras for people counting are being evaluated before installation.
Additional FCB/BSC Data Sources	*FCB: Facility Plans* *Third Party Datasets* *Historical Data* *Non-IoT Data* *Device and Sector Mapping* *Structure Mapping*	Completed	This group refers to relevant data for the project that does not come from IoT sources. We use two types of data sources: first and third party. Third party data has been collected by BSC, either specifically for this project or through other projects held at the institution. The status is **completed**: these datasets have already been gathered and analysed.
FCB Edge	*Data Preprocessing & Management* *FCB API*	Completed	This group refers to setting up the infrastructure needed to transfer data from FCB to BSC in a secure and regulation compliant way. The status is **completed**: the API has been accessed by BSC to collect data for the simulator.
(BSC Cloud) Machine Learning	*Third Party Datasets* *Demographic Attributes* *Tourist or Local Origin* *Influx Characterization* *Spectator Group Sizes* *Length of Stay Prediction* *Spectator Profiling*	Ongoing	The main outcome of this stage is what we denote Spectator Profiling, defined as several characteristics of the Camp Nou visitors that become parameters of each agent or group of agents in the simulation. The status is **ongoing**: there is a first working version of Spectator Profiling, however, some aspects need completion.
(BSC Cloud) Simulation	*Agent Synchronization* *Natural Path Finding* *Agent Configuration* *Space Configuration* *Pandora Simulator*	Ongoing	We implement the simulation using Pandora, developed internally at BSC and available as open source. We have introduced several improvements in the implementation (although they are not public yet) that relate to natural path finding and agent synchronization, as well as agent configuration using the outcomes from the analysis on the previous groups directly on the simulator configuration. The status is **ongoing**: there is a first working version of the Simulator. Currently we are working on improving its performance to be able to perform simulations at scale.
(BSC Cloud) Simulation Assessment	*Scenario Assessment* *Visitor Counts per Camera (Validation)* *Replay Extraction* *Aggregated Simulation Evaluation* *Visualization*	Ongoing	In this group of tasks, we measure the outcomes of the simulations for each scenario. We quantify the quality of the simulation using external data, particularly, Visitor Counts per Camera, considering the design of the project where Edge devices will report crowd behavior to the system in real time. The status is **ongoing**: our current implementation provides replay extraction for visualization. We are working on the other aspects of this component.
Outcomes / Use Cases	*Evacuation Report* *Arrival Management Optimization* *Demand Prediction* *Time to Evacuation* *Agglomeration Factor* *Plan Heatmaps* *Time of Arrival and Entrance Assignment* *Spectator Placement Distribution* *Tourist Origin Forecast* *Attendance Volume Forecast*	Ongoing	This group of tasks aggregates, analyses, and contextualizes the results from the previous groups into concrete outcomes aligned with value creation for the partners involved. The status is **ongoing**: currently, we have concrete outcomes for the Demand Prediction use case.

2.2 The Digital Twin

In a crowd, no one behaves exactly like another person. Each of us is unique. In a virtual world, a digital twin represents a physical person. A digital twin does not necessarily have an identity, but it has realistic traits. In this pilot, these traits are inferred from digital traces and other data sources. In this section we describe the high-level concepts that describe our approach to solve this problem.

We implement digital twins through Agent-Based Modelling (ABM)[1]. Agents are virtual entities that behave according to rules. These virtual entities include people, as well as other elements from the scenario at hand that are subject of analysis. ABM puts agents in an artificial space to make decisions and interact in a simulation, according to world resources, interaction rules, learning patterns, and agent memory. The interaction rules are applied per agent, however, their collective decisions shape insights regarding the world. Thus, if correctly designed, these simulations allow us to capture emergent collective patterns. In the context of the pilot, the rules include methods to find a path toward each agent destination and to follow (or not) others. Then, by aggregating thousands of simulations and the behaviour of their agents, we will understand global and local evacuation patterns at the venue.

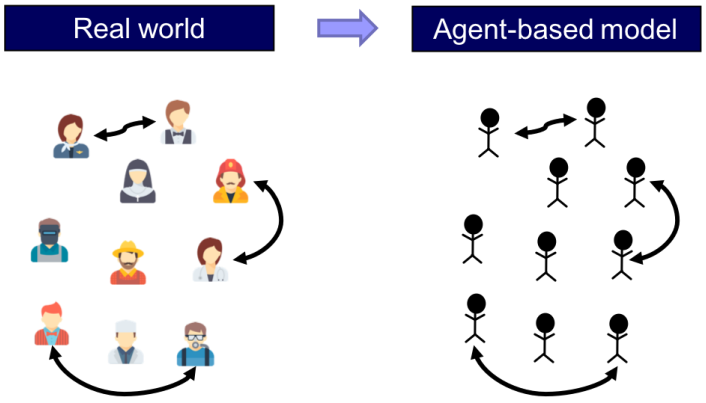


Figure 2.2 A schematic view of digital twins through Agent-Based Modeling.
Source: https://wisc.pb.unizin.org/agent-based-evolutionary-game-dynamics/chapter/0-2/

However, writing behavioural rules in a complex scenario is not easy. In addition to the influence of the environment on the difficult decisions behind an evacuation, there are individual and cultural differences[2]. A stadium presents a variety of visitors, unlike typically simulated facilities which have some control on who accesses the facility. For instance, women do not follow the same factors as men, nor do the elderly with respect to people in their 30s. Locals are more likely to be in smaller groups than tourists for some countries of origin. Furthermore, there are differences in the role of the stadium as a place of visit, as it is more than just sport: for some it is also a place of worship, a place of heritage, a place to eat, and a place of excursion[3].

To be able to identify, learn, and model these characteristics of agents, we rely on mobility analysis. Being primarily an urban discipline, where a city is composed by several layers that interact within them[4], a mobility analysis of these layers and their relationship with a venue enables us to define a framework for our

[1] Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the national academy of sciences, 99(suppl 3), 7280-7287.
[2] Hofstede, G., Hofstede, G. J., & Minkov, M. (2010). Cultures and organizations: Software of the mind, third edition (3rd ed.). McGraw-Hill Professional.
[3] Edensor, T., Millington, S., Steadman, C., & Taecharungroj, V. (2021). Towards a comprehensive understanding of football stadium tourism. Journal of Sport & Tourism, 1-19.
[4] https://cityprotocol.cat/

implementation (see Figure 2.3). The relationships between the different city layers (such as the *Society*, *Communication Network*, *Environment*, and *Built Domain*) provide transitions and exchange of input/output information of the several stages of the project, from a conceptual and a technical perspective. The objective of facilitating Camp Nou's management and planning (a task within the *Built Domain* layer) can be formalized in terms of performing three tasks:

- Evaluate **arrivals** at the stadium.
- Evaluate **evacuation** procedures.
- Predict **demand** of the premises.

These tasks can be performed through visualization & ML. The input is the result of scenario evaluation (*Built Domain*), performed through simulation using ABM. The scenario evaluation requires behavioural rules (from the *Society* layer) and a space configuration (*Built Domain*). At this stage, these rules and its association to the space configuration are determined through data-driven mobility models. The models and their outcomes are learned from behavioural data (resulting from the *Communication Network* layer), city data (from the *Environment* layer) and sensor data from the venue (*Built Domain*). In technical terms, these input sources are known as digital traces and open data.

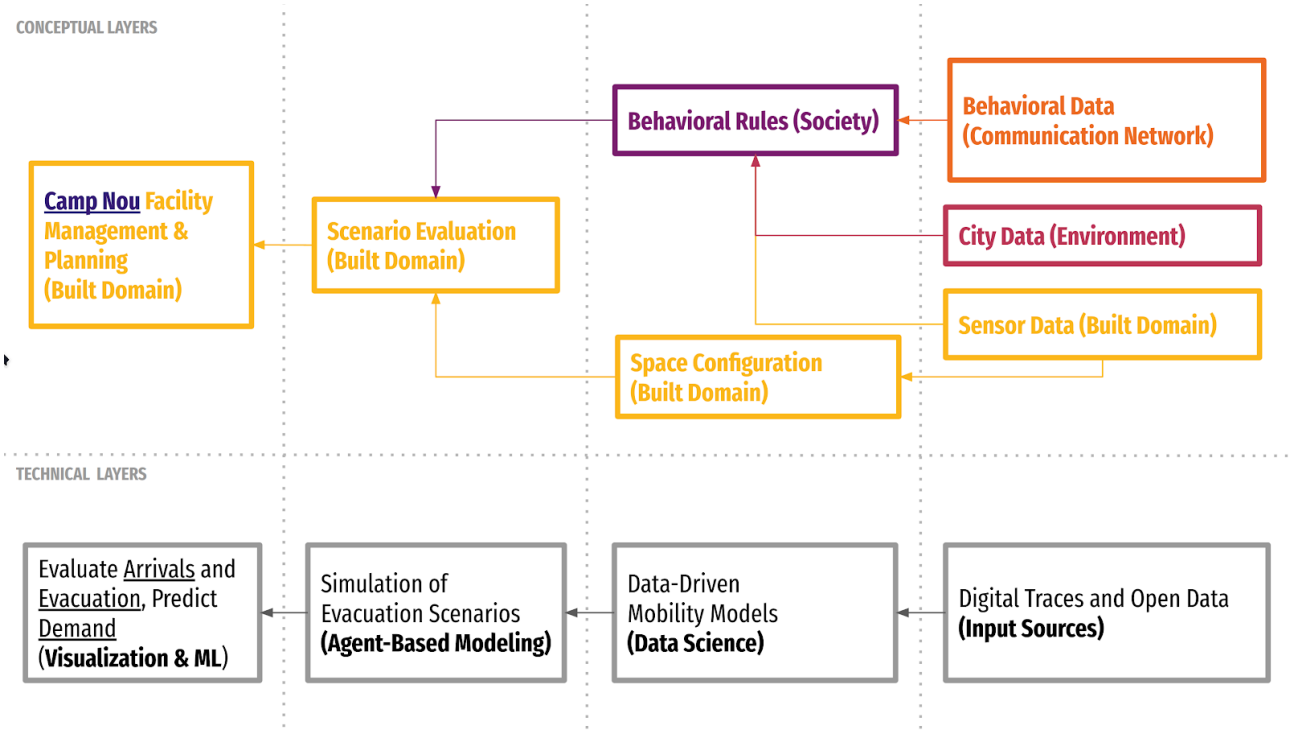


Figure 2.3 Conceptual layers of abstraction in this pilot.

This flow of concepts and layers explain the overall work in this pilot. Next, we describe the implementation details and the current status.

In the next section we explain the implementation of the pilot and its components. Figure 2.3 shows the global schema of the implementation, where each cluster of nodes contains *data sources*, *concepts*, and *technical resources*. These nodes are connected through arrows in the direction of flow of information. Each arrow is coloured in blue (implemented) or grey (planned or work-in-progress). The flow starts from the *IoTwins Architecture* and the *Static Data Sources* available. Both feed input data into the *Space Configuration* and *Data Analysis and Prediction (ML)* stages. This last stage already produces some *Outcomes*, as well as input

data for the *Simulation* stage, which, in turn, activates the *Assessment & Reporting* Stage. This last stage generates the remaining *Outcomes* of the project.

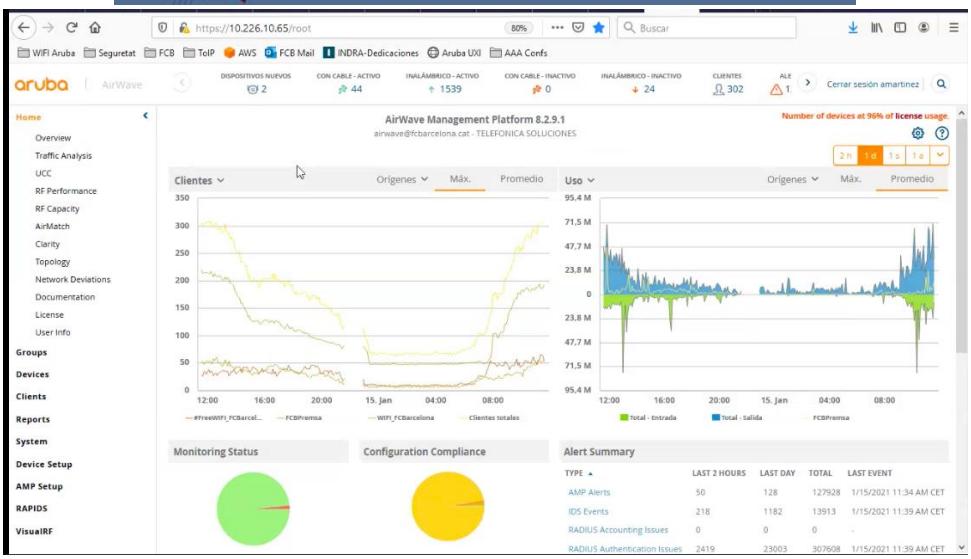
2.3 Components

2.3.1 Data Sources

We use two types of data sources: first and third party. First party data comes from FCB in two forms: static and through its API (which follows the IoTwins Architecture). Third party data has been collected by BSC, either specifically for this project or through other projects held at the institution. We list the data sources used below:

First party sources

- Static data:
  - Access records from Wi-Fi Access Points located in Camp Nou (1464 APs total, see Figure 2.4) in a three-month period. These records contain anonymized trajectories of mobile phones with Wi-Fi connectivity on through the venue that need to be reconstructed from records that contain a timestamp at which an AP identified an individual (anonymized) user device.



User	Timestamp	Location (closest AP)
508385291287920	2019-12-01 13:30	40:e3:d6:1b:17:30
488633074697187	2019-12-01 01:37	40:e3:d6:1e:28:80
470668660234431	2019-12-01 11:57	40:e3:d6:1e:54:40

Figure 2.4 (Top) Sample location of Access Points in the stadium. (Middle) Interface of the Aruba system used to control and monitor the APs. (Bottom) Sample data collected from WiFi Access Points

- Facility plans are provided for each floor of the stadium in Autocad's DWG format[5]. Each DWG file contains the following layers of objects: permanent obstacles to the circulation of people (such as walls and pillars), removable obstacles (such as stadium seating), architectural elements for the vertical circulation from one floor to another (such as stairs and slopes), vertical circulation elements within the same floor (such as few-step stairs and small slopes), doors, their direction of opening and a text layer with descriptive labels (see Figure 2.8).

- FCB API:
  - Matches Master. This dataset contains a curated list of all the games played at the FCB premises from the 2010-2011 season, including Camp Nou, Johan Cruyff Stadium and Palau Blaugrana.
  - Camp Nou accesses. This dataset is composed of all the registered entries to the stadium during match days, including all visitors, staff, press, among others. The granularity of the data allows for precise estimations and modelling of the arrivals at Camp Nou during day matches.
  - Tickets sold. This dataset contains information about all the tickets sold for the Camp Nou (including those sold online and in person at the stadium). It includes many variables such as the time of the purchase, the zone of the stadium, inlet, etc. It can be linked with the access dataset.
  - Seats freed. The way Camp Nou seats work is that the seats are associated with FCB annual memberships, and thus, these are reserved for them. In case a member decides to "free" the seat, this ticket can be purchased by a new visitor for a particular game. It contains the time of "freeing" the ticket and can be linked to the time of the purchase.
  - FCB Annual Memberships. This dataset contains information about the annual memberships, and thus allows for knowing the exact place and seat allocated for a person during a given season. Knowing this, together with the number of seats freed we can know how many tickets are available at each given time period and estimate the supply of available seats for any particular game.

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

```
_start = datetime.now()
limit = 1000
dict_errors = dict()
for year in ['11-12', '12-13', '13-14','14-15', '15-16','16-17', '17-18', '18-19'][::-1]:
    _hasMore = True
    count_req = 0
    list_dfs = []
    offset = 0
    dict_errors[year] = list()
    count_fail_requests = 0
    _limit = limit
    while _hasMore == True:

        _req =                                                                    :?q=Aforament=ESTADI;Tempora
da={}&offset={}&limit={}'.format(year, offset, _limit)
        print('Request to make: ')
        print(_req)
        print('-'*50)
        try:
            r = requests.get(_req,
                    auth=(dict_auth['auth'], dict_auth['pwd']),
                    headers={'token':dict_auth['token']})
            assert r.status_code == 200
            _df = DataFrame(dict(r.json())['items'])
            _df.to_csv('../../data/raw/seasons_sales/data_season_{}_sales_offset_{}.csv'.format(year, offset))
            list_dfs.append(_df)
            _hasMore = dict(r.json())['hasMore']
            count_req += 1
            offset += _limit

            print('Successful requests: ', count_req)
            print('=' * 50)
            sleep(0.1)
            count_fail_requests = 0
            _limit = limit

        except:
            print('Status code ', r.status_code)
            print(r.text)
            dict_errors[year].append(r.status_code)
            count_fail_requests += 1
            sleep(1.01**count_fail_requests)
            if count_fail_requests > 25:
                _hasMore = False
                print('Too many failed requests. Moving to other year.')
                print('+' * 60)

            print('Count of failed requests: ', count_fail_requests)
            print('Time since start: ', datetime.now()-_start)
            _limit = round(_limit / 2)

    concat(list_dfs).to_csv('../../data/raw/seasons_sales/data_season_{}_sales.csv'.format(year))
```

Figure 2.5 Sample code used to connect to and gather data from the API tickets sold endpoint

Third party sources

- Foursquare data[6]. It is composed of millions of records of worldwide check-ins between 2012 and 2014. This includes check-ins in Barcelona and the Camp Nou[7].

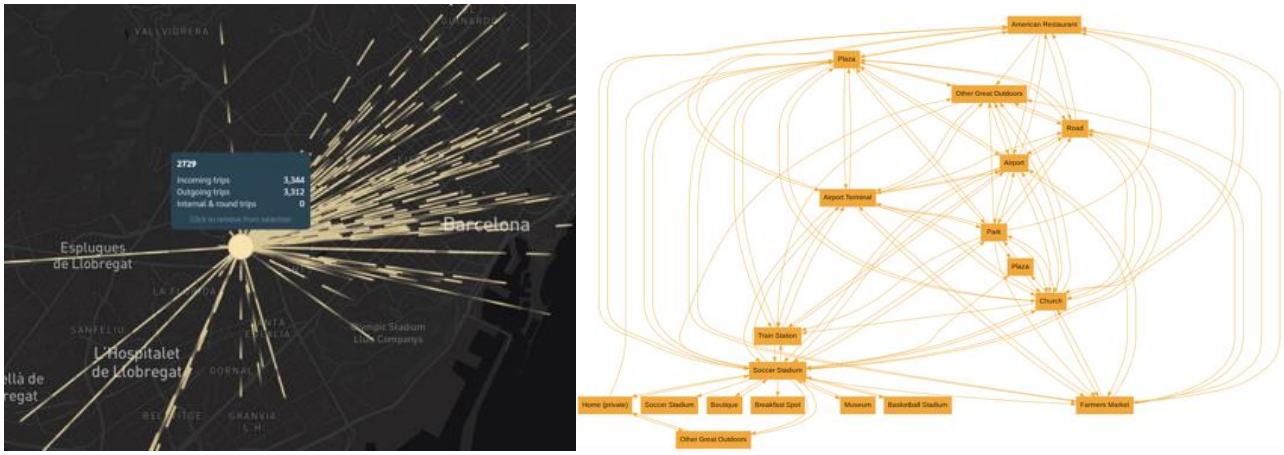


Figure 2.6 Charts showing insights extracted from FourSquare check-ins. Left: Flow to Camp Nou from local sites. Right: Connected check-ins from the same user lets us build trajectories of sequential visits before and after the check-in at Camp Nou

[6] Available at https://sites.google.com/site/yangdingqi/home/foursquare-dataset

[7] An exploratory analysis is available in this repository: https://github.com/zorzalerrante/iotwins_foursquare

- Mobile Phone Data from an agreement between the Barcelona City Council and Vodafone[8]. This dataset contains visitor statistics disaggregated per type, age cohort, and gender for several areas of the city (not at venue level).

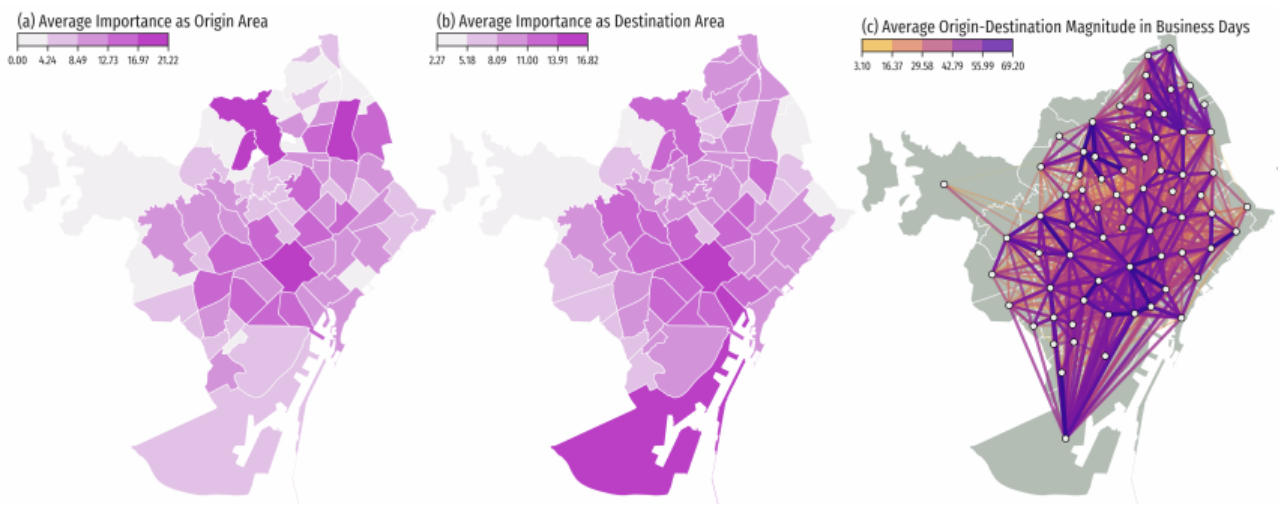


Figure 2.7 Origin-destination flows between neighbourhoods in Barcelona based on telephony data (Vodafone). a) Map of average importance as origin area (neighbourhoods). b) Map of average importance as destination area (neighbourhoods) for local people in Barcelona.

These datasets are stored in secure machines hosted at BSC.

2.3.2 Space Configuration

We need to create an artificial, virtual space that represents Camp Nou (and its future states) based on real plans. This is an expensive manual operation. The abstraction needs to be as simple as possible to be able to program complex behavioural rules.

Firstly, the plans in AutoCAD DWG format (see Figure 2.8, obtained by partner FCB from the original Revit format) are reviewed for consistency: obstacles to horizontal displacement are checked (e.g. beams may result depicted as obstacles like walls, given the export process to DWG, when they are indeed above the walkable floor) and vertical displacement elements are verified to match among floors (given the complexity of the stadium architecture). Secondly, the drawing is simplified, and all the elements are manually rearranged in layers and assigned colour codes, according to the requirements of the Pandora simulator. Finally, the plans are exported to PNG (Figure 2.9) and pixelated at a specific scale (1 pixel = 20 cm) so that every pixel contains a different piece of information about the environment for the agent in the simulation.

[8] Graells-Garrido, E., Meta, I., Serra-Burriel, F., Reyes, P., & Cucchietti, F. M. (2020, April). Measuring Spatial Subdivisions in Urban Mobility with Mobile Phone Data. In Companion Proceedings of the Web Conference 2020 (pp. 485-494).

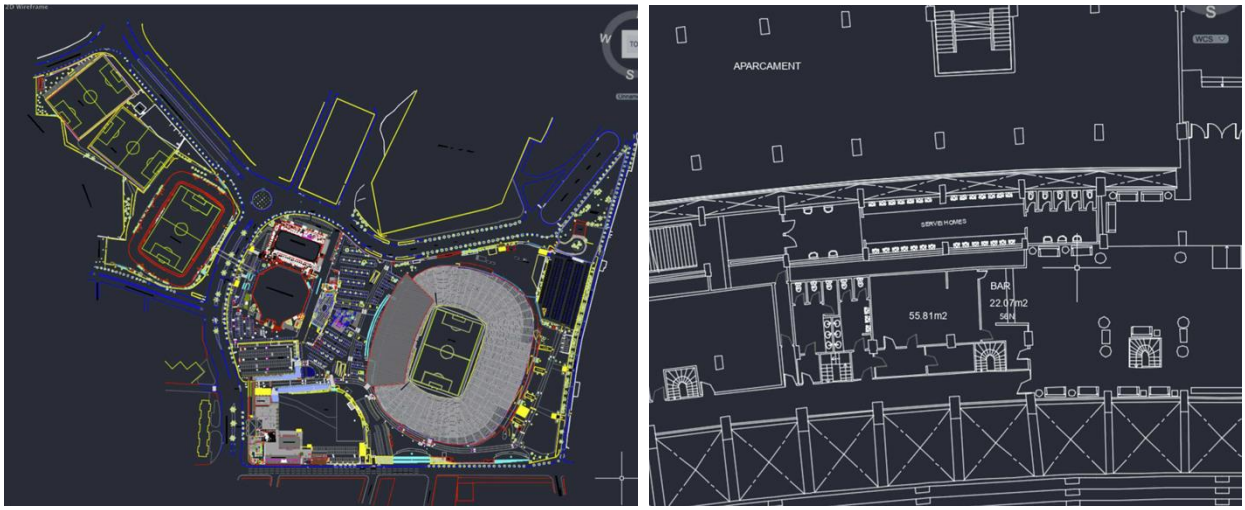


Figure 2.8 (Left) Autocad plan of the whole Espai Barça space. (Right) Detail

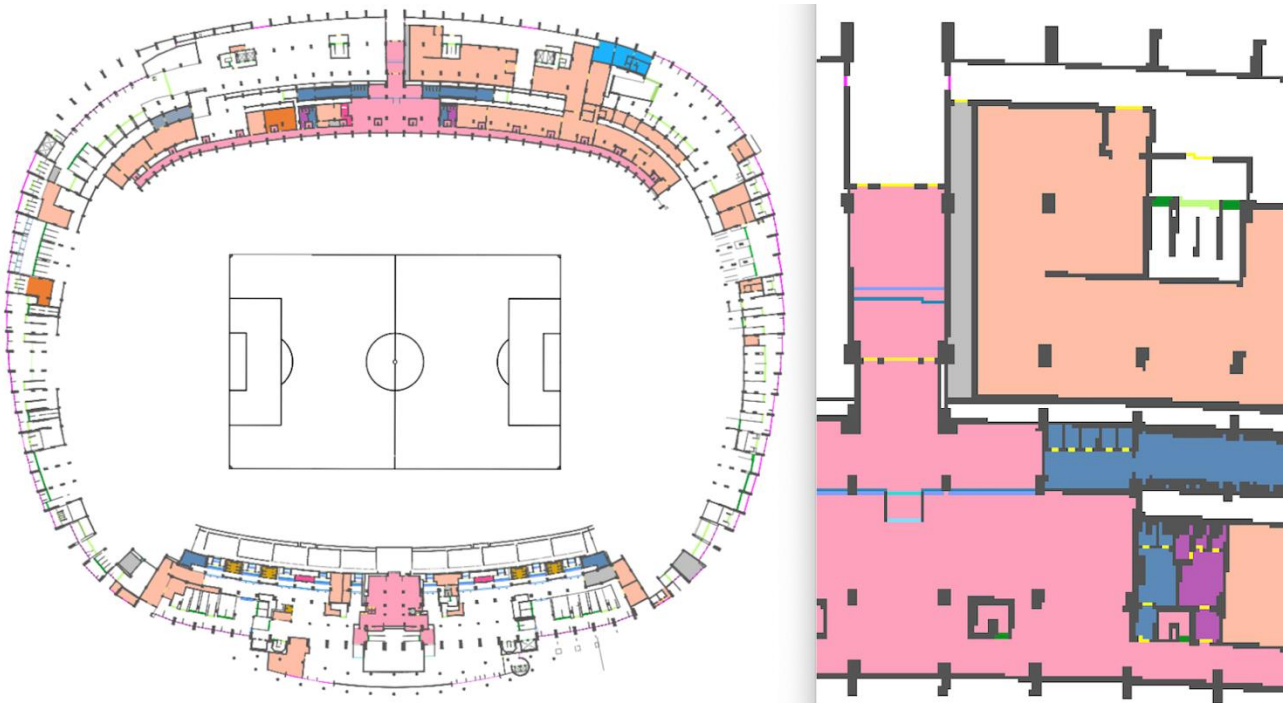


Figure 2.9 Processed plans in PNG format ready for the Pandora simulation. The colours represent different types of spaces/access for the agents (e.g. doors, stairs, restricted spaces, walls, bathrooms, vending space, etc.)

2.3.3 Spectator Profiling from Data Analysis and Prediction

The data sources collected by the IoT/Edge devices and the external sources feed the Machine Learning models whose main task is what we denote *Spectator Profiling*, defined as the characteristics of the Camp Nou visitors that become parameters of each agent or group of agents in the simulation. These include: the ***influx characterization*** (distribution of number of visitors and their arrival times at different types of events), the ***spectator group sizes*** (distribution of number of people in groups that visit the venue), the ***length of stay prediction*** (the total time people will spend in the venue), the ***demographic attributes of visitors*** (particularly, gender and age group), and the ***origin of visitors*** (either local or tourist), each described in the next subsections.

2.3.3.1 *Influx Characterization*

Our simulator requires as input the rate at which people enter the simulation along with their characteristics. To create this, we use historical data to estimate expected amounts of people to arrive at each gate or section of the stadium, which can also be used to improve the game experience and the facility management. We segment the historical data by features such as match characteristics, time of the day, and attendance profiles (see section below), since, for example, tourists might arrive earlier at the game and stroll around the stadium while taking pictures, while frequent visitors might arrive at the last minute before the game and know exactly how to go to their seat.

From the datasets mentioned above, there were 111 games played during the seasons 2016-2017, 2017-2018, 2018-2019 and 2019-2020. The average number of visitors to the stadium was 72919 people, with a standard deviation of 18157. Three of the games studied show a suspiciously low number of visitors, but two of those occurred in June 2020 after the Covid-19 induced lockdown in March, hence the low number of visitors is reasonable. The third game was discarded to avoid biasing our estimates. Hence, 110 of the games were used for the modelling of the arrivals.

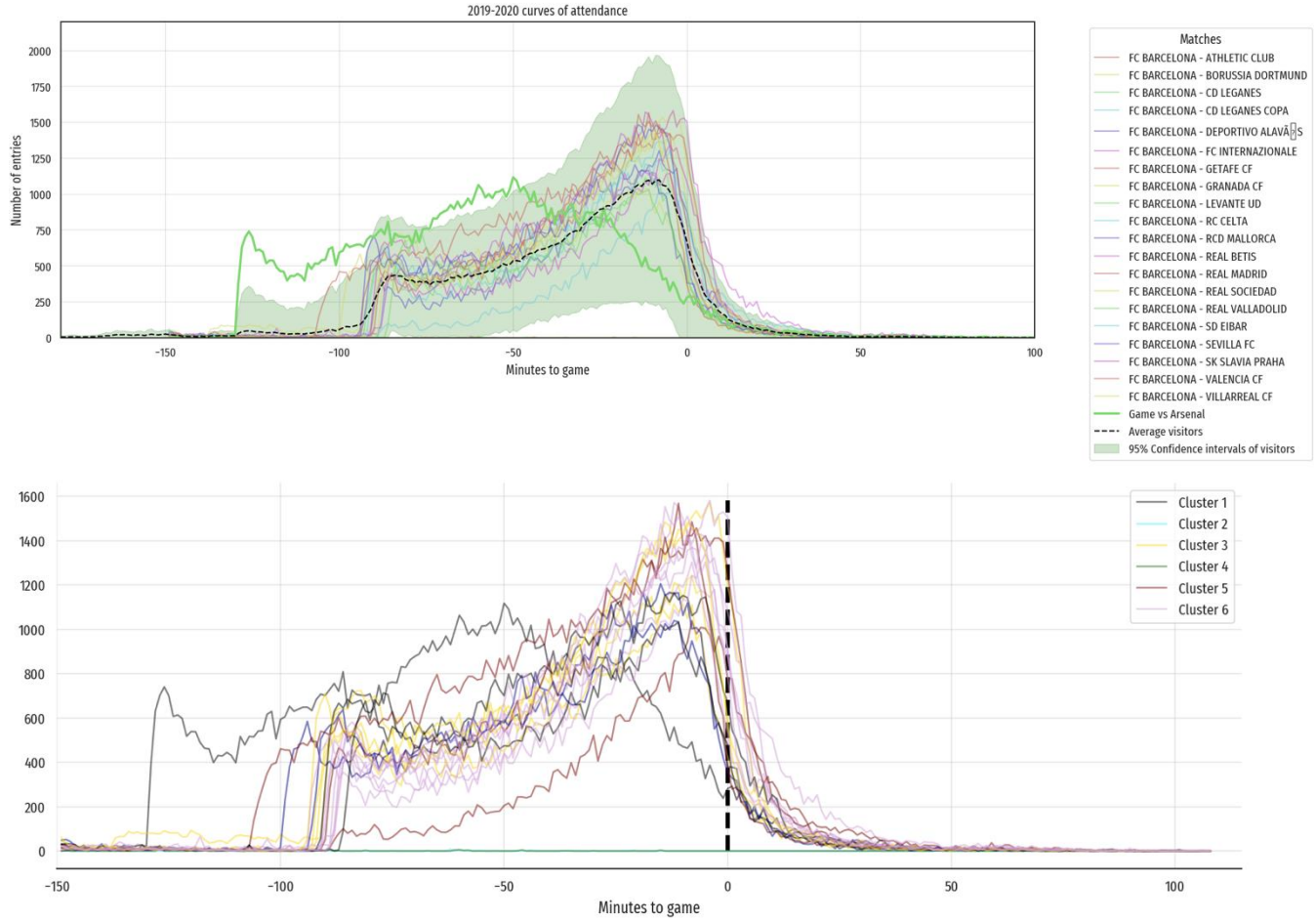


Figure 2.10 (top) Attendance pattern chart showing the overall arrivals at the venue for one entire season (2019-2020). Each line represents one game. The green area contains the standard deviation computed from all the games, and the black dashed line shows the average amount of arrivals per game with respect to the minutes before and after the game for any given match. (bottom) Although each game has a unique pattern of attendance, there are some clusters with similar behaviour.

To predict match attendances, several features were created using the *"Matches Master"* data from the FCB API. For a given match, we extract the time of the game, the day of the week, the competition of the game, the opponent, if it was a derby, if it was played during local holidays, if it was after the lockdown, the

matchday, the month of the year and the season of the year. We grouped games by applying a k-means clustering to the arrivals curves (Figure 2.10), which gave us a proxy of the popularity or branding of the opponent, whichattendance predictors[9]. These clusters can be used in the future to improve the prediction, as most of the opponents are repeated from season to season and belong to one of the clusters. The target to predict was the attendance computed from the *"Camp Nou accesses"* dataset, obtaining the number of arrivals per minute at the Camp Nou premises. Figure 2.10 shows the number of arrivals at the Camp Nou premises for all the games during the season 2016-2017.

For the prediction of the overall attendance, which is the sum of the number of visitors over a match day for any given match, a few models were implemented showing good benchmarks that will be further expanded in future iterations of this project. These forecasts are timeless, meaning that only constant variables were used. In Figure 2.11 we show the results of a simple linear regression fit including all variables and observations, where we can identify clearly which variables are relevant and which are not.

[9] Pawlowski, T., & Anders, C. (2012). Stadium attendance in German professional football–The (un) importance of uncertainty of outcome reconsidered. Applied Economics Letters, 19(16), 1553-1556.

OLS Regression Results

Dep. Variable:	P-01	R-squared:	0.730
Model:	OLS	Adj. R-squared:	0.587
Method:	Least Squares	F-statistic:	5.104
Date:	Tue, 23 Feb 2021	Prob (F-statistic):	2.45e-09
Time:	18:31:47	Log-Likelihood:	-573.47
No. Observations:	108	AIC:	1223.
Df Residuals:	70	BIC:	1325.
Df Model:	37		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	107.9779	11.642	9.275	0.000	84.758	131.198
Jornada_quartiles_2	51.3665	27.655	1.857	0.067	-3.789	106.522
Jornada_quartiles_3	57.1950	37.927	1.508	0.136	-18.447	132.837
Jornada_quartiles_4	3.3598	48.498	0.069	0.945	-93.367	100.087
Holidays_Cat	-3.2277	41.896	-0.077	0.939	-86.787	80.332
NomCompeticio_CHAMPIONS	122.0466	30.637	3.984	0.000	60.942	183.151
NomCompeticio_COPA DEL REI	21.0298	35.551	0.592	0.556	-49.875	91.935
NomCompeticio_LLIGA SANTANDER	23.5765	26.415	0.893	0.375	-29.106	76.259
NomCompeticio_SUPERCOPA DE ESPAÑA	-151.5916	67.294	-2.253	0.027	-285.804	-17.379
NomCompeticio_TROFEU JOAN GAMPER	92.9165	45.778	2.030	0.046	1.614	184.219
Month_1	21.4260	16.491	1.299	0.198	-11.464	54.316
Month_10	22.4122	18.646	1.202	0.233	-14.776	59.600
Month_11	0.7095	22.506	0.032	0.975	-44.177	45.596
Month_12	-8.7269	17.957	-0.486	0.629	-44.542	27.088
Month_2	23.7409	17.745	1.338	0.185	-11.650	59.131
Month_3	-16.4793	19.143	-0.861	0.392	-54.658	21.699
Month_4	9.5506	21.288	0.449	0.655	-32.907	52.008
Month_5	59.7272	25.106	2.379	0.020	9.655	109.800
Month_8	-18.8910	16.901	-1.118	0.268	-52.600	14.818
Month_9	14.5087	21.061	0.689	0.493	-27.497	56.514
Dayofweek_0	21.2346	66.221	0.321	0.749	-110.840	153.309
Dayofweek_1	-22.1733	25.662	-0.864	0.391	-73.355	29.008
Dayofweek_2	-4.3591	20.393	-0.214	0.831	-45.031	36.313
Dayofweek_3	10.8662	38.444	0.283	0.778	-65.808	87.541
Dayofweek_5	62.9847	24.353	2.586	0.012	14.414	111.556
Dayofweek_6	39.4247	23.540	1.675	0.098	-7.524	86.373
Year_2016	32.5890	19.002	1.715	0.091	-5.309	70.487
Year_2017	63.1674	13.637	4.632	0.000	35.969	90.366
Year_2018	39.6029	13.217	2.996	0.004	13.242	65.964
Year_2019	24.8726	14.370	1.731	0.088	-3.787	53.532
Year_2020	-52.2541	24.944	-2.095	0.040	-102.003	-2.505
Hour_13	-31.0126	62.802	-0.494	0.623	-156.266	94.241
Hour_16	-20.5859	27.461	-0.750	0.456	-75.356	34.184
Hour_18	-27.9858	26.317	-1.063	0.291	-80.473	24.502
Hour_19	-6.4973	39.447	-0.165	0.870	-85.172	72.177
Hour_20	-37.7376	22.852	-1.651	0.103	-83.315	7.839
Hour_21	-50.5703	24.376	-2.075	0.042	-99.186	-1.954
Hour_22	73.1337	31.157	2.347	0.022	10.992	135.275
Hour_23	209.2337	95.494	2.191	0.032	18.777	399.690
Season_1	45.1668	15.197	2.972	0.004	14.858	75.476
Season_2	52.7985	21.218	2.488	0.015	10.481	95.116
Season_3	-18.8910	16.901	-1.118	0.268	-52.600	14.818
Season_4	28.9035	12.850	2.249	0.028	3.275	54.532
Derbi_0	36.3161	11.956	3.037	0.003	12.470	60.162
Derbi_1	71.6618	18.076	3.964	0.000	35.610	107.714
Cluster_1	122.0080	17.533	6.959	0.000	87.039	156.977
Cluster_2	34.6159	12.161	2.847	0.006	10.362	58.869
Cluster_3	-41.5531	15.628	-2.659	0.010	-72.721	-10.385
Cluster_4	-7.0929	16.988	-0.418	0.678	-40.975	26.789

Omnibus:	0.407	Durbin-Watson:	2.001
Prob(Omnibus):	0.816	Jarque-Bera (JB):	0.311
Skew:	0.131	Prob(JB):	0.856
Kurtosis:	2.974	Cond. No.	1.47e+16

Figure 2.11 Output of the ordinary linear regression for predicting attendance to the stadium

We used a Decision Tree algorithm for predicting the number of visitors at the Camp Nou premises using a Leave-one-out Cross validation (LOOCV) technique for parameter selection. Decision trees evaluate the data and create branching points to arrive to a forecast, with the advantage that this method can incorporate naturally non-linearities in the model or the absence of data. Figure 2.12 shows a sample decision tree. We further decided to use some other less interpretable models such as Random Forest Regressor and Gradient Boosting Regressors, which could give better performance compared to simple algorithms, at the risk of overfitting as the total number of samples is not quite large and these models have more degrees of freedom. Their performance is still being evaluated.

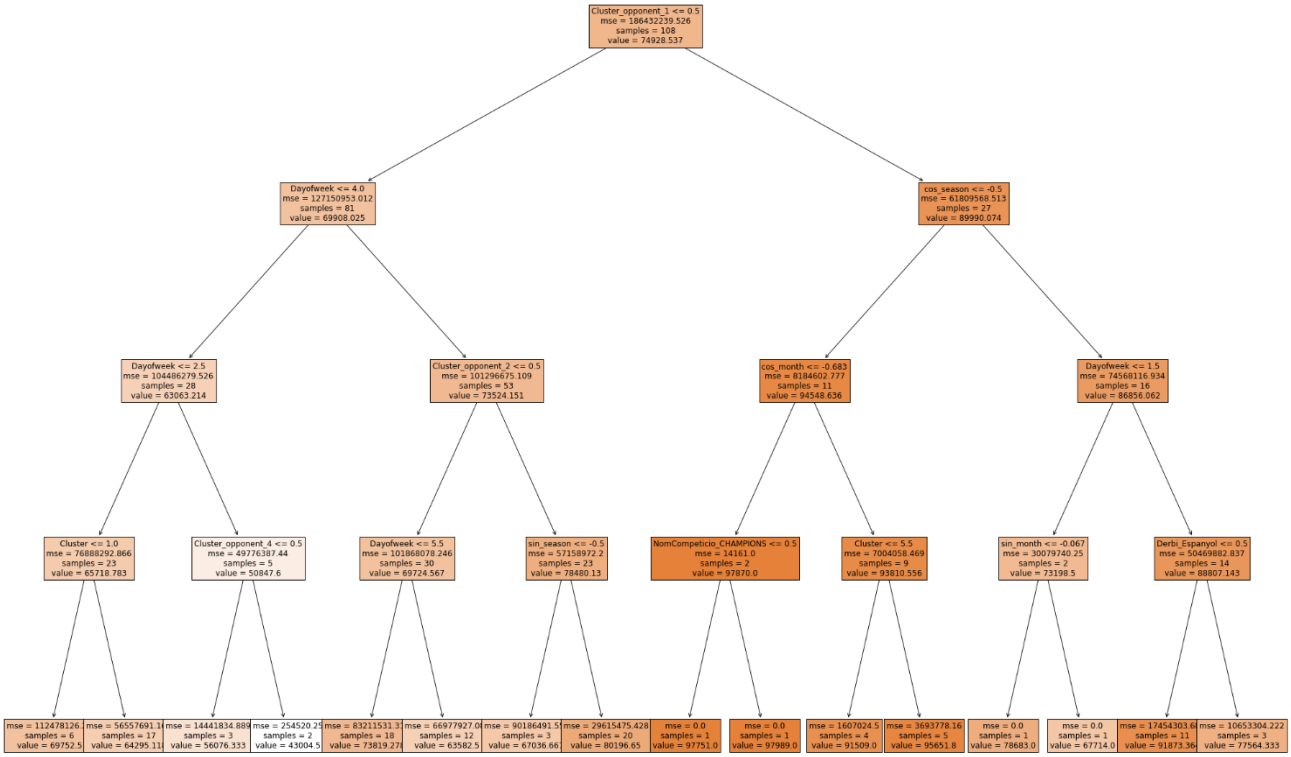


Figure 2.12 A sample decision tree for predicting the attendance at the Camp Nou Stadium

The current iteration of the model takes this prediction one step further and forecasts the attendance by minute and by entrance gate to the stadium, which allow us to increase the detail of the digital twin (since this is one of the main inputs to reproduce mobility patterns), and also to better understand the behaviour of assistants to the stadium, for example to anticipate potential overflows of people at certain specific regions and times before and after the game (see use case discussion below).

The final model sent to the stadium digital twin is the prediction, by minute and by gate, of the number of people entering the stadium, drawn from the distributions predicted by the decision trees described above.

2.3.3.2 Demographic Attributes of Visitors

To identify the distribution of visitors to Camp Nou with respect to socio-demographic characteristics, we resort to third party data available at BSC from other projects held by the institution. Particularly, we use a mobility dataset from 2018 originated in a collaboration between BSC, Barcelona City Hall, and the mobile phone operator Vodafone. This dataset contains aggregated visitor counts at several areas of the city. It is a dataset that respects privacy as it only contains the number of people in a given area, according to the following groups: men and women, several age cohorts, and tourists (national and foreign) or locals (residents from the Barcelona Metropolitan Area). Figure 2.13 shows the distribution of women, elderly, and tourists in the city. The Camp Nou area on the western part of the city is easily identifiable in the third map (tourists), however, the size of each area contains much more than the stadium, and connection patterns in mobile phones do not guarantee that each device is connected to the nearest cell station.

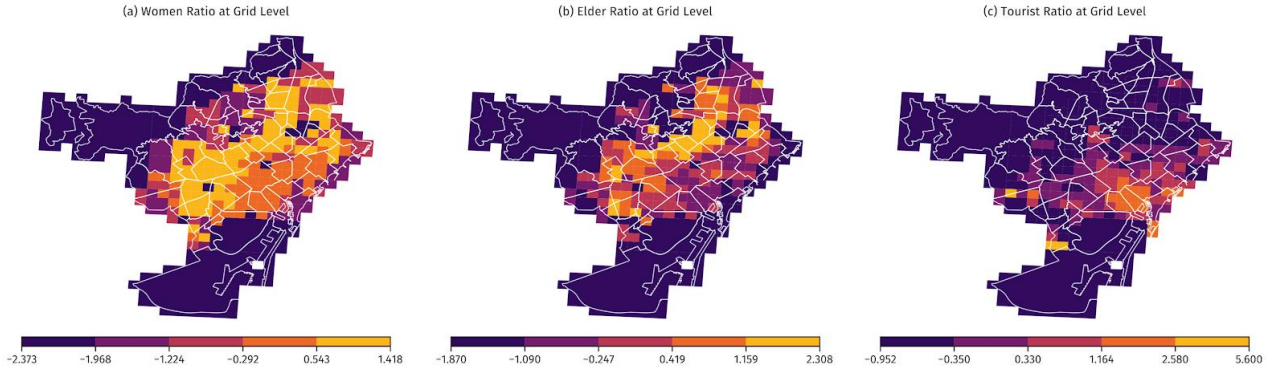


Figure 2.13 Heatmap of tendency to receive visitors of three types in the city: women, elderly, and tourists.

Thus, the dataset requires processing before being incorporated into the pipeline. In order to localize the results to the Camp Nou cell, we compare the visitor counts in match days and non-match days to remove the constant "visitor noise" from the signal and compare the target cell density for a given subgroup with the distribution across the city.

2.3.3.3 Origin of Visitors

The usage of mobile phone data already allows to measure a number of local and tourist visitors at the venue. However, we observed that the tourist category is severely biased by the availability of the mobile operator in foreign countries. For instance, according to the data there were no visitors from China, although FCB knows from its records that Chinese tourists frequently visit their installations. Hence, we needed to balance the distribution found in this third-party dataset using another third-party dataset from the (then) social network Foursquare. Foursquare allowed its users to broadcast their position to the social network in the form of a check-in, that is, a record of the form "*user A was in place B at date C*." This makes it possible to build a spatial-temporal trajectory for all users in the dataset, and to focus on users who have visited a particular venue (Camp Nou in this case), see Figure 2.14.

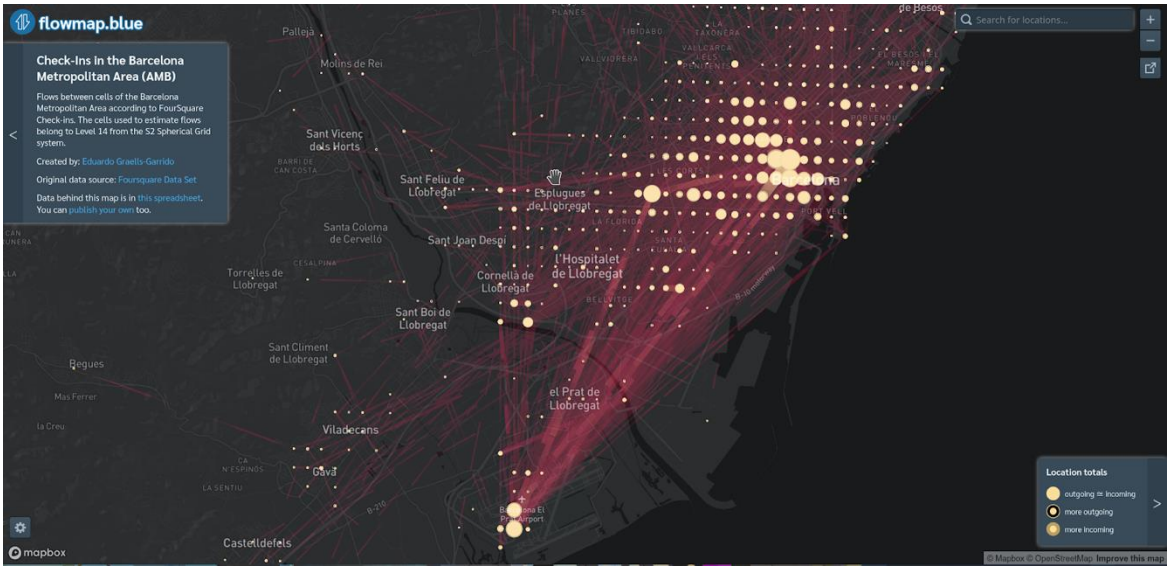


Figure 2.14 Network of visits for tourists and locals in Barcelona according to their check-ins.

Although the data is anonymized, the country of origin of users can be inferred based on their trajectories, under the assumption that many of them have more check-ins at their countries than in touristic destinations. Additionally, residents from the Metropolitan Area may have a check-in at their own homes, and such check-ins are marked as such. Hence, we build a visitor-venue matrix for all check-ins in Barcelona, and then determine a locals/tourist's distribution from this matrix and the inferred country of origin of some visitors. We do so with a Semi-supervised version of the Non-Negative Matrix Factorization (NMF)[10] algorithm, known as Topic-Supervised NMF[11].

Using this dataset allows us to estimate distributions of time-of-arrival according to visitor origin, as well as to estimate visitor counts for non-match days (useful for the next iteration of the project, testbed 11, with a focus on replicability). Although the time-of-arrival distribution from the official API data prevails, this dataset gives us a different perspective, as we can relate time of arrival to the previous and posterior activities held by each visitor. This is relevant for business use-cases in FCB, but also to a next iteration of the simulation, for instance, knowing which percentage of visitors have eaten before their visit changes the decision process within the agent-based simulator[12]. In the next iteration we will include recent data made available through the FCB API, and potentially other sources of qualitative information, such as TripAdvisor.

2.3.3.4 Group Size and Length of Stay

The Wi-Fi data contains dense user trajectories in terms of Access Points (APs), transmission devices where each mobile phone connects to the network. However, in addition to being dense the data are noisy, as a mobile device that is just passing by near the venue is also registered: even weak connections are registered. These characteristics impose challenges regarding the estimation of group sizes and length of stay. Thus, we are in an intermediate stage where we are exploring the dataset to identify patterns that allow us to separate the signal (real Camp Nou visitors) from the noise (devices not used by humans, people passing through or nearby the venue). Our first exploration shows that the visitor distribution fairly reflects the binary categorization of match-day and non-match day.

[10] Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755), 788-791.

[11] MacMillan, K., & Wilson, J. D. (2017). Topic supervised non-negative matrix factorization. arXiv preprint arXiv:1706.05084

[12] Hristova, D., Liben-Nowell, D., Noulas, A., & Mascolo, C. (2016, April). If You've Got the Money, I've Got the Time: Spatio-Temporal Footprints of Spending at Sports Events on Foursquare. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 10, No. 1).

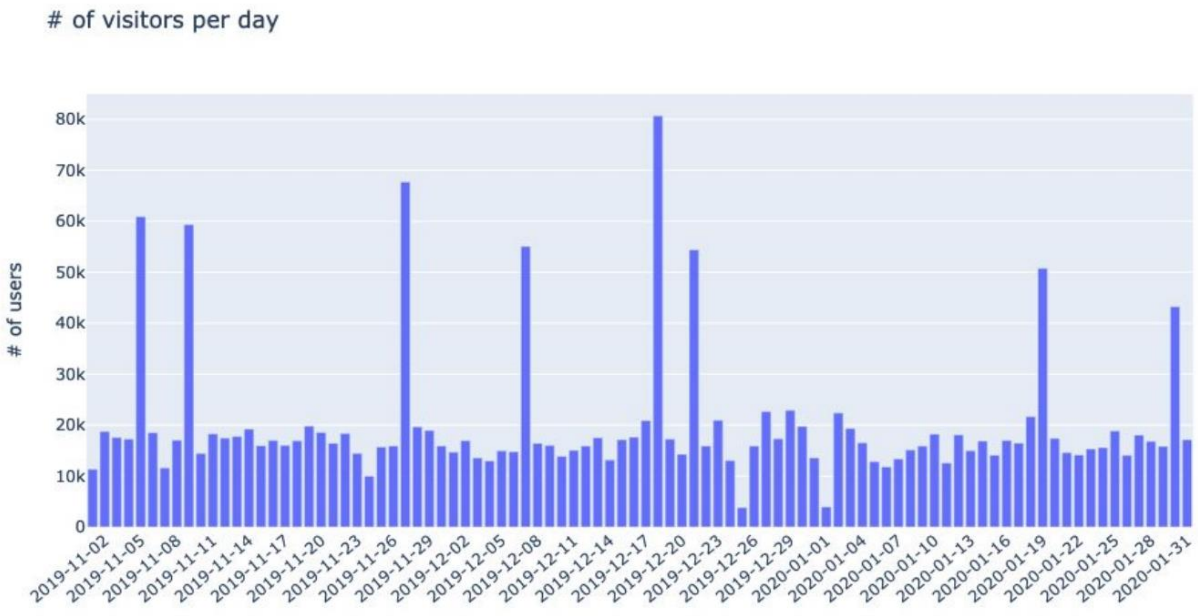


Figure 2.15 Number of visitors per day according to the WiFi data.

The need to remove noisy data becomes evident when evaluating the average time spent within the stadium. The expected length of stay on match days has a minimum of around 90 minutes of play time, plus a maximum around 120 minutes of waiting. However, the mean values tend to be near 250 minutes, which is higher than one would expect. Also, stays on non-match days are much higher. The devices of employees and venue personnel are too few (less than 1%) to influence these distributions that much, and we are currently working on a noise classifier to remove the records that are not meaningful to our use case.

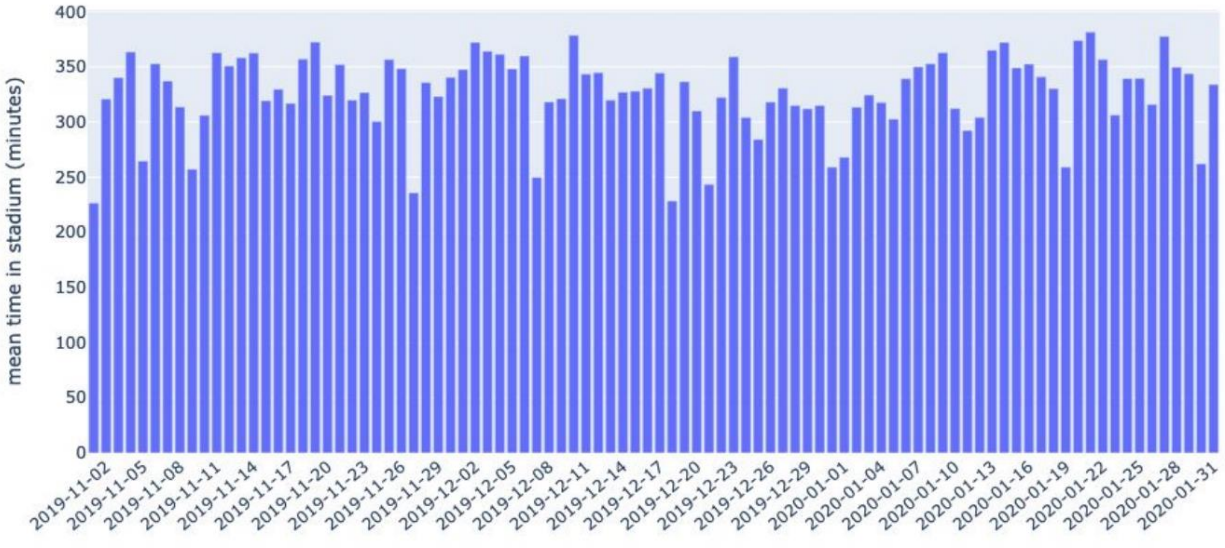


Figure 2.16 Average time spent within the stadium, per day.

For the next version of the digital twin, after cleaning the data, we will identify group sizes by clustering device trajectories according to the sequence of APs connected to.

2.4 Agent-Based Simulation

We implemented the simulation using Pandora[13], an agent-based simulation software developed internally at BSC and available as open source[14]. We have introduced several improvements in the implementation (although they are not public yet) that relate to *natural path finding* and *agent synchronization*, as well as *agent configuration* (age, gender, knowledge of terrain, propensity to make random stops or detours, etc.) using the outcomes of the ML algorithms described in the previous section. Another relevant input to the simulator are the predictors of arrivals, which provide the overall creation of new agents that enter the simulation as well as their initial location and profile.

Here we describe the specifics of the natural path finding and agent synchronization algorithms.

2.4.1 Natural Path Finding

The agents follow a pedestrian movement model[15] in two distinct scenarios, following the two main situations of the system, the usual day at the facility, being it a match or a regular day, and an emergency situation that requires its evacuation. The decision structure for agents during the simulation is always the same. First the agents update the knowledge of their environment, then, they explore their surroundings within their field of view. Each agent has an immediate target defined according to the scenario being simulated; according to this, the agent chooses one of the two main branches of the decision process. Depending on its knowledge, attributes and surroundings, the agents can perform a series of actions within each step of the simulation[16]. This decision tree, depicted in the flux diagram of Figure 2.17, has two main branches, one representing a regular day at the facility (in blue, bottom part), and one representing an evacuation scenario (in yellow, right part).

[13] Rubio-Campillo, X. (2014). Pandora: A versatile agent-based modelling platform for social simulation. *Proceedings of SIMUL*, 29-34.
[14] https://www.bsc.es/es/research-and-development/software-and-apps/software-list/pandora-hpc-agent-based-modelling-framework
[15] Michael Batty. Agent-based pedestrian modeling - editorial.Environment and Planning B: Planning and Design, 28:321–326, 02 2001
[16] Fasheng Qiu and Xiaolin Hu. Spatial activity-based modeling for pedestrian crowd simulation.SIMULATION, 89:451–465, 04 2013.

Figure 2.17 Execution flow diagram of an agent's decision process.

The blue path represents the decision-making flow on a regular match day at the facility. At its beginning the agents check if they have arrived at their target (which may not be final, there could be intermediary targets). If so, two situations are possible. If the match has not finished yet and the target is the agent's corresponding seat, it stays in place for as long as the match is being played. If the final target is an access, then the agents leave the simulation as they have exited from the stadium. Conversely, when agents haven't reached their target, they move to the next position toward it. The strategy to do so depends on various factors and group dynamics. Depending on the personal attributes of each agent, they may follow the optimal path or they may follow the crowd around them. This includes some code-level optimizations that trade-off accuracy in the

optimal path estimation with simulation performance, while maintaining a coherent and accurate movement of the Agents[17]. Arguably, this trade-off enhances the simulation as it provides a better approximation of how people think. Many people get distracted and move in an optimal way just in their line of sight or for short periods of time, then when they realize where they need to go, they recalculate the optimal path within these parameters. However, in some situations people just follow the trend they see around them.

The yellow path represents the evacuation scenario. All agents set their target to the nearest exit doors. If there are no close exit doors, the targets are set to the closest stairs to a lower level. Then all agents follow the aforementioned strategy regarding optimal paths and crowd behaviour. Since the objective is to determine the bottlenecks of the system, following this strategy stresses the simulation on the areas that can present more threats in a real evacuation situation[18].

2.4.2 Agent Synchronization

In a typical simulation, the decision process of each agent is executed one after another. Being a straightforward way of implementing a simulation, it is not a suitable approach for a large number of agents (~100K), since this kind of simulation will take a very long time. Besides, a sequential approach is hardly scalable, and it will have issues sooner or later when increasing the number of agents involved. Furthermore, the shorter time it takes to complete a simulation the shorter it takes for the bugs to be fixed. This means optimal developing timings, a key point when dealing with huge scenarios.

We developed a new method to run the simulation (known as a *scheduler*) that allows us to perform the simulations in parallel. The approach, schematized in Figure 2.18, is based on the concept of *Space Partitioning*: the simulation space (such as each level of the stadium) has been split off in terms of physical 2D space. Each of the resulting partitions has been assigned to an MPI process, which communicates with other processes only when needed.

[17] Soraia Musse and Daniel Thalmann.A Model of Human Crowd Behavior : Group Inter-Relationship and Collision Detection Analysis, pages 39–51. 01 1997.

[18] Pau Fonseca i Casas, Josep Casanovas, and X. Ferran. Passenger flow simulation in a hub airport: An application to the Barcelona international airport. Simulation Modelling Practice and Theory, 44:78–94, 05 2014.

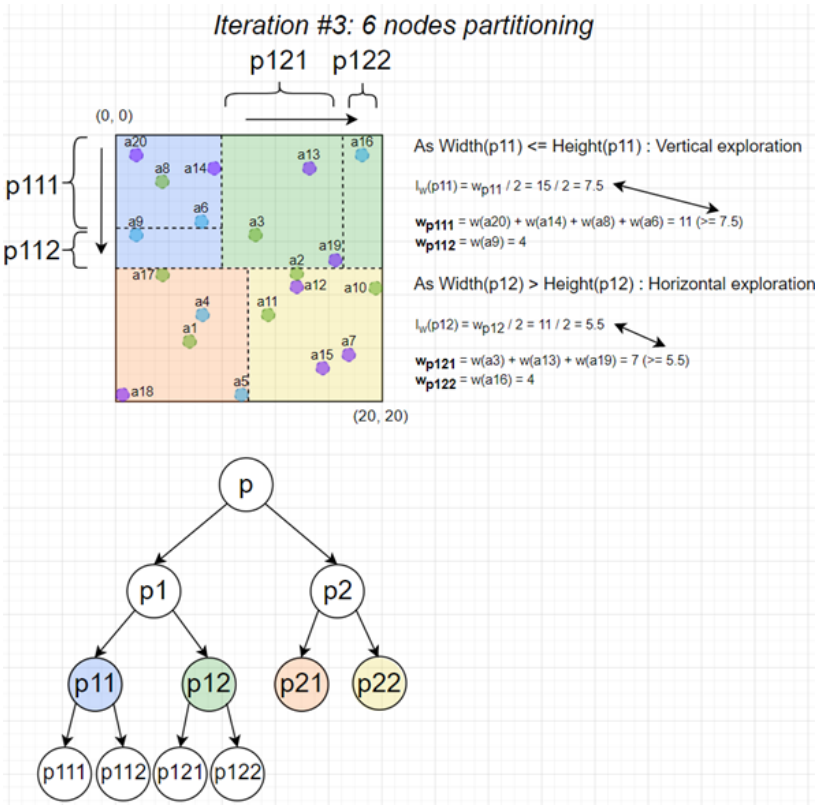


Figure 2.18 Schema of spatial partitioning.

The partitioning algorithm tries to split the space in a balanced way, considering the current state of the agents (their position and type, mainly). From there on, the simulation runs in parallel, i.e., agents being executed simultaneously in time. To avoid agent collisions when they move from one cell to another near the boundaries, the internal space of each MPI process is, in turn, also split. Then, these partitions are executed in a particular order: all partitions with the same identifier at the same time, e.g., partitions with ID #1 are executed. Then, the modified agents in that partitions are sent to their adjacent nodes if necessary. When a node has sent/received all the agents to/from its neighbouring nodes, and only then, its partition with ID #2 is executed, and so on.

Then, boundaries between processes are replicated for each node. We name these areas the *overlap* surfaces. In the literature, agents in these areas are called *ghost agents* since they are not coherent all the time among their copies if you consider an arbitrary moment within a time step of the simulation. They are effectively synchronized at the end of each step, *i.e.*, after executing all the node sub-partitions. The overlap area is a parameter that depends on the model and it is usually set to the maximum moving length of the agents.

Finally, the space is rebalanced recurrently along the simulation. This period is stated as a parameter too (e.g., each 100 simulation steps). Both the partitions size (height and width) and the number of processes is revaluated in order to get the smallest possible execution time regarding the current state of the simulation. It is important to point out that a bigger number of MPI processes does not always mean a better performance. Each process needs to be synchronized, which implies an overhead. Besides, the scheduler tries to make balanced square-like partitions which heuristically generates a lesser amount of overlap surfaces. In the way it currently works, this might imply that adding a non-power-of-two number of processes could be in fact counterproductive.

2.4.3 Output & Reporting

Because of the non-deterministic nature of the agent-based simulations, we must measure the outcomes of thousands of simulations for each scenario under analysis. This *Scenario Assessment* is done in Pandora with two tools: *Replay Extraction*, which is used for *Visualization* purposes, and *Aggregated Simulation Evaluation*, where we quantify the quality of the simulation using external data, particularly, *Visitor Counts per Camera*, considering the design of the project where Edge devices will report crowd behaviour to the system in real time. During the IoTwins project, we have improved the functionality of Pandora in these tasks according to the needs of the testbed: we implemented outputs for the variables of interest, we included postprocessing capabilities to measure inside the code the metrics while simulation is running, and we developed post-processing scripts to convert Pandora's output format into standard CSV format so that it can be used and read in other software.

The model will be validated when comparing the simulation results with the historical data and Key Performance Indicators (KPIs) defined in the project. This task will be done in the Scenarios Evaluation task.

2.4.4 Outcomes

Finally, this group of tasks aggregates, analyses, and contextualizes the results from the previous groups into concrete outcomes aligned with value creation for the partners involved. These outcomes are *Demand Prediction*, *Evacuation Report*, and *Arrival Management Optimization*. The first two have results available.

Demand Prediction focuses on providing forecasts that aid in the planning of future operations of the venue: *Spectator Placement Distribution*, *Tourist Origin*, and *Attendance Volume*. Most of these results are directly determined from the output of *Spectator Profiling* and are already available. Its results have been shown in the previous sections.

Evacuation Report aims at providing Key Performance Indicators (KPIs) of potential evacuation situations in a given scenario: *Time to Evacuation* and *Agglomeration Factor*, as well as a visualization of hotspots of influx in the venue through *Plan Heatmaps*.

Both these modules are being implemented at the time of this report. The Arrival Management Optimization will be developed in the next iteration of the pilot. However, its functioning mirrors the *Evacuation Report*, as such, it will be straightforward to generate its output.

2.5 Applications

For testing, validating, and producing the first exploitations of the Digital Twin, we have developed three use cases that have a direct impact on different parts of the facility management operation.

2.5.1 Use Case 1: Arrival Management Optimization

Context: When people arrive before the start of a match there can be several agglomeration spaces and holdups leading to long queues, and potentially issues as people get stressed that they might not arrive to their seats on time. By forecasting problematic areas (which depend not only on the entrance location but also on the match type, time of day, audience profile, etc.) extra personnel can be dispatched beforehand and alleviate the situation.

Figure 2.19 Queues and holdups form at the entrance before matches

Goal: Measure attendee's arrival to stadium and their optimal path to seat/location.

Description: In this first iteration of the Digital Twin, we are able to measure current arrival to define a prediction model for attendees' arrival from the door/turnstile and in stadium optimal path to the assigned 'boca' (section of a group of seats).

Expected Outcome:
KPIs can be aggregated for all stadium and segmented for each sector:

- **Arrival forecast KPIs:** Total stadium arrival prediction; Relation of stadium's sectors arrival prediction; Relation of stadium's 'bocas' arrival prediction [per minute].
- **Optimal path (from door to seat) KPIs** aggregated for all stadium and segmented for each sector: Relation of average time prediction; Number of collisions between agents; Density; Agglomeration ratio; (and, ideally, a Heat map)

Pilot/test: Each stadium sector is different: volume, number of doors, distance from street access, floor plant distribution and available services. For a first test, we are planning to execute simulation in a concrete sector, concretely 'Gol Sud' that represents 15-20% of stadium's capacity and it's the closest part to crowded streets and accesses on match days.

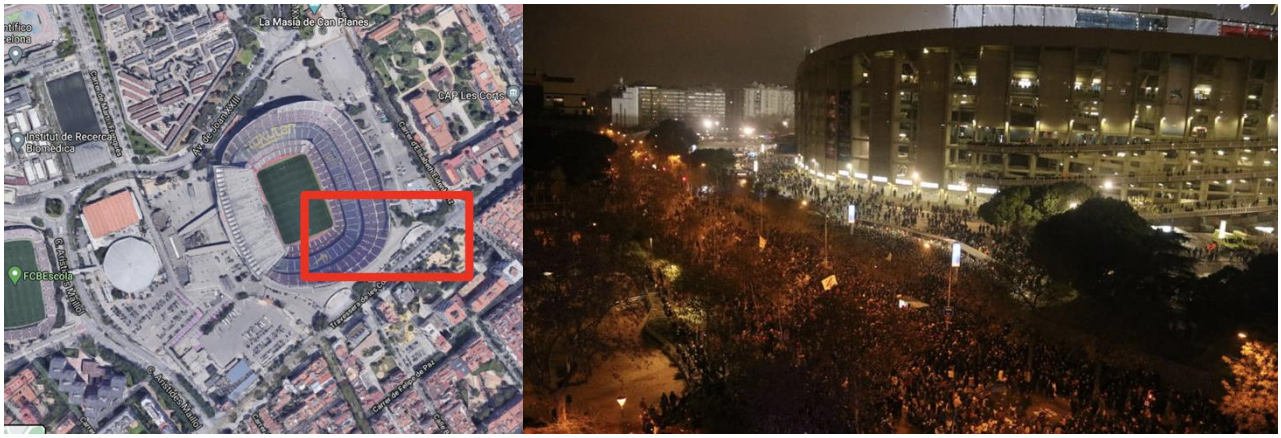


Figure 2.20 Images of Camp Nou's access nº 19 – Closest to Gol Sud sector

Considering an attendee's arrival path only from the stadium door to their seat, we can measure exactly the arrival and the density through turnstiles. Before and after crossing the door, with simulation, we can estimate agglomeration and queues and execute (manually) activators.

For the first DT iteration, this test can be validated directly with turnstiles, which have a much smaller range of error than the estimated Wi-Fi and cameras. For next iteration, we are planning to use a few people counting cameras (from an ongoing installation) in order to validate and add some extra aggregated profiling info, especially for outside stadium arrivals.

Actuators (on study / Edge level) are: As a basic orchestration test, we will send automatic messages to staff; Increase available doors assignation to affected attendees' tickets or passes; Next iteration: showing them new assignation using digital signage and notifying via app or SMS.

Planning: As long as the public restrictions on stadiums are maintained due to the COVID pandemic, tests will not be possible in real matches. Fortunately, there are options to simulate matches and also test with historical data.

2.5.2 Use Case 2: Evacuation and affectations management

(Stadium sector restrictions due to emergency scenarios or new stadium's works affectation)

Context: The FCB safety team is constantly working on preparing, adapting, and executing protocols (Pla d'Autoprotecció) for each potential emergency scenario. This task includes real simulations which, being a large facility capable of holding so many thousands of people, are difficult to execute. Although some specific estimations, currently the club does not have a tool with which to model and optimize the protocols for each scenario, much less to execute part of its tasks in a 100% automated way.

Figure 2.21 Simulation training firefighters emergency in stadium

Additionally, the FCB has designed a new stadium that must also comply with safety regulations, and it will be necessary to control the effects during the main works that will coexist with sports competitions and is expected to last for a few years.

Season/ Year	**Future Stadium's works main planned affectations** (only for crowd movement)
0	Demolition of access bridges between stadium 'Tribuna' and museum Construction of temporally access bridges directly to stadium outside
1	Execution of foundations works on Gol Sud and Gol Nord sectors Building 'Tribuna' sector evacuation/emergency exits through temporally access bridges directly to stadium outside Execution of foundations and structure works on Lateral sectors.
2	Building 'Tribuna' 1st floor & Gol Nord sectors temporally evacuation/emergency exits through VIP Boxes.

	Execution of stadium's perimeters structure works.
3	Building 'Tribuna' 1st floor & Gol Nord sectors evacuation/emergency exits through new vomitories. Building 'Lateral' 1st floor & Gol Sud sectors evacuation/emergency exits through VIP Boxes. Increasing 'General' 3d floor and Gol Sud capacity (number of seats). Building evacuation through new ladders.
4	Building 'Lateral' 1st floor & Gol Sud sectors evacuation/emergency exits through new vomitories. Increasing 'General' 3d floor and Gol Sud capacity (number of seats). Building final evacuation through new ladders.

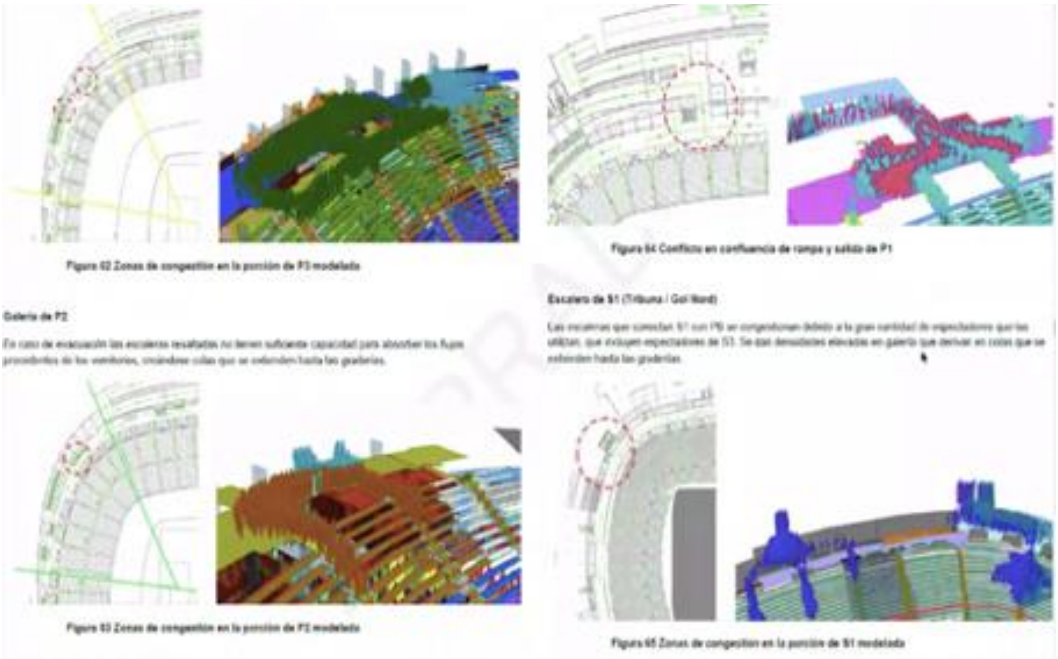
Goal: To measure attendee leaving path from seat/location to stadium door in evacuation of current stadium or works scenarios, and potentially improve the evacuation time.

Description: In this first iteration we're able to estimate current path to define a prediction model for attendees' exit from the seat / location to stadium door. At the exit of the stadium there is no ticket validation. Even in the final part of the game, the turnstiles are open to speed up exit. In emergency situations that require evacuation, the turnstiles will also be opened so we only can track with wi-fi data. For the next iteration, it is planned to add a few people counting cams in order to validate and add some extra aggregated profiling info, especially for stadium doors and street accesses.

Expected Outcome:

KPIs (Aggregated for all stadium and segmented for each sector of current stadium):

- Optimal path (from seat to door) KPIs aggregated for all stadium and segmented for each sector: Relation of average leaving time prediction; Number of collisions between agents; Density; Agglomeration ratio; (and, ideally, a Heat map)
- Comparison/validation to the current official evacuation plan and associated computer simulations



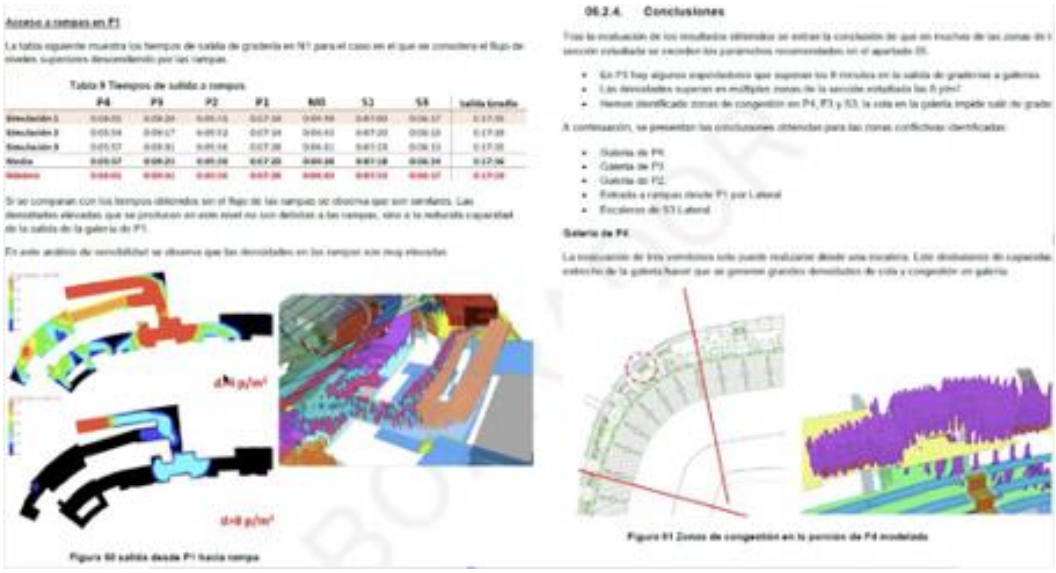


Figure 2.22 Evacuation Plans and 3d simulation (MassMotion) used by architects for stadium projection

Pilot/test: For a first pilot, we're planning to execute simulation with a scenario of only partial evacuation in order to avoid collapsing campus and closer streets, where we need to ensure mobility for emergency and operations teams' interventions.

Concretely, we plan to simulate and test at 'Gol Sud' that represents 15-20% of stadium's capacity and it's the closest part to crowded streets and accesses on match days.

Considering as attendees' evacuation path only from seat to stadium door, with simulation, it's possible to estimate agglomeration and queues in any point of the average path in different evacuation scenarios.

In reference to future stadium works, the conclusions of security simulations will impact and optimize the projection of future stadium. The same functionality can be used to simulate affected and blocked sectors due to future stadium works, combining both works and emergencies, or solving concrete needs of other stadium uses (e.g. music concerts).

For the first iteration of the digital twin, we only have available WiFi tracking to estimate the density at each point (and with considerable error bars). For the next iteration, FCB is planning to add a few people counting cams in order to validate and add some extra aggregated profiling info, especially for outside stadium to street evacuation.

Actuators (on study / Edge level) are: Actuators will work manually because all emergency tasks must be coordinated with the emergency and security authorities, who always assist and supervise the operation of any match at the stadium and have full knowledge of the legal requirements and limitations. Thus, automating actuators does not make sense for future stadium renovations. For an Edge level basic orchestration test, we can consider sending automatic messaging to staff, and ideally, using digital signage and notifications via app or SMS evacuations instructions.

Planning: As long as the public restrictions on stadiums are maintained due to the COVID pandemic, tests will not be possible in real matches. Additionally, real executions are conditioned to current security regulations. For that reason, we're only considering a first iteration based on simulated matches and testing models with historical data, as well as comparing to the official simulations done by the stadium designers and Catalunya firefighters.

2.5.3 Use Case 3: Attendance Forecast

Context: FC Barcelona currently uses some attendance estimation, mainly in four concrete moments:

- First, at the beginning of the season to define defining earnings expectations and operational plans, with an internal process that assigns category and tickets pricing to all known matches.
- Second, 2-3 weeks before the match when it is scheduled definitively (mostly for Spanish La Liga matches, not for UEFA).
- Third, from the end of the previous match (generally 5 days in advance), the main selling period starts because of media focus that increases interest and because locations start becoming available through the *Seient Lliure* program: members with season pass who report that they will not attend so FCB that can sell their seat and share benefits (see Figure 2.24).
- Finally, 72-48 hours before the match, when the main operational works begin, which are reviewed every few hours until the end of the match.

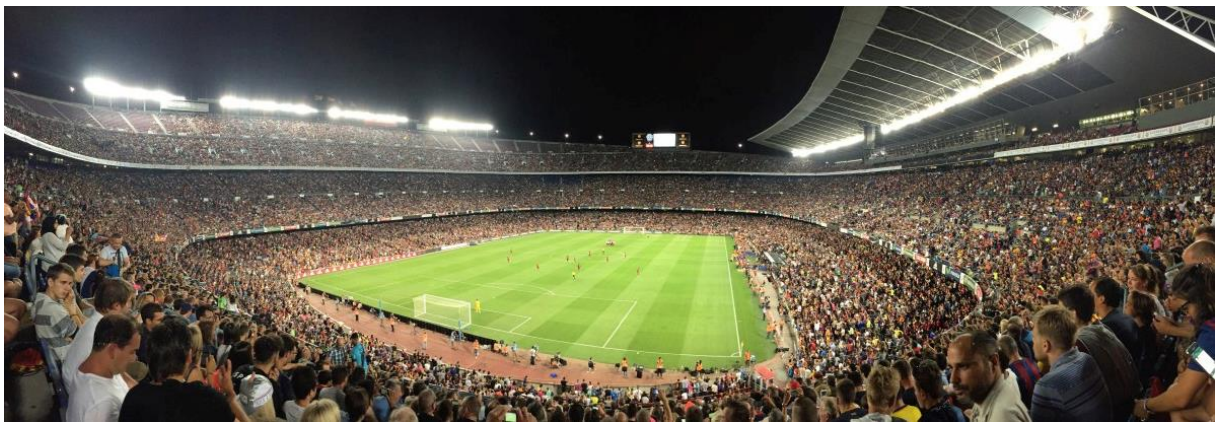


Figure 2.23 Camp Nou is the largest stadium in Europe, with an attendance that fluctuates strongly depending on complex factors

FC Barcelona attendance estimation (see Figure 2.25) is calculated 72-48 hours before the match with a relatively high reliability. The *Seient Lliure* forecast is based on historical data of each member. Considering that *Seient Lliure* activations increases stock of locations to sell, it conditions completely the resultant attendance. There is a relevant segment of members with season ticket called N-N (Not attending & Not SLL activation) that can create empty locations, so FCB encourages members to 'free' their seats. Other factors are: Ticket selling data with an extrapolation for next 72-48h; Number of invitations reserved; Number of child invitations requested (children accompanied by adults can attend the match for free, but they need an assigned location for security reasons); and number of tickets requested by the opponent team.

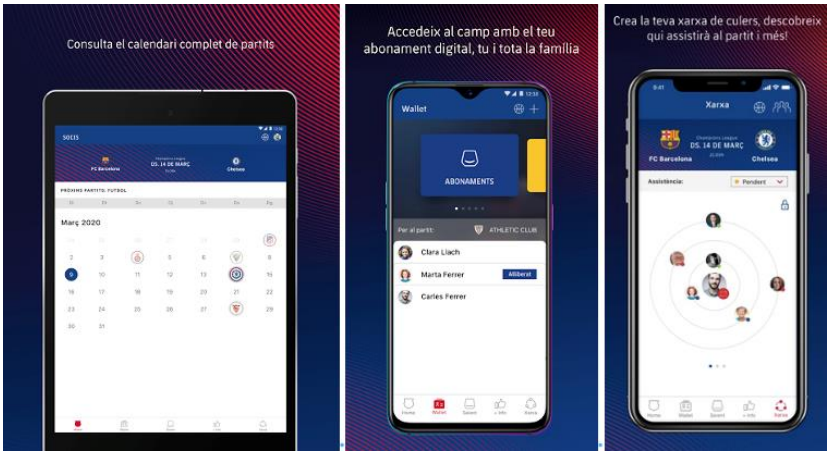


Figure 2.24 Seient Lliure app by FCB allows members to free their seats before a game

DATE	RIVAL	REAL ATTENDANCE	ATTENDANCE FORECAST 72-48h before match			ERROR		ABS ERROR	
	All Competitions	Official data	MIN	MAX	AV.	num	% Error	num	% Error
4-8-2019	ARSENAL	**98.812**	85.000	85.000	**85.000**	-13.812	-13,98%	**13.812**	**13,98%**
25-8-2019	REAL BETIS	**79.159**	80.000	83.000	**81.500**	2.341	2,96%	**2.341**	**2,96%**
14-9-2019	VALENCIA CF	**81.617**	80.000	83.000	**81.500**	-117	-0,14%	**117**	**0,14%**
24-9-2019	VILLARREAL CF	**70.316**	70.000	72.000	**71.000**	684	0,97%	**684**	**0,97%**
2-10-2019	FC INTERNAZIONALE	**86.141**	80.000	82.000	**81.000**	-5.141	-5,97%	**5.141**	**5,97%**
6-10-2019	SEVILLA FC	**81.331**	80.000	82.000	**81.000**	-331	-0,41%	**331**	**0,41%**
29-10-2019	REAL VALLADOLID	**59.896**	64.000	66.000	**65.000**	5.104	8,52%	**5.104**	**8,52%**
5-11-2019	SK SLAVIA PRAHA	**67.023**	67.000	69.000	**68.000**	977	1,46%	**977**	**1,46%**
9-11-2019	RC CELTA	**71.209**	75.000	77.000	**76.000**	4.791	6,73%	**4.791**	**6,73%**
27-11-2019	BORUSIA DORTMUND	**90.071**	90.000	92.000	**91.000**	929	1,03%	**929**	**1,03%**
7-12-2019	RCD MALLORCA	**71.072**	75.000	77.000	**76.000**	4.928	6,93%	**4.928**	**6,93%**
18-12-2019	REAL MADRID	**93.426**	98.000	99.000	**98.500**	5.074	5,43%	**5.074**	**5,43%**
22-12-2019	DEPORTIVO ALAVÉS	**63.054**	70.000	72.000	**71.000**	7.946	12,60%	**7.946**	**12,60%**
19-1-2020	GRANADA CF	**65.444**	75.000	77.000	**76.000**	10.556	16,13%	**10.556**	**16,13%**
30-1-2020	LEGANÉS	**43.216**	48.000	50.000	**49.000**	5.784	13,38%	**5.784**	**13,38%**
2-2-2020	LEVANTE UD	**60.295**	65.000	67.000	**66.000**	5.705	9,46%	**5.705**	**9,46%**
15-2-2020	GETAFE CF	**80.409**	80.000	82.000	**81.000**	591	0,73%	**591**	**0,73%**
22-2-2020	SD EIBAR	**66.970**	68.000	70.000	**69.000**	2.030	3,03%	**2.030**	**3,03%**
7-3-2020	REAL SOCIEDAD	**77.035**	83.000	85.000	**84.000**	6.965	9,04%	**6.965**	**9,04%**
		71.209	75.000	77.000	**76.000**	2.341	3,03%	**4.928**	**5,97%**

Figure 2.25 Attendance forecast and measured error for season 2019-20 using FCB internal methods

Goal: Define and attendance forecast algorithm with different levels of aggregations and including some attendee's profiling to optimize revenue and operational performance.

Description: FC Barcelona's match attendance estimations accuracy is acceptable, but currently it's only available on short-term 72/48h and aggregated for the whole stadium. FCB also have some profiling data for season tickets and regular tickets sold online. Excluding tickets sold on lockers, tour operators, some invitations, and opponent tickets.

To improve this process, FCB needs a segmented forecast for stadium sectors, *bocas (groups of locations)* or locations, including some profiling details, to optimize pricing and selling process, adapt commercial offer (cross selling) and to improve operation and security performance. (Note: a segmented forecast for each stadium door and the arrival hours and minutes before the match can be very useful for all use cases).

To have the whole picture for Digital Twin projection, in future iterations the simulation should add all people: sport teams, staff, press and media operators, security and stewards and also number and profiling of people that come around the stadium without ticket.

Expected Outcome:
KPIs Aggregated for all stadium and segmented for each sector and boca of current stadium**:**

- Attendance volume forecast KPIs
- Attendance profiling forecast KPIs: % of each profile
- Algorithm/model operational KPIs: relation of accuracy vs days in advance

Pilot/test:
A first phase consisting of monitoring and analysing a match complete life cycle, from confirmation to match day in order to detect and validate any behaviour model of potential attendees (see Figure 2.26). And a second phase, test its relation with current FCB process and simulate or execute some A/B test in a specific stadium sector.

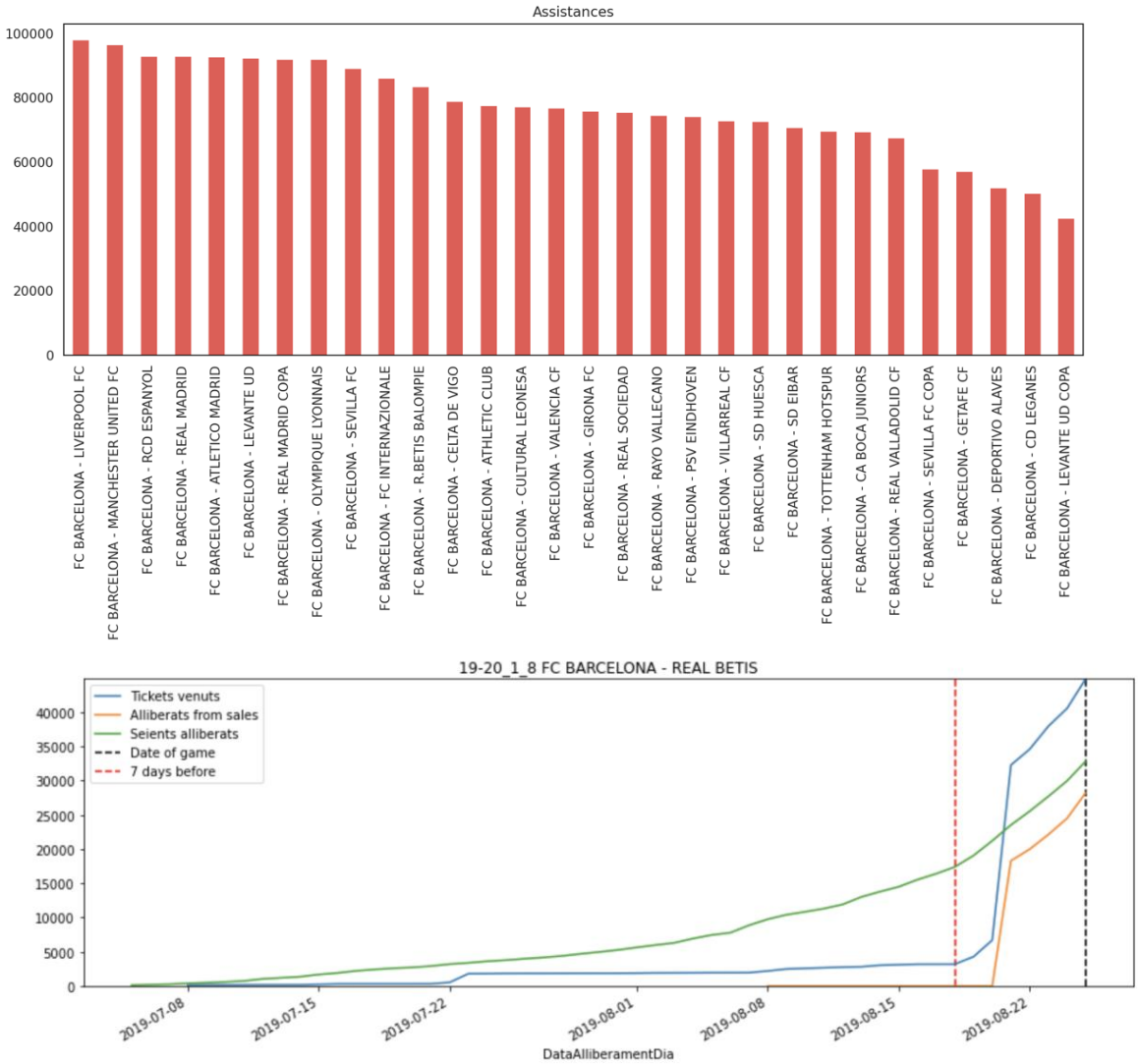


Figure 2.26 (top) Historical assistance to games and (bottom) typical behaviour of ticket sales (blue), freed seats (green), free seats sold (orange). Usually around 7 to 5 days (red dashed line) before the game (black dashed line) a reminder to free seats is sent.

Actuators (on study / Edge level): In order to test an actuator, for the first pilot phase we can consider sending automatic messaging to staff with descriptive data and available forecast. For the second pilot phase,

we can enrich messages with commercial and operational recommendations. As potential orchestrated activators adapted to volume and profiling forecast, there are: some sub process like activate *Seient Lliure* App push notification reminders; sending ticketing campaigns for specific stadium sector through CRM integration; activation of extra stocks for in stadium merchandising and food & beverage selling points; activation of last-minute offers to increase attendance; and modify some parameters of the FCB's dynamic pricing algorithm to increase revenue margin.

Planning: As long as the public restrictions on stadiums are maintained due to the COVID pandemic, tests will not be possible in real matches. Additionally, real executions are conditioned to current security regulations. For that reason, we're only considering a first iteration based on simulated matches and testing models with historical data.

2.6 Results

2.6.1 Influx prediction

The results from the LOOCV show that the Gradient Boosting Regressor (GBR) is the best predictor for the global stadium attendance, with 50 boosted trees and a maximum depth of 2, with an R2=62.67% and a mean absolute error (MAE) of 8112.77, followed by the Random Forest Regressor with an R2=60.27% and MAE=8211.06. Then, the Decision tree regressor has a R2=45.14% and MAE=8425.30, and finally the Ordinary Least Squares regression has an R2=34.7%and MAE=10661.02. In Figure 2.27 we show the performance of the GBR for all the games of the dataset. Even though the simpler models show larger errors, these provide better insights on the covariates used and allow us to understand the inaccuracies of these models. Lastly, all of the above algorithms show larger errors when predicting certain types of games that have strange numbers of visitors. To correct these errors, we should include new covariates that capture this unexplained variance, and also add timely covariates with a certain number of days before the game, as these might help predict the attendance more accurately.

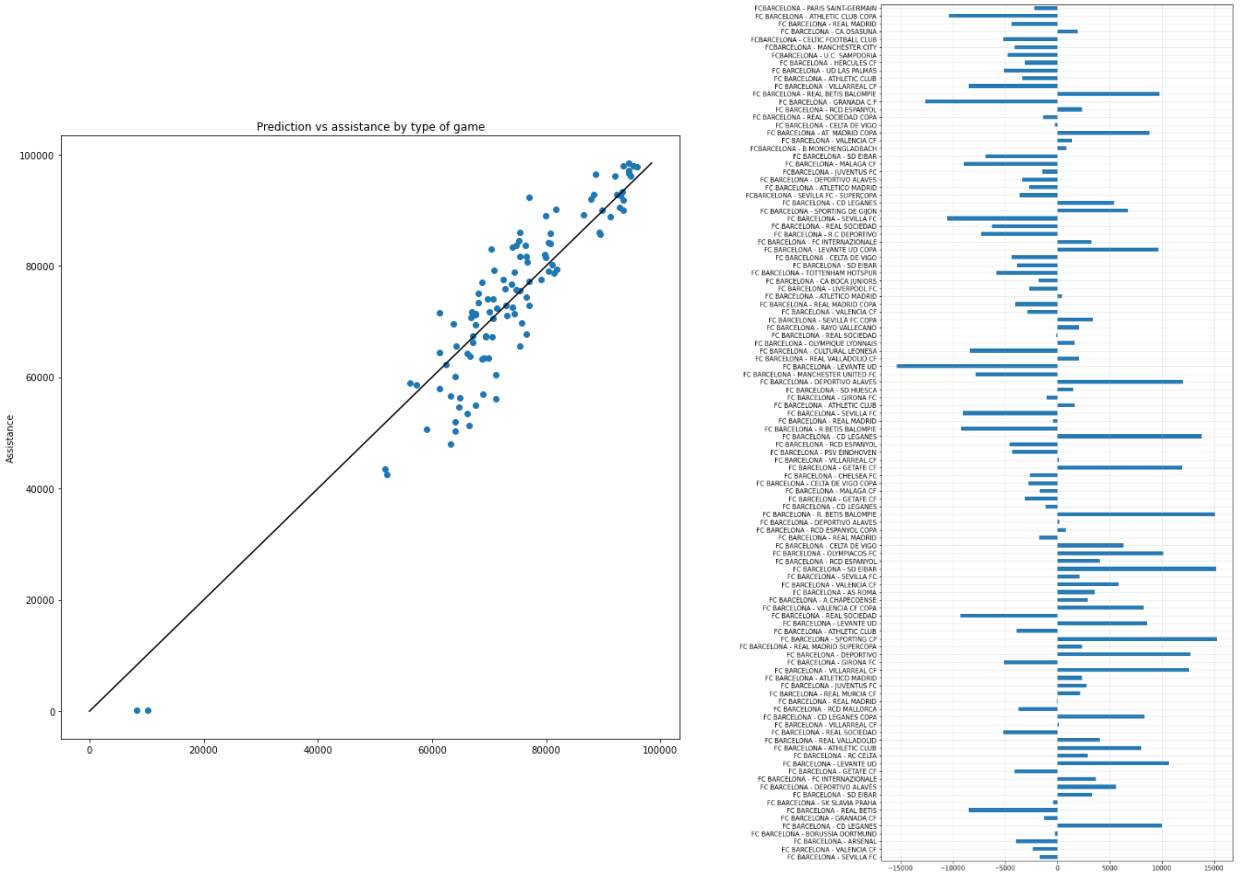


Figure 2.27 (Left) Predicted attendance (horizontal axis) vs actual attendance (vertical axis) for the gradient boosted regression model. (Right) Residuals by game for the 2016-2020 period

In Figure 2.28 we can see the performance of the model by gates, comparing prediction vs observation for a single gate, and showing that the R^2 coefficient varies greatly among the gates between 0.4 and 0.8, with a large peak around 0.65. The predictability by gate appears to show some interesting spatial patterns, as shown in the Figure, which might be interesting to explore.

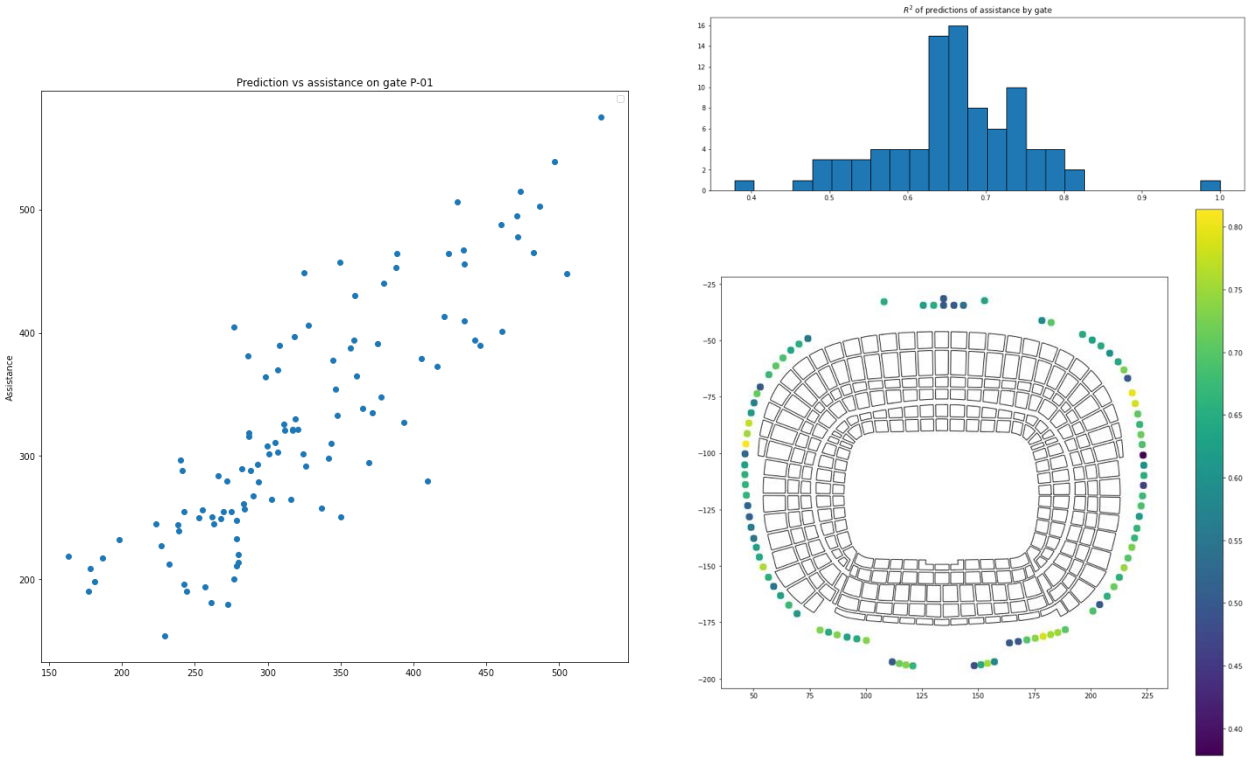


Figure 2.28 (Left) Predicted attendance (horizontal axis) vs actual attendance (vertical axis) for a single entrance gate at the stadium. (Top right) Histogram of R2 values for the different gates. (Bottom right) Plot of the R2 values for the prediction of each entrance.

2.6.2 Attendance profile prediction

Using the methods described above based on telephony data we are able to predict the proportion of attendance (Figure 2.29) that comes from the local metropolitan area (AMB), from the Catalunya region, from Spain, or their continent of origin with sufficient accuracy.

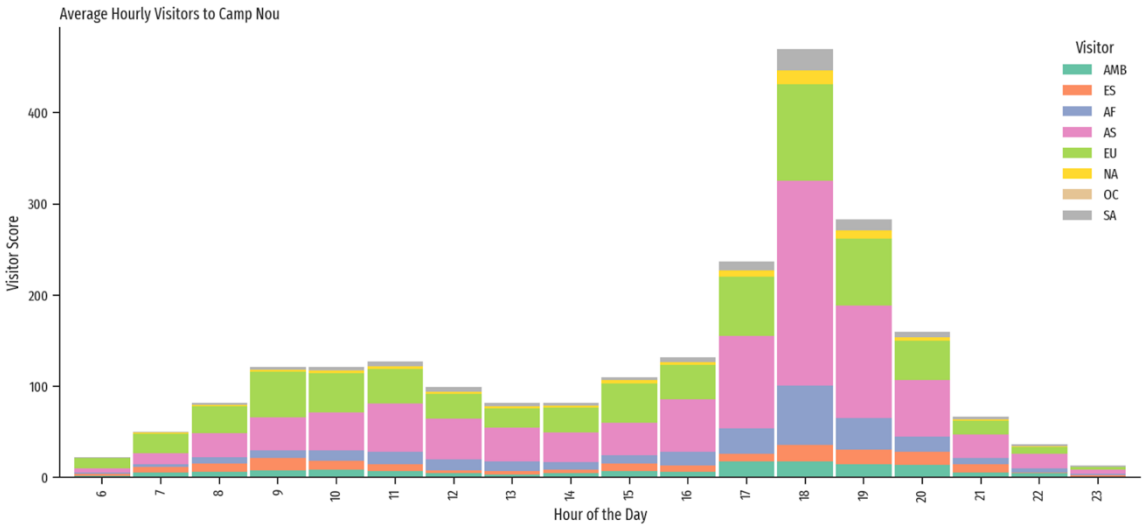


Figure 2.29 Distribution of visitors at Camp Nou according to their origin and time of the day.

From the same data we can compute roughly the gender, age bracket, or origin distribution probabilities in the whole city as a function of time of the year, giving us an estimate of the profile of the visitors at Camp Nou (Figure 2.30).

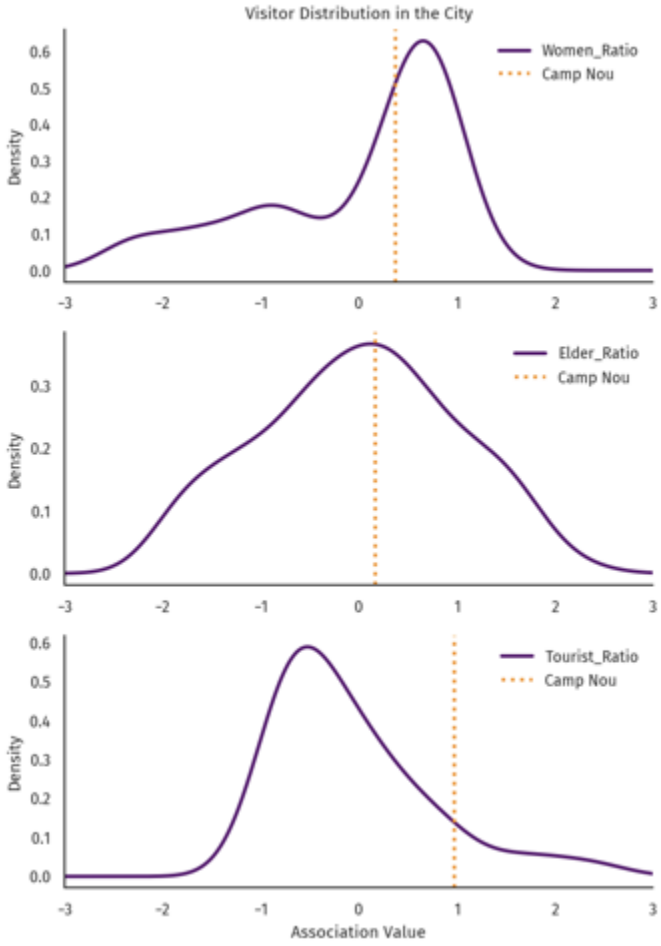


Figure 2.30 Density of association value of a given subgroup with places around the city (dark line) compared to the value found at the Cap Nou cell. While the ratio of elders to other population is similar to the average value in the city, there are slightly less women than average and quite a few more tourists.

Because these estimates are probabilistic, for each run of the digital twin we draw a different demographic profile from the probability distributions resulting from this analysis.

2.6.3 Agent based simulations

We are currently using and testing the Digital Twin from end to end, identifying technical issues, debugging code, and measuring performance. For this reason, we have not yet simulated a full match but rather short episodes of a few minutes of real time (see e.g., Figure 2.31 for a one-minute simulation of entrance).

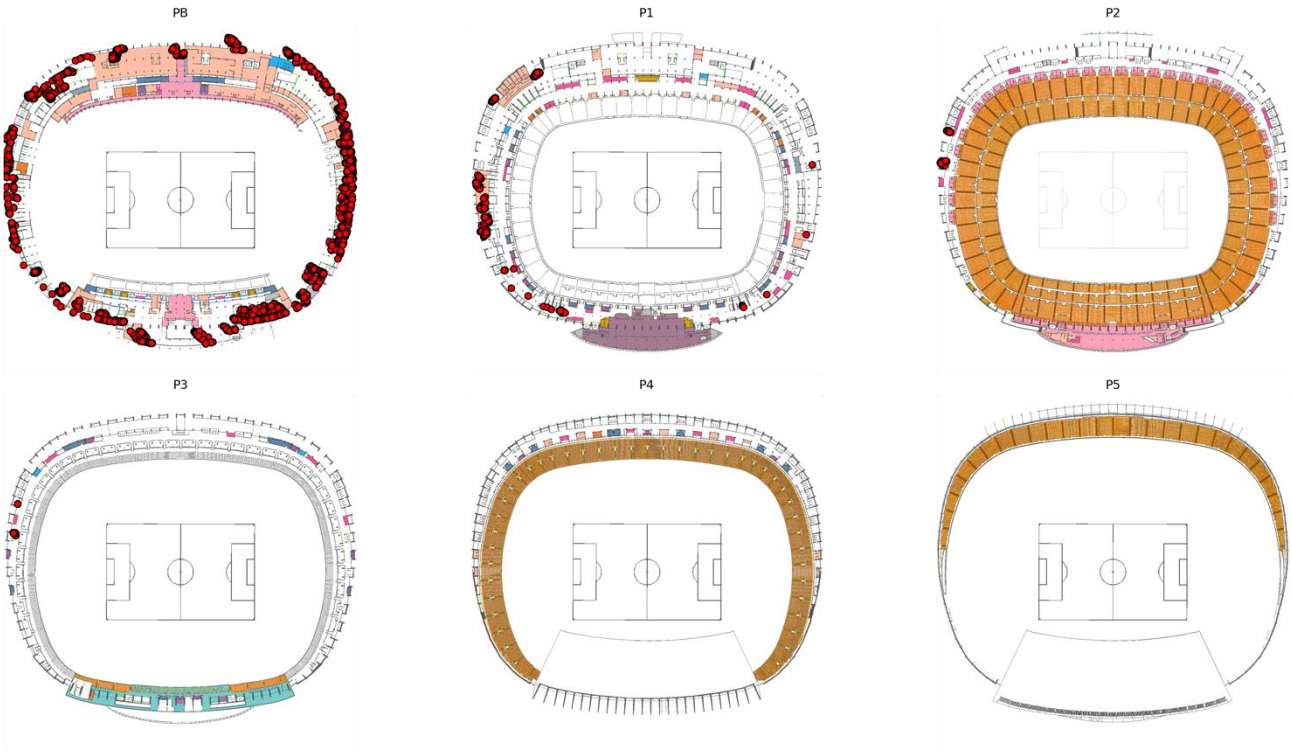


Figure 2.31 Partial result, state of the stadium different levels after 4000 agents enter through gates and head towards their seats.

Upon entering the building, the agents pick a path from their surrounding and guided by a general direction where they have to go. We show in Figure 2.32 an example path for a single agent going from entrance to seat.

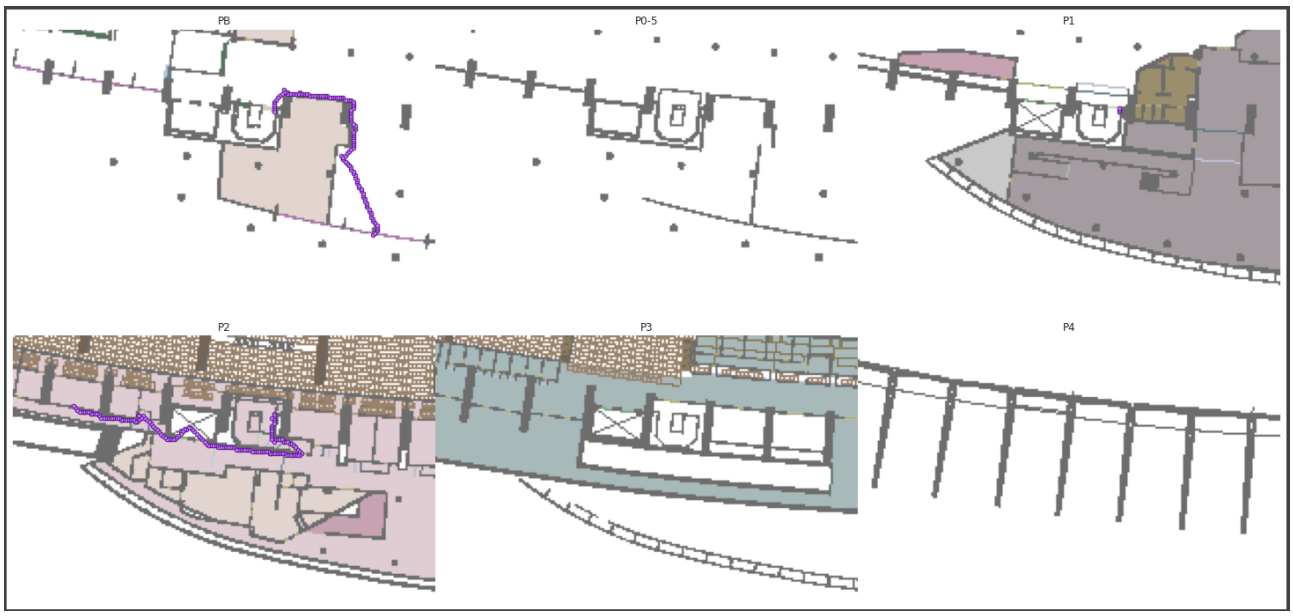


Figure 2.32 Sample path for an agent that enters the stadium and goes to their assigned seat. The agent steps, marked in purple dots, enters through the entrance at floor level PB (top left), and has a seat at the level 2 (P2). In floor PB the agent goes from the door to the closest stair that communicates with P1 and then P2, leaves the stair and goes to the section where their seat is.

Once results are validated, we will turn to improving the parallel performance of the code. Current results show that indeed the simulations improve as we increase the number of processors (see Figure 2.33), but

mostly when the number of agents per processor is large enough to compensate for the inter-node communications.

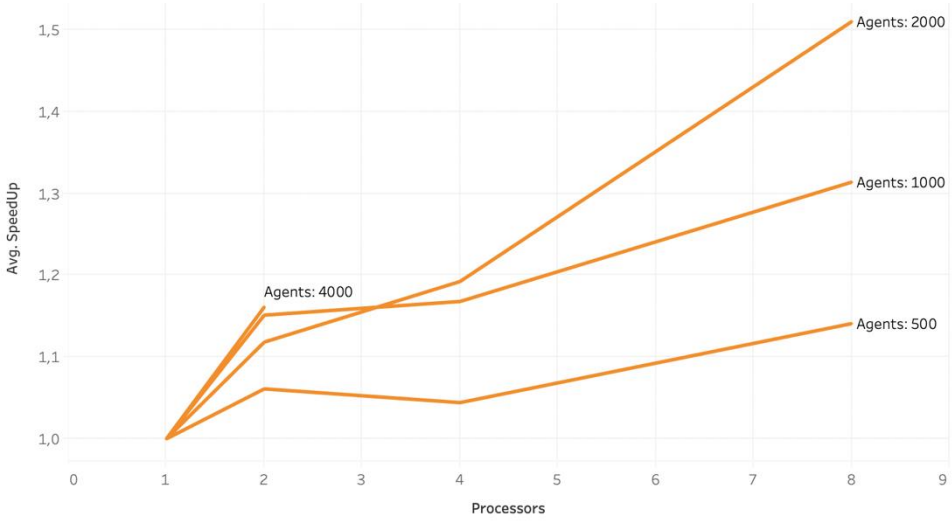


Figure 2.33 Speedup in Pandora's total simulation time as a function of number of processors and number of agents. The speedup profile is better for larger number of agents as the parallelization is more efficient (less communication to computation ratio).

Only when the simulations are validated, we will turn to improving the performance, functionalities, and to obtain final results for the use cases developed (see section above).

3 Testbed N6

3.1 General description of the system

CINECA is one of the HPC Large Scale Facilities in Europe. Its hardware resources are one of the most powerful available in Italy and among the most powerful available in the world. The HPC environment in CINECA is made of general-purpose computers constantly maintained at the technological leading Edge. They are integrated into a common working environment to allow users easy access, everyday working and portability of data and applications among different platforms.

At present, the main HPC system (Tier-0 level) is MARCONI100, an accelerated cluster based on IBM Power9 processors and NVIDIA Volta GPUs (980 nodes) and MARCONI, a LENOVO system with Intel Skylake architecture (3188 nodes). The set-up of MARCONI (including MARCONI100) started in June 2016 and was completed in April 2020. The final performance of MARCONI will be around 38 PFlops peak. Along with MARCONI, a Tier-1 level system, the GALILEO system equipped with Intel Xeon E5-2697 v4 (Broadwell) nodes.

During the IoTwins project, the datacenter has been evolved to offer users the state-of-the-art HPC technologies. So, the data collected in testbed 6 involved other two HPC clusters, MARCONI KNL and DAVIDE, both replaced by MARCONI100.

All the systems share a large storage infrastructure for medium-long term archive (about 20 PB of disk storage and 20 PB on magnetic support). The CINECA HPC environment is completed and placed side by side to a HPC-Cloud infrastructure, managed by OpenStack, to offer users flexible and self-configurable services.

3.1.1.1 Galileo

The GALILEO supercomputer has 1022 compute nodes. Each one contains 2 x 18-cores Intel Xeon E5-2697 v4 (Broadwell) at 2.30 GHz. All the compute nodes have 128 GB of memory. Of these compute nodes, 60 are equipped with nVidia K80 GPUs and two with nVidia V100 GPUs. There are 8 Login & Viz nodes available (5 are academic nodes and 3 are for industrial users). There are 8 service nodes for I/O and cluster management. All the nodes are interconnected through a high-performance network Intel Omni-Path, capable of a maximum bandwidth of 100Gbit/s.

3.1.1.2 Marconi

MARCONI is a Tier-0 system, co-designed by CINECA and based on the Lenovo StarkRacks platform. It is composed of 45 racks. It has 3188 nodes equipped with 2 x 24 cores Intel Xeon 8160 CPU (Skylake) at 2.10 GHz, 153.024 cores in total. All the compute nodes have 192 GB of memory DDR4. There are 3 login nodes available for regular users. Each one contains 2 x Intel Xeon Processor E5-2697 v4 with a clock of 2.30GHz and 128 GB of memory. The cluster is managed by SLURM server. The peak performance of the single node is 3.2 TFLOPs while that of MARCONI is about 10 PFLOPs.

3.1.1.3 Marconi KNL

The Marconi KNL (Knights Landing) cluster was based on the Lenovo Adam Pass model and composed of 50 racks. It had 3600 compute nodes equipped with 68-cores Intel Xeon Phi 7250 CPU (Knights Landing) at 1.40 GHz, 244.800 cores in total. All the compute nodes have 16 GB of MCDRAM, directly on Xeon Phi, and 96 GB of memory DDR4. All the nodes are interconnected through a high-performance network Intel Omni-Path, capable of a maximum bandwidth of 100Gbit/s. MARCONI KNL peak performance reaches 11 PFlop/s.

3.1.1.4 M100

M100 is an accelerated cluster based on Power9 (IBM Power AC922 (Whiterspoon)) processor and Volta NVIDIA GPUs, acquired by Cineca within the PPI4HPC European initiative. M100 is composed of 55 racks in total of which 49 computing. It had 980 nodes equipped with 2x16 cores (SMT 4) IBM POWER9 AC922 at 2.6 GHz. The compute nodes have 256 GB of memory. M100 has 8PB raw GPFS disk space storage. There are 8 Login IBM Power9 LC922. The peak performance of M100 (980 compute nodes + 8 login nodes) is about 32PFlops. The performance of the single node is 32 TFlops. The nodes are interconnected through a Mellanox Infiniband EDR DragonFly+ network.

3.1.1.5 Davide

D.A.V.I.D.E. (Development of an Added Value Infrastructure Designed in Europe) was an Energy Aware Petaflops Class High Performance Cluster based on Power Architecture and coupled with NVIDIA Tesla Pascal GPUs with NVLink. It was composed of 45 nodes connected with an efficient Mellanox Infiniband EDR 100 Gb/s networking. DAVIDE had total peak performance of 990 TFlops and an estimated power consumption of less than 2kW per node. Each node was a 2 OU OCP form factor and hosted two IBM POWER8 Processors with NVIDIA NVLink and four nVIDIA Tesla P100 (Pascal) GPUs, with the intra-node communication layout optimized for best performance.

3.1.2 Large-scale data center specific challenges

The EXAMON monitoring system can collect a lot of information regarding computing activities, power consumption related both to each single HPC cluster and the whole datacenter infrastructure. Examon is focused on collecting the precise information about nodes, power sources, temperature but does not integrate them with the information about HPC users and the types of workload.

From the point of view of system administration, at the moment EXAMON interfaces with other monitoring system frameworks, such as Nagios or Ganglia, to take timeseries of the workload of HPC nodes, IO and network traffic. The challenge would be having a system that interfaces the previous mentioned data with users' behaviour through log analysis. Another important challenge would be using it in a production environment. The first step would be splitting the project in two paths, one with a stable release to use it in production and to develop new features to include in in the stable version in the near future.

From the point of view of facility management, at the moment EXAMON collects data of metrics related to power consumption and performance of the server in the computing rooms. The Examon data made it possible for CINECA to obtain the ISO 50001 certification and was also used during the testing of new server clusters. The challenge would be having a spatially model with more information about the power consumption and performance of the different components of the electrical and mechanical plants that provide power and cooling to the server. Furthermore, would be useful collect data about the temperature and relative humidity of the server rooms and tools to carry out simulations related to the reconfiguration of the room (e.g. predict the temperature level that would be reached by installing new racks in the server rooms).

3.2 EXAMON monitoring infrastructure

The first step towards the creation of a digital twins for a supercomputer consists in endowing the target system with a fine-grained monitoring infrastructure, as this allows to collect a large amount of data which can be used to characterize the HPC system and to create accurate models; for test-bed N.6 the data collection framework is EXAMON. This is the activity conducted during for Task 5.2.1. In this deliverable we describe in a detailed how the infrastructure was deployed on CINECA machines, illustrating how EXAMON's architecture adopted some of the characteristics proposed by the IoTwins platform defined in WP2. For a more detailed description about the collected data (e.g., metrics and measurements, sampling rate, number of data sources, amount of data generated and stored per day, etc.) we refer to a previous deliverable, namely D5.1.

EXAMON is a holistic framework for HPC facility monitoring and maintenance19, designed for very large-scale computing systems, such as supercomputers. It has been developed for the Exascale, thus stressing the capability to handle big data from many heterogeneous sources. At the lowest level, there are components to read the data from several sensors scattered across the system and deliver them, in a standardized format, to the upper layers of the stack. There are collectors with direct access to hardware resources and collectors that sample data from other applications, such as batch schedulers and software diagnostic tools. EXAMON has been deployed on CINECA machines since 2017[20].

The infrastructure is built using the MQTT protocol. The collected metrics are stored on a distributed and scalable time series database (DB), KairosDB, built on top of a NoSQL DB, Apache Cassandra as back-end. A specific MQTT subscriber (MQTT2Kairos) is implemented to bridge the MQTT protocol and the KairosDB data insertion mechanism. The bridge leverages the MQTT topic structure to compose the KairosDB insertion statement automatically. This gives a twofold advantage: first, it lowers the computational overhead of the

[19] Bartolini et al., "The D.A.V.I.D.E. big-data-powered fine-grain power and performance monitoring support", Proceedings of the 15th {ACM} International Conference on Computing Frontiers, {CF} 2018, Ischia, Italy, May 08-10, 2018

[20] Bartolini et al., "Paving the way toward energy-aware and automated datacentre", Proceedings of the 48th International Conference on Parallel Processing: Workshops, 2019

bridge since it is reduced to a string parsing operation per message; and secondly, it makes it easy to form the DB query starting only from the knowledge of the matching MQTT topic. The sampling rate of the metrics measured by EXAMON can vary among the different metrics, depending on the underlying sensors. However, EXAMON aggregates the data in 5-seconds windows; namely, a data point with time-stamp t represents the average of measurements sampled over the [t - 5s, t] window. Every five seconds a MQTT packet is built and sent to the broker; EXAMON low-level plugins are in charge with aggregating sensors measurements with higher sampling rates.

EXAMON architecture follows the well-established *Lambda architecture*[21], widely adopted to overcome the limitation of traditional approaches in handling big data and IoT requirements. Traditional analytics frameworks are based on batch processing style: firstly, data is collected in a database (DB), then applications process the data in chunks. This approach is not effective when answers are needed from the data in short time. Conversely, stream processing provides fast responses as no intermediate storage layer is needed. Lambda architecture offers the best of both worlds: it is formed by a ``batch'' layer, where the data is stored in a distributed database and processed in a batch fashion. On the other side, a ``speed'' layer provides stream processing capabilities on the data flow, enabling fast processing of data as it arrives. Following the Lambda architecture, EXAMON is composed by a set of five layers, as depicted in Figure 3.1; the rest of the section is devoted to describing the five layers.

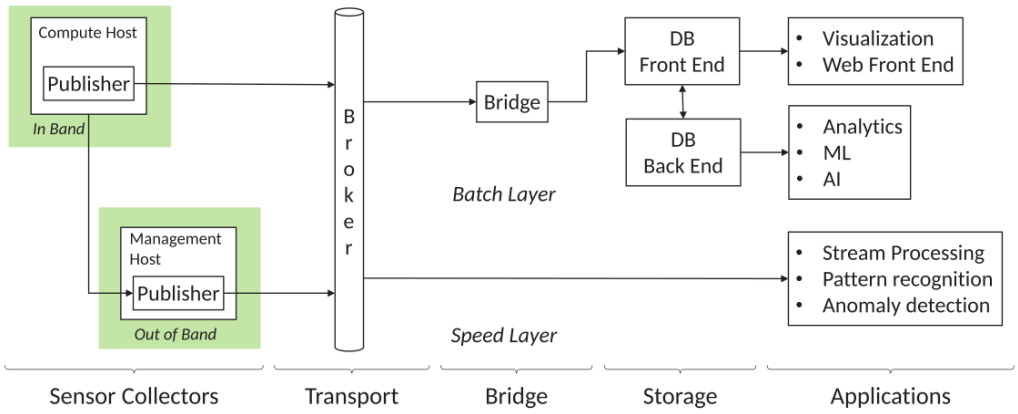


Figure 3.1 EXAMON Lambda Architecture

3.2.1 Transport Layer

At the heart of the framework, there is the data transport layer, responsible for the data sharing between all the framework components. Moving data has a cost in terms of network bandwidth, energy, protocol handling and scalability. To contain with these costs, we leveraged an IoT approach, where sensor networks and scalable communications protocols are well established; namely, we adopted MQTT (MQ Telemetry Transport)[22], a TPC/IP-based protocol designed for low-bandwidth and high-latency networks with minimal

[21] "Lambda architecture for cost-effective batch and speed big data processing", Kiran et al.,IEEE International Conference on Big Data (Big Data), 2015
[22] http://docs. oasis-open. org/mqtt/mqtt/v3

resource demands, thanks to the small size of its packets. From the application point of view, MQTT delivers (pushes) messages following the publish-subscribe mechanism, already proven to be effective in IoT scenarios.

This mechanism is very effective in applications where sensor values need to be pushed, as soon as they are available, to the consumers. A typical topology is composed of a server (broker) and many clients (publishers and subscribers); when a publisher sends a message (payload) to a topic on the server, the message is delivered to each client that previously subscribed to that topic. The protocol implements three Quality-of-Service (QoS) levels and a security layer (SSL), making it suitable to be deployed in a production HPC system.

Topics names are strings defined in a hierarchical form and follow the filesystem path convention.

3.2.2 Sensor Collectors

The sensor collectors are the components responsible for retrieving the data from the sources and share it with the rest of the framework.

In HPC systems the data sources are very heterogeneous, ranging from numeric metrics (e.g., the CPU performance counters or physical sensors) to textual events such as logs generated by the job schedulers and resource managers, passing through alerts generated by profiling tools. Thus, EXAMON is endowed with multiple sensor collectors, that is SW agents, devoted to different sources; each agent is composed by a ``MQTT API'' object and a ``Sensor API'' object. The former represents an abstraction layer between the transport layer and the specific sensor logic -- it has mostly the same implementation for all sensors; the latter is tailored on the specific source. This abstraction and decoupling greatly simplify the development of new collectors for new data sources.

The sensors collectors can be grouped into two types: 1) in-band, executing on the compute nodes, and 2) out-of-band collectors, running on nodes not involved in computation (e.g., service nodes). In-band collectors must have minimal impact on the local node resources, to guarantee the lowest overhead possible on users' applications.

3.2.3 Protocol Bridge

A standard transport layer is essential when dealing with a vast variety of heterogeneous data sources, each having its specific protocols details; a standard data bus interface allows to abstract the heterogeneous data sources and simplify the access for higher-level components accessing the data. This is the role of the protocol bridge, that has two tasks: it translates the specific metric source complexity to the transport layer and, on the other side, translates the transport protocol to the data sinks.

3.2.4 Data Storage

The storage layer stores the data coming from the several sensors source and provides a data back-end for the end user application layer such as data visualization and predictive maintenance. Considering the large number of sensors involved in an exascale sized HPC cluster, it should have a distributed architecture, able to scale horizontally to seamlessly cope with the growing input load. Moreover, it should be capable of adding new data types (e.g., new sensors) without changing the storage settings (e.g., the table schema). For these reasons, we opted for a NoSQL DB, Apache Cassandra[23].

[23] https://http://cassandra.apache.org/

3.2.5 EXAMON Components

EXAMON components follow a master-worker model to exploit available HW resources while maximizing throughput; the master sets the execution environment and assigns the jobs to a pool of workers. The jobs are executed entirely by the workers and, during the execution, the master runs only health-checking tasks for the workers. In this architecture, the message broker represents the input/output data queue that feeds the workers or receives resulting data. In EXAMON there may be publishers, subscribers, and stream processors components, described in the rest of this section.

Publishers are the SW components in charge of collecting sensors data, form the payload and delivery it to the transport layer. The publisher worker contains two objects: the sensor API that implements all sensor-specific routines and the interfaces to EXAMON transport layer. The MQTT API implements all the communication routines and acts as an abstraction layer between the sensor data buffer and the low-level routines provided by the MQTT protocol. For each different type of publisher, we specialize the sensor API implementing the specific sensors routines while the MQTT API remains almost the same.

3.2.5.1 Pmu_pub

Modern high-performance processors provide HW support to profile and optimize their applications. In general, the HW offers an interface to measure and collect information about the CPU performance state; for instance, in Intel processors this HW unit is called Performance Monitoring Unit (PMU). Multi-core processors expose multiple PMUs, one for each core plus one dedicated to the uncore. Performance metrics are called "events" that can be evaluated by accessing the PMU through its software interface. Events can be programmed and evaluated mainly using a series of dedicated MSRs (Model Specific Registers) subdivided into two categories: control and counter register. The first are used to set the specific event type to be measured while the latter are used to read the event value.

The target of the pmu_pub plugin (structure shown in Fig. 3.2.2) is to expose all the performance, temperature, and energy metrics to the communication layer in a common and easily accessible format. In order to do this, we developed the user space application in a modular way to support several libraries and modality to efficiently access the CPU registers while maintaining a low impact on the system resources. The pmu_pub plugin executes as a Linux service on the compute nodes. It is developed as a single thread process. It starts by reading the configuration file specifying the set of events to be logged, retrieve CPU info from the OS and sets the MQTT communication parameters. Then, it continuously samples the counters corresponding to the programmed events, storing them in memory. Afterwards, the data is serialized according to the data format and transmitted to the MQTT broker. Sampling and publication are performed with a user-defined sampling rate (e.g., we adopted 1ms); synchronization across all computing nodes is obtained via NTP protocol.

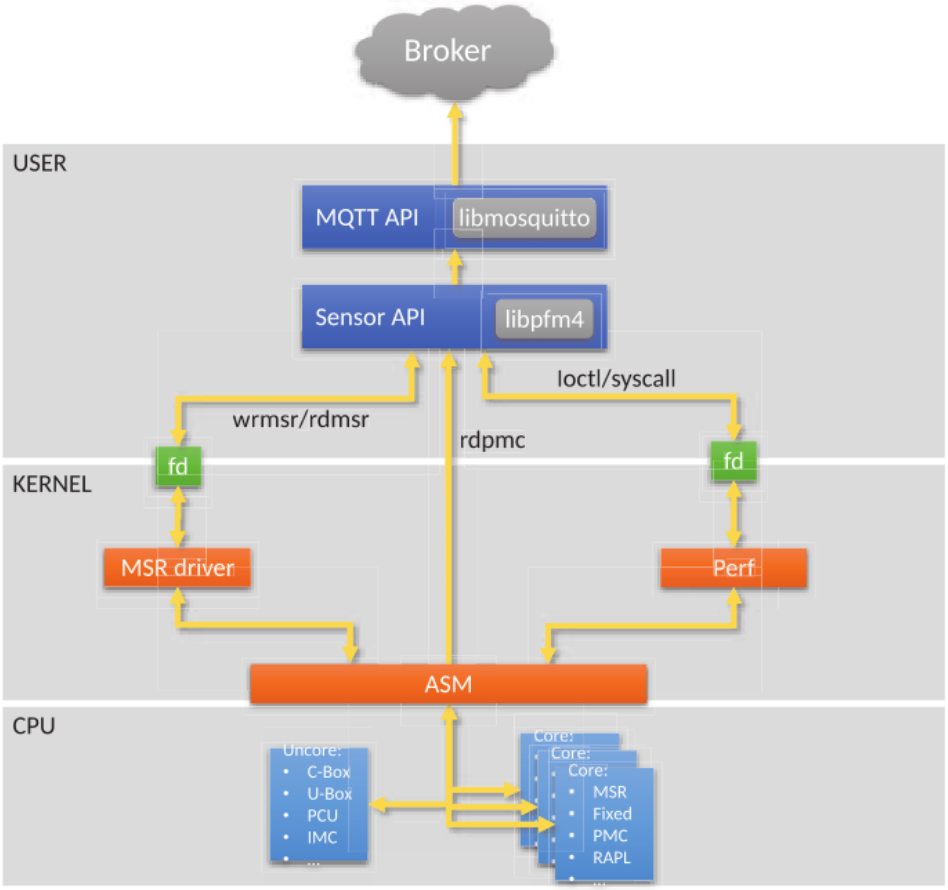


Figure 3.2 pmu_pub plugin structure

The pmu_pub plugins access to the data using three different modalities:

1. MSR driver, this modality exposes the MSR software interface through the kernel driver present in the Linux based OS;
2. rdpmc instruction, non-architectural instructions that can be used from the user space without the intervention of a system call or kernel driver, with an extremely low latency access to the performance counters;
3. Perf interface: a high level abstraction layer used mainly to hide the complexity of the processing managing units.

The MSR driver interface allows pmu_pub to collect and deliver temperature and energy sensor values.

3.2.5.2 Ipmi_pub

IPMI (Intelligent Platform Management Interface) is an interface created for the management and monitoring of computing resources in out-of-band mod; this standard is largely adopted by all the major HPC hardware vendors. IPMI interface relies mainly on the BMC (Baseboard Management Controller), an on board HW controller used to manage low-level board components exploiting embedded protocols. Several of these components are equipped with sensors for measuring temperatures, fans speed and power consumption that are collected by the BMC firmware and exposed on the IPMI interface.

EXAMON gathers this data through the ipmi_pub collector, creating a path from IPMI sensors to the MQTT data bus. Moreover, when the BMC firmware does not provide all the available sensors, the collector is able to directly communicate with the low level protocols using the bridge functionality offered by the IPMI ``raw'' commands and thus retrieve additional sensor data. The IPMI communication pattern is asynchronous and I/O bound; thus, the ipmi_pub collector is implemented as a multi-thread application that executes in a service node. Each thread maps a worker that can be employed to monitor one or more compute nodes.

The workers, in parallel and at regular intervals, connect to the IPMI network interface of the assigned node and issue the IPMI command to collect all sensors data. When the response arrives, the worker wakes up and the payload is decoded and translated to the MQTT packet. This scheme relies on the performance provided by the single instance of the IPMI driver present on the host OS. To increase scalability, the ipmi_pub collector can be replicated among multiple virtual machines (VMs), thus splitting the driver load among each VMs OS instance.

3.2.5.3 MQTT2KairosDB

EXAMON creates a common interface for heterogeneous data sources using a data format capable of abstracting them. However, this abstraction is not always possible, as some tools do not provide a native way to connect to the MQTT data bus -- for instance, KairosDB[24], which is used to store data by EXAMON. Data can be inserted in KairosDB using the provided REST API or via a low-lever TELNET mechanism (to allow higher ingestion rates). We implemented a middleware (MQTT2KairosDB) that acts as protocol bridge between the MQTT layer and the input interface of the DB. Its architecture follows the general plugin scheme but only the KairosDB API object of the worker had to be implemented.

3.2.5.4 Stream Processing

EXAMON integrates a low-level native stream processing mechanism, aimed at light-weight processing tasks on a set of MQTT streams (e.g, data filtering, threshold, and event-based alerting). Moreover, it is possible to create virtual sensors that produces high level data streams combining the low-level metrics. It implements an event-based streaming processing model where the basic data element is the (key, value, event_time) 3-tuple; in EXAMON the topic acts as the key while the timestamp corresponds to the event_time. In the general case where time series with different and irregular sampling rates, are processed, a synchronization buffer is used. The window defines which events will be inserted in the synchronization buffer (see Figure 3.3), that represents the data matrix that will be processed.

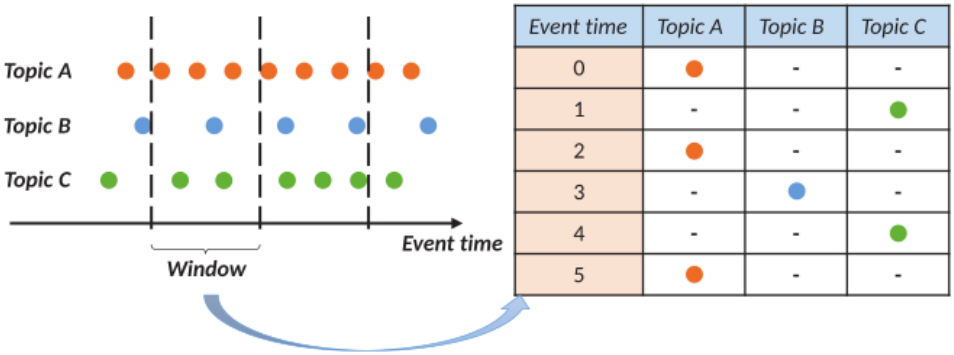


Figure 3.3 Synchronization Buffer

[24] https://github.com/kairosdb/kairosdb

The buffer is a time-indexed table where data can be reordered or uniformly re-sampled and the missing values can be filled with interpolated data. The processing stage is a user defined function that takes the buffer as input and produces the final results, published on the appropriate topics. For instance, we used this component to implement a streaming application consisting of a ``virtual'' sensor calculating, in real-time, CPU and DRAM power consumption using lower-level metrics reporting energy and CPU performance counters.

3.2.5.5 Job Information

This sub-module has two functions: 1) gather jobs related information from the job dispatcher and 2) aggregate physical measures on a coarser granularity. There are a set of job dispatcher extensions and python scripts that perform these tasks. The job dispatcher in a supercomputer is the software component that handles users' requests and decide when each job should start and which resources to allocate. We extended the job dispatchers employed in CINECA supercomputers (Slurm[25]) with the additional plugins to send information about job requests. The plugins collect data regarding the job request (e.g., job id and name, requested resources, etc) and the job execution (such as start and end times); then, the information is sent to the MQTT broker. There, a software daemon periodically retrieves the fine-grained measurements available on KairosDB and combine them with job-related info, storing the merged information in long-term Cassandra tables.

3.2.5.6 Node Status Information

In addition to data coming from physical sensors, EXAMON also collects information related to service monitoring and node status using Nagios[26]; by using Nagios and its alarm generators, system administrators are warned about potentially critical conditions, which are then manually verified – if the alarm was real, the involved computing nodes have to be removed from production (“drained”). When nodes are marked as not in production by system administrators their different state is registered in Nagios as well. The data sampling frequency is 15 minutes, as determined by Nagios monitoring functionalities. In most HPC systems service-reporting tools are decoupled from physical monitoring infrastructure, but thanks to EXAMON we can merge them. Nagios is composed by a core part which monitors and visualizes critical IT infrastructure components. It also provides alerts and historical logs of variations in the state of each monitored components. To monitor the state of a given SW and HW component Nagios can be extended with predefined and custom plugins, which reads metrics related from target components and based on specific rules defines its state as “Normal”, “Warning” and “Critical”. Only state transition events are recorded in the logs. These values are periodically sent as MQTT messages EXAMON to be stored as additional metrics. The critical observation is that the inspection done by system administrators through Nagios can be exploited to distinguish between nodes in normal states and nodes in critical conditions. This information can be framed as labels describing the status of the supercomputer, thus providing (for free) the automated annotation operation which is lacking in most modern HPC systems.

3.3 AI-based optimization

On top of EXAMON infrastructure a series of services based on artificial intelligence (AI) technology were developed, as part of the Task 5.2.2. These services were tailored following the guidelines proposed by the IoTwins WP3 – as described in D3.2. These services aim at improving the management of the supercomputer in diverse areas, ranging from failure identification and prediction (predictive maintenance) to thermal and

[25] https://slurm.schedmd.com/documentation.html
[26] https://www.nagios.org/

power prediction; the outcome of these preliminary service will be the creation of a digital twins for large-scale datacenters. The current version of the AI-based services hasn't been deployed yet, and they have been tested exclusively on data batches. Deployment, validation and testing on live-data will be part of the activities performed for Tasks 5.2.3 and 5.2.4.

3.3.1 Anomaly Detection

The first step towards predictive maintenance is to be able of *detecting anomalous states* – this is not a trivial issue in complex machines with thousands of parts such as supercomputers. On CINECA machines (from D.A.V.I.D.E. to Marconi100) we then decided to create Deep Learning (DL) models to automatically detect failures and anomalies. We focus on models working on single nodes; in future works, we will "combine" these single-node models to cope with the entire cluster (or sub-clusters). Our idea is to use a particular type of neural network called autoencoders to learn the normal behaviour of supercomputer nodes and then to use them to detect abnormal states. These networks are trained using only normal data, with no need for labelled anomalous classes at training time – hence this approach belongs to the *semi-supervised* area within DL. The autoencoder models were developed in Python language, using the library TensorFlow (version 2.1), and the corollary library Keras; these are the same technologies adopted by WP3 for the AI-based services – in fact, the autoencoders used in this instance are those presented in deliverable D3.2 but tailored for the TB06 context. We create a set of separate autoencoder models, one for each node in the system. Each model is trained to learn the normal behaviour of the corresponding node and to be activated if anomalous conditions are measured. If an autoencoder can learn the correlations between the set of measurements (features) that describe the state of a supercomputer, then it can consequently notice changes in these correlations that indicate an abnormal state. Under normal operating conditions these features are linked by specific relations -- these correlations will be perturbed if the system enters in an anomalous state. The reconstruction error is the element we use to detect anomalies. An autoencoder can be trained to minimize this error. In doing so, it learns the relationships among the features of the input set. If we feed a trained autoencoder with data not seen during the training phase, it should reproduce the new input with good fidelity, at least if the new data resemble the data used for the training. If this is not the case, the autoencoder cannot correctly reconstruct the input and the error will be greater. We detect anomalies by observing the magnitude of the reconstruction error.

All autoencoders have the same structure. We opted for a fairly simple structure composed by three layers: I) an input layer with as many neurons as the number of features (166), II) a densely connected intermediate sparse layer with 1660 neurons (ten times the number of features) with Rectified Linear Units (ReLu) as activation functions and a L1-norm regularizer, III) a final dense output layer with 166 neurons with linear activations. This network was obtained after an empirical evaluation, after having experimented with different topologies and parameter configurations. To summarize, our methodology has the following steps: 1) create an autoencoder for each computing node in the supercomputer; 2) train the autoencoders using data collected during normal operating conditions; 3) identify anomalies in new data using the reconstruction error obtained by the autoencoders.

The experimental results demonstrate that this approach has a good potential; the experiments were conducted on historical data set collected from CINECA machines. The results were partially presented in previous works[27]. The experiments were performed in an offline setup, not to hinder the supercomputer in production and to operate in a controlled setting. Several months of D.A.V.I.D.E. activities were collected and

[27] Borghesi et al., "Anomaly Detection using Autoencoders in High Performance Computing Systems", AAAI 2019
Borghesi et al., "A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems", Engineering Applications of Artificial Intelligence, 2019

used as the main data set. Most of the data corresponded to normal state of the system; on a subset of 20 nodes, we injected anomalies in a controlled way for time periods ranging from few hours to a couple of days. The anomalous time-periods were only fed to the model at test time. The injected anomalies were misconfiguration of the power governor deciding the clock frequency of the supercomputing nodes. The experimental results are very promising. On average the accuracy is pretty good with an accuracy higher than 90% over all twenty nodes; to be precise we obtained an average F-score (weighted over all classes) equal to 0.935, with maximum and minimum values equal to 0.999 and 0.828. In general, the accuracy classification for the anomaly class is significantly smaller than the normal class; this is due to the relatively higher number of false negatives compared to the number of false positives.

The visual inspection confirms the quantitative analysis. Let's look at the reconstruction errors for two supercomputing nodes, as plotted in Figures 3.4 and 3.5 The x-axis and y-axis show, respectively, the time and the normalized reconstruction error. The reconstruction error trend is plotted with a light blue line; the gaps in the line represent periods when the node was idle and that have been removed from the data set. We observe various anomalous periods (highlighted by colored lines along the x-axis; different colors indicate different types of anomalies). The reconstruction error is never exactly zero, but this is not our concern: our analysis does not rely on the absolute value of the error, but rather on the relative magnitude of the errors computed for different data sets. The reconstruction error is indeed greater when the nodes are in an anomalous state, as underlined by the higher values in the y-axis in the periods corresponding to anomalies. Hence, the autoencoder struggles to recreate the "faulty" input data set.

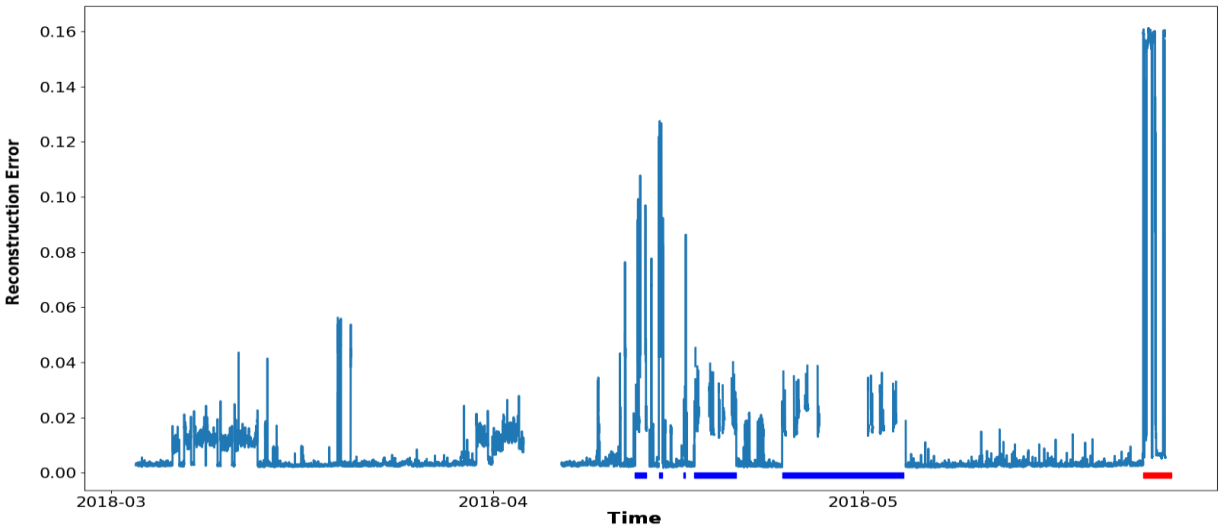


Figure 3.4 Node A

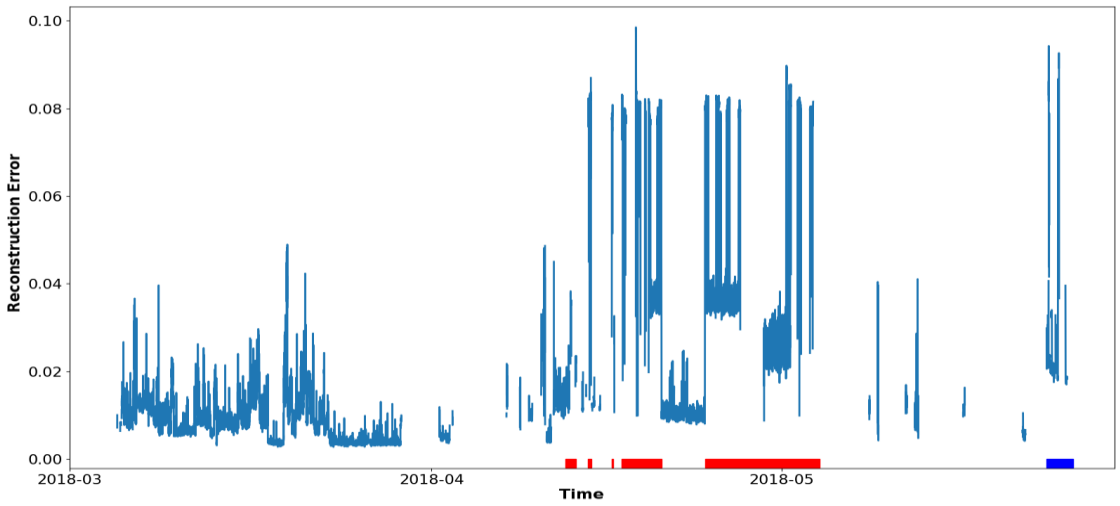


Figure 3.5 Node B

Anomaly cause identification

An additional benefit of the autoencoder network is the possibility to "interpret" the prediction of the model in a natural way. In particular, one can look at the different reconstruction errors obtained for each feature taken as input and output of the autoencoder. The reconstruction error of an autoencoder with multi-variate input and output can be obtained in different ways. The reconstruction error is the difference between the output and the input; this difference is computed feature-wise and then averaged. Using the aggregated reconstruction error is useful for obtaining a unique and coherent anomaly signal, but we can also observe the error generated in reconstructing each feature. Features with higher reconstruction errors have more anomalous behaviours, hence the autoencoder struggle to reconstruct them. These features can then be interpreted as the "root causes" of the failure, or at least a very strong indicator of which components of the systems are behaving in the most anomalous way. This can be observed by looking at the heatmaps portraying the feature-wise reconstruction errors along the temporal line, as shown in Figure 3.5 and 3.6 for two distinct nodes of Marconi (dubbed "Node C" and "Node D" for simplicity). The x-axis represents the time while they-axis shows the reconstruction errors for a subset of the features of the nodes. The heatmaps contain 4-hours period containing fault events on the nodes; the red markers on the top row indicate the anomalous periods. The magnitude of the reconstruction error is conveyed through the colour intensity, with hues close to white signifying low errors and hue close to intense blue representing high reconstruction error.

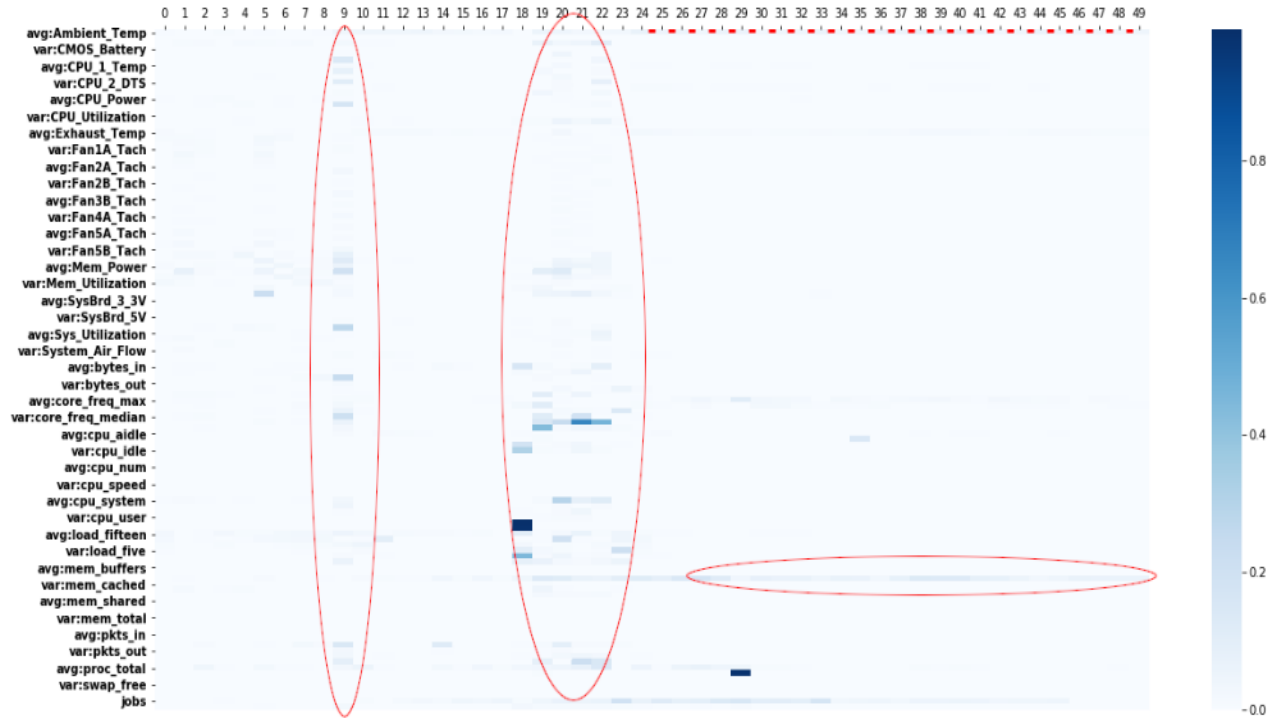


Figure 3.6 Node C

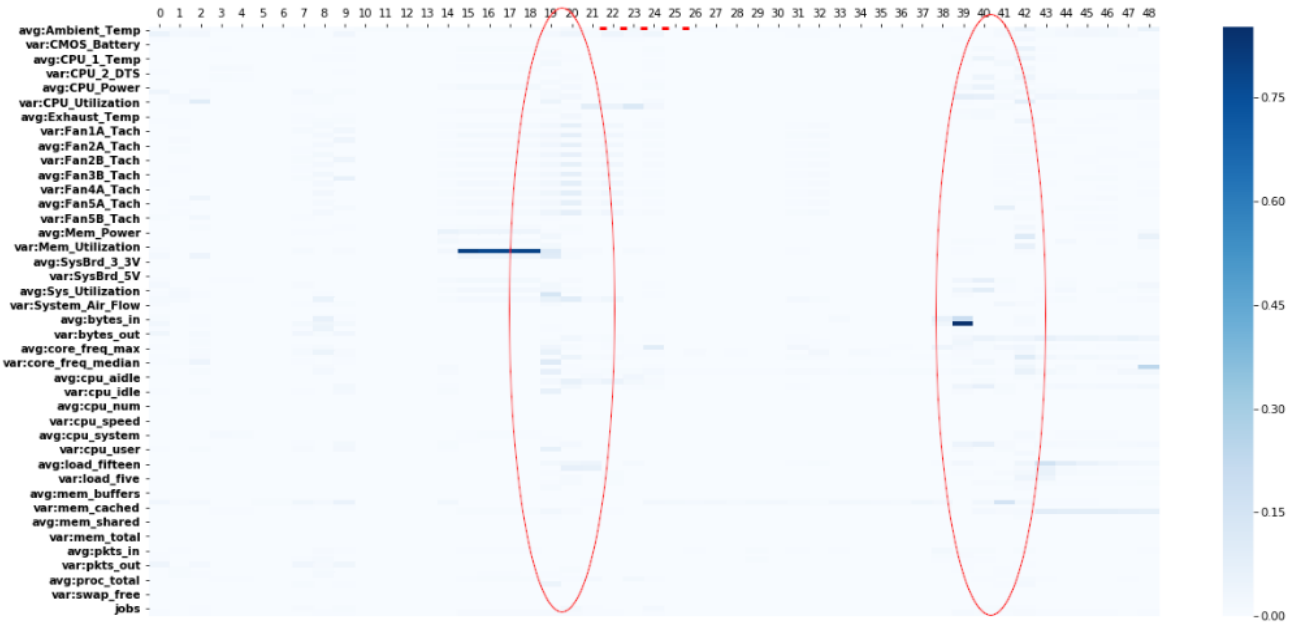


Figure 3.7 Node D

Let us start with Node C; we highlighted interesting areas of the map with red ellipses. The reconstruction error during the anomalous periods, which is higher for three specific features, “avg:mem_cached” (the amount of cached memory), “avg:DC_Energy” (the energy consumed by the entire node), and “jobs” (the number of jobs completed in the last five minutes) -- features enclosed by the three horizontally aligned ellipses. We can see that shortly before the anomaly a significant spike of the reconstruction error involves many features, in a marked contrast with the normal situation (the vertical ellipse). This strongly suggests that the semi-supervised autoencoder is aware that something strange is happening, or at least starkly different from the norm, hence it raises its anomaly signal, generating a false positive as the Nagios label is

still set to normal state. In the case of Node D we see similar patterns but also different ones. The heatmap displays a noticeable higher reconstruction error for many features before the anomaly insurgence, especially for features involving metrics which describe the cooling components (fan speed, air flux, etc).

3.3.2 Anomaly Prediction/Anticipation

After the detection of anomalies, we decided to evaluate whether failures can be predicted as well. To do so we exploit Nagios-provided labels to train a DL model for anomaly prediction/anticipation. One thing must be noted: a failure is recognized only when a system administrator notices the issue. This has two main implications:

1. There is an implicit delay between the insurgence of an anomalous situation and the corresponding label.
2. Labels indicating anomalies tend to be associated to nodes in idle state, as during the maintenance performed by system administrator to identify and fix the source of the issue no new jobs are submitted on the node.

This situation complicates the creation of an automated model for anomaly detection. We do not want a DL model that simply "learns" the correspondence between failure state and idleness. Concerning the evaluation of trained models, we cannot simply consider standard classification metrics such as the accuracy. For events identified as "failures" by a DL model *before the actual insurgence of the failure as registered by system administrators* might be a good sign, as these "false positives" indicate that the DL model is recognizing that something is off before a human operator could – hence **anticipating** the anomalous situations.

To cope with these issues, we decided not to rely on pure semi-supervised approaches (as the one described in Section 3.3.1), nor on entirely supervised methods, typically adopted in the literature but never applied to real production systems[28]. Hence, the core of the approach proposed in this paper is the union of two DL models, namely a 1) *semi-supervised autoencoder deep neural network* and 2) a *supervised neural network* composed by an autoencoder (distinct from the semi-supervised one) and a series of classification layers. The overall model is portrayed in Figure 3.8. Both models produce an anomaly signal, classifying data points as representing normal or faulty states of the supercomputing nodes. The predictions of the two models are combined to generate the final signal: if any model raises the anomaly flag, then the data point is classified as anomalous. The anomaly signals of the two models are thus merged in a logic-OR fashion. For the moment, we are training a different couple of models (supervised and semi-supervised) for each node in the supercomputer.

[28] Netti et al.,"A Machine Learning approach to online fault classification in HPC systems", Future Generation Computer Systems, 2019
Tuncer et al., "Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning", IEEE Transactions on Parallel and Distributed Systems, 2018

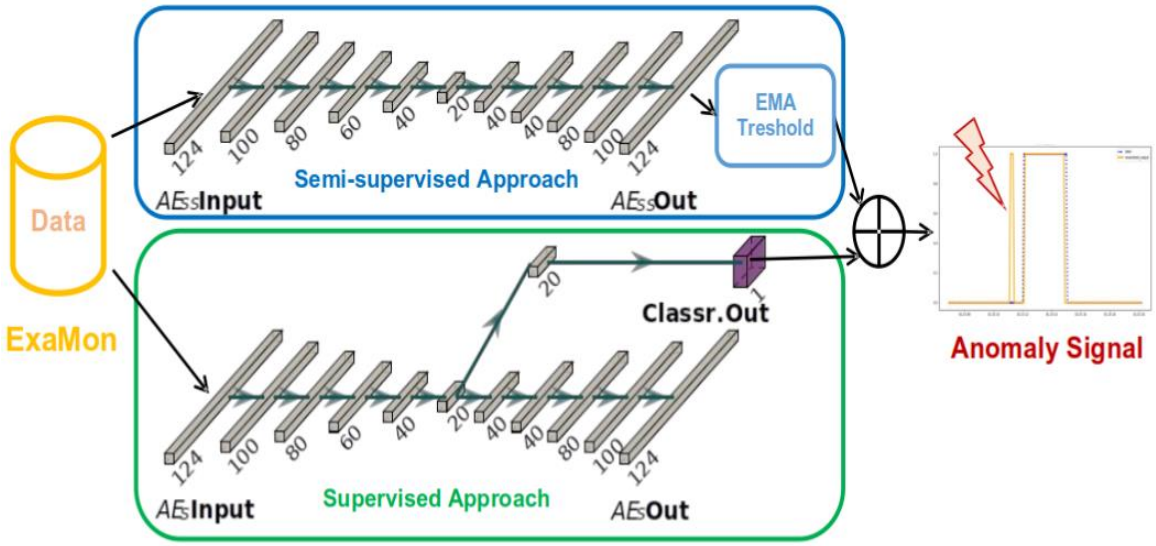


Figure 3.8 Combined approach scheme

All layers belonging to the autoencoders have ReLU activation function, the final classifier layer has a softmax as activation function. The autoencoder networks used for the semi-supervised approach and for the supervised one have the same topology (number of layers, neurons, etc) but are trained separately (using only normal data in the semi-supervised case, both normal and anomalous in the supervised one).

3.3.2.1 The semi-supervised Model

The semi-supervised model is composed of an autoencoder DNN trained *using only normal data*; in this way the autoencoder learns the correct behaviour of a node. Then, the network can be used for anomaly detection by observing its reconstruction error computed on new data: if it is greater than a threshold, the new sample is classified as anomalous, normal otherwise (high error means that the autoencoder does not "recognize" the data). The data fed to the autoencoder is pre-processed through a convolutional step. The convolution is used to "smooth" the raw data and to more explicitly consider the temporal dynamics involved in the computing nodes behaviour; we employed a convolutional 1-dimensional layer with a filter of a size corresponding to 2 hours.

The reconstruction error is computed as the average error over all the features (by construction, the autoencoder has the same number of features both in input and in output). The selection of the anomaly threshold is a non-trivial issue: we did not adopt a fixed one, but we employed Exponential Moving Average (EMA), obtaining a variable threshold which follows the error trend in previous time steps, giving more weight to more recent observations. A data point is classified as anomalous only if the reconstruction error is larger than the EMA threshold plus a Δ value, in order not to have a signal which overreacts to any non-significant oscillation. We compute the Δ as the average reconstruction error over the previous 25 minutes.

3.3.2.2 The supervised Model

The supervised approach consists of an undercomplete autoencoder network for feature extraction ("undercomplete'' means that the latent representation learned by the network has a lower dimension compared to the input features); this autoencoder is pre-trained in an unsupervised manner (minimizing the reconstruction error plus a regularization term), using all data points in the training set, including both normal and anomalous points. On top of the autoencoder two classification layers have been added; these latter layers are trained using the labels (provided by Nagios) by minimizing the categorical cross-entropy. The training happens in two phases: first, the unsupervised autoencoder is trained, then its weights are fixed and only the encoder is used to train the classifier layers in the supervised manner. At inference time, the anomaly

signal is generated by feeding new data to the encoder, obtaining a compressed representation, and then the latent representation is passed to the classifier, which provides a real number as output (i.e., the probability of belonging either to normal or faulty class); if the output is higher than a threshold the point is classified as failure – we employ a threshold equal to 0.2.

3.3.2.3 The results

The combined approach has been tested on two months of actual production on the Marconi supercomputer, over thirty different nodes (to ensure the statistical validity of the performed analysis). The results on the historical data sets are very promising, as Figure 3.9 reveals. The meanings of the columns are the following: TP, True Positives; TN, True Negatives; FN, False Negatives; FP-Pre, False Positives in the period preceding the Nagios-annotated label (two hours earlier); FP-Post, False Positives in the two hours following the cessation of the failure according to Nagios; FP-Rnd, False Positives randomly picked over the whole test set; FP, total number of False Positives; TPR, True Positive Rate (also known as recall); Prec., Precision; FNR, False Negative Rate; F-score is the harmonic mean of precision and recall -- values close to one indicate greater detection accuracy. The table reports the average results computed over all nodes used for testing. The month indicate the time-period corresponding to the test data – two months were selected, January and May 2020. α is a value which characterize the sensitivity of the EMA threshold; α weighs the importance of more recent time steps, higher values discount older observations faster. The last column of the table reports the anticipation period with respect to the insurgency of the anomalies. If there are at least three consecutive false positives before a Nagios-labelled failure, we assume that the DL model is *predicting* the anomalous state, detecting that something in the computing node is not behaving properly *before the system administrators*. We calculate the anticipation period (reported in minutes) by measuring the difference between the time stamp when the failure is labelled as such (via Nagios) and the time stamp of the first false positive in the series of consecutive ones leading to the anomaly.

Month	α	TP	TN	FN	FP-Pre	FP-Post	FP-Rnd	FP	TPR	Prec.	FNR	F-score	Anticipation
Jan. 2020	0.3	19.70	5652.60	2.80	4.30	4.70	17.80	26.80	0.88	0.42	0.12	0.57	42 min
Jan. 2020	0.5	19.70	5664.10	2.80	2.70	4.70	7.10	14.50	0.88	0.58	0.12	0.69	33 min
Jan. 2020	0.7	19.70	5668.50	2.80	1.90	4.70	3.50	10.10	0.88	0.66	0.12	0.75	33 min
Jan. 2020	0.9	19.70	5669.50	2.80	1.50	4.70	2.90	9.10	0.88	0.68	0.12	0.77	33 min
May 2020	0.3	186.40	6901.00	8.30	10.60	9.70	63.10	83.40	0.96	0.69	0.04	0.80	58 min
May 2020	0.5	186.40	6923.60	8.30	10.20	9.60	40.80	60.60	0.96	0.75	0.04	0.84	57 min
May 2020	0.7	186.40	6930.10	8.30	10.20	9.60	34.40	54.20	0.96	0.77	0.04	0.86	57 min
May 2020	0.9	186.40	6932.20	8.30	10.00	9.60	32.50	52.10	0.96	0.78	0.04	0.86	57 min
Jan. & May	0.3	206.10	12553.60	11.10	14.90	14.40	80.90	110.20	0.95	0.65	0.05	0.77	50 min
Jan. & May	0.5	206.10	12587.70	11.10	12.90	14.30	47.90	75.10	0.95	0.73	0.05	0.83	45 min
Jan. & May	0.7	206.10	12598.60	11.10	12.10	14.30	37.90	64.30	0.95	0.76	0.05	0.85	45 min
Jan. & May	0.9	206.10	12601.70	11.10	11.50	14.30	35.40	61.20	0.95	0.77	0.05	0.85	45 min

Figure 3.9 Comparison of the experimental results obtained by combining the supervised and the semi-supervised approaches; average results over all nodes; the α value refers to the EMA computation.

The experimental results are really promising. Indeed, it is possible to anticipate the Nagios-annotated labels by a significant amount of time, roughly around 45 minutes when we compute the average over both January and May. The accuracy of the combined method is slightly inferior (or almost equal) compared to the semi-supervised one and worse than the supervised approach. This is not an issue, as this is primarily due to the higher number of false positives detected before the label. This is a feature of the model we were aiming at: we are not just interested in a classification method, but we want to demonstrate the capability to anticipate anomalies -- and for this reason a lower detection accuracy is a reasonable price to pay.

The quantitative analysis is also backed up by the graphical representation of the anomaly signal. For instance, Figure 3.10 portrays the anomaly signal for a node in Marconi; on the x-axis there is the time, while the y-axis represents the anomaly signal – a value equal to zero indicates normal behaviour (the model thinks that the supercomputing node is behaving correctly), while a value equal to one indicates a failure condition. The yellow solid line represents the anomaly signal generated by the combined DL model ("ensembled_output") and the dotted blue line represent the actual label registered by Nagios. It can be easily seen that the anomaly signal closely follows the target, that is the true label, and the time period with the node in failing state is correctly recognized. Moreover, the interesting part is that the anomaly signal starts to be "uncertain" well before the true label, as noted in the area highlighted with the red rectangle. This is a situation where the DL model is able to anticipate the insurgency of the anomaly as detected by system administrators; it is counted as a false positive, but in this case, the model is "wrong" for the right reason (e.g., there is an unavoidable delay in the Nagios-provided labels).

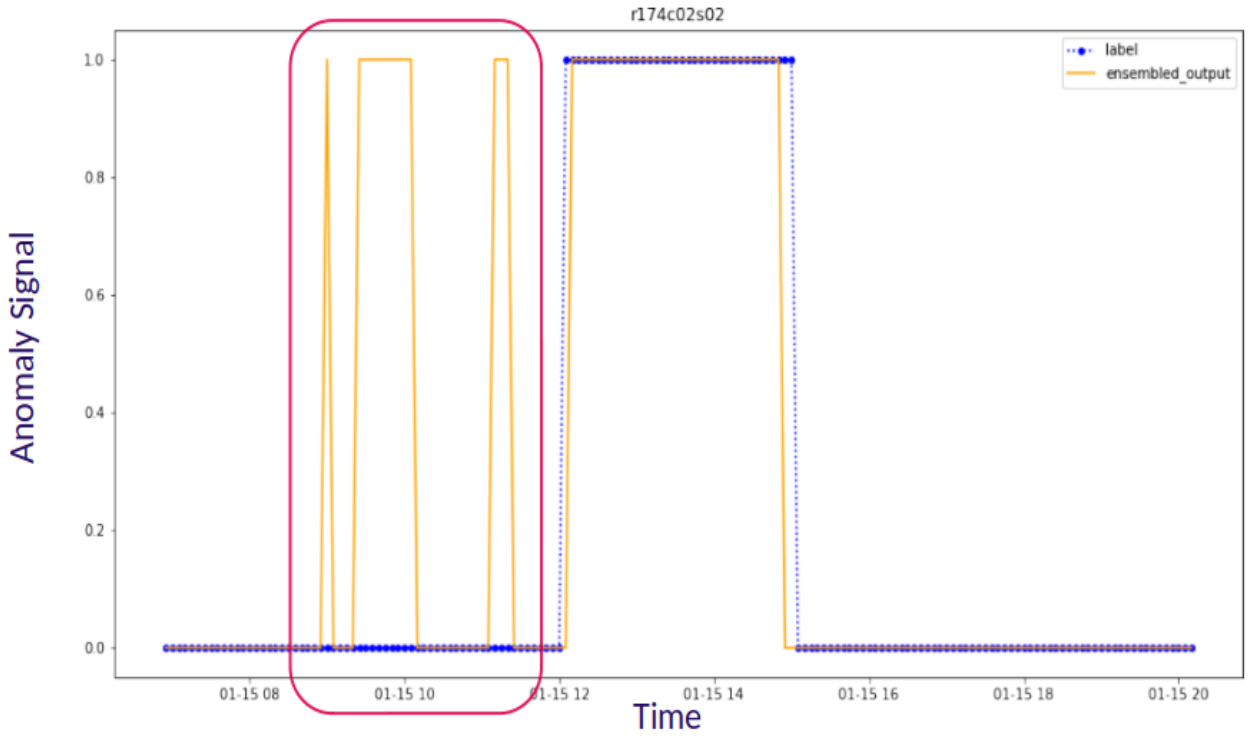


Figure 3.10 Anomaly anticipation – Node E

3.3.3 Job Power Prediction

A serious problem in large-scale data centers is the power and energy consumption. There exist several techniques to reduce energy consumption in HPC systems, both using hardware-based (e.g., power capping) or software-based (e.g., job scheduling) solutions. However, a limiting factor towards the adoption of energy-saving techniques is the increased difficulties in terms of accounting and pricing mechanism. To cope with this problem, Machine Learning models can be used to predict the power consumption of HPC applications. Using the data collected by EXAMON infrastructure, we can combine information coming from HW sensors and data from the job dispatcher it is possible to build ML models for estimating the power consumption of a job *before* its actual execution.

We have collected the data generated in 12 months of in-production usage. This data has then been used to train ML models and to test their predictive capabilities. For every completed job EXAMON stores information available at submission time, such as job user, account, requested duration and HW resources, etc.; in

addition, the metrics collected during the lifetime of the job on the execution nodes were also collected and stored in an aggregated fashion (by taking the mean value over the whole job duration) -- thanks to this aggregated data it was possible to know the average power consumption of each completed job. We processed this data to create suitable data set and to normalize it; afterwards, it was used to train and test a ML model for predicting the average power consumption. In this case we opted for a Random Forest regressor, as previous works in the HPC settings revealed it to be a good choice, in terms of model's accuracy and ease of training[29]. For the moment, both training and inference/validation were performed on batches of historical data.

To evaluate the power prediction model, we measure the accuracy of the power estimation by considering the prediction error. The Absolute Percentage Error is computed as (predicted value – real value)/ divided by (real value * 100). The Mean Absolute Percentage Error (the average value computed considering all jobs in the test set), or MAPE, is equal to 7.35%, quite a good value. Figure 3.11 plots the histogram of the estimation errors. The x-axis specifies the percentage error (not the absolute value) while in the y-axis there is the number jobs whose power was predicted with that error magnitude. The histogram agrees with the results of the numerical analysis, with most jobs having a percentage error smaller than 5%.

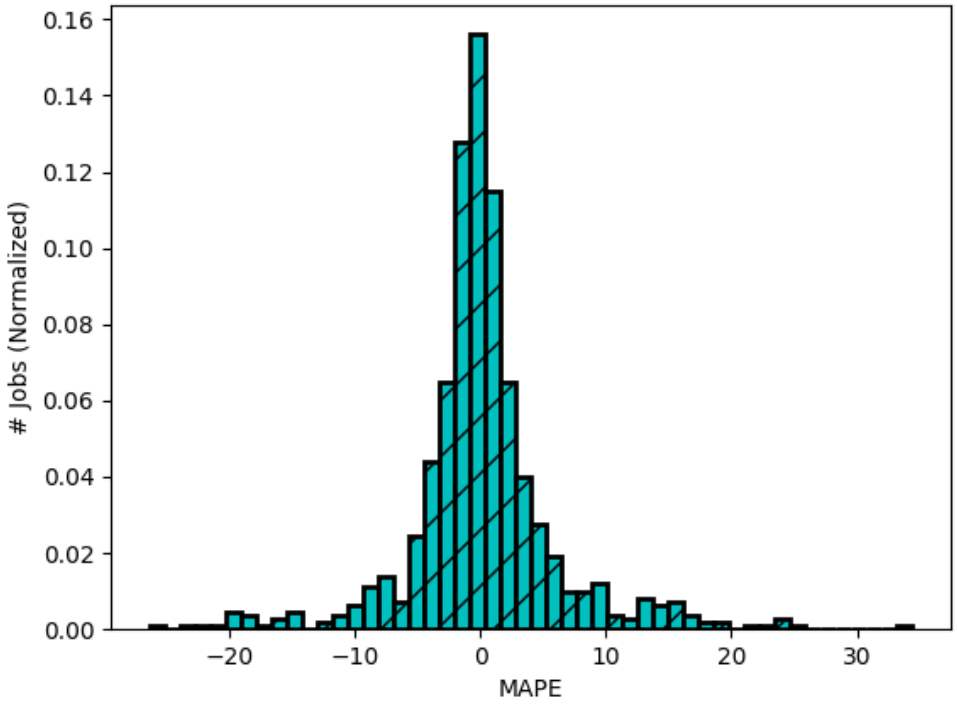


Figure 3.11 Job power prediction use case: errors histogram.

3.3.4 Job Failure Analysis

Another key issue for operators of HPC systems is understanding why jobs submitted by users fail. This is a constant concern as the number of failing jobs is a critical measure of the quality-of-service provided by the data center. In this context, we are still studying the possibility of using DL models to understand why HPC jobs do not conclude in correct states, and whether it is possible to forecast their failures. At this stage, we haven't reached conclusive results, yet, as the problems is not simple.

[29] Borghesi et al., "Predictive modeling for job power consumption in hpc systems", International conference on high performance computing, 2016

However, we have devised a useful indicator for system accountants, based on the observation of the failure rates of HPC applications as registered by EXAMON. The first observation is that system accountants need to understand the reason behind sudden spikes in the number of failing jobs, as these spikes might indicate underlying problems in the machines. However, this is not necessarily the case, as steep increase in the number of failing jobs can be due to improperly created batches of applications. To help system accountants discern these situations, we devised a metric based on the *entropy* of the failing jobs with respect to the users submitting them. We group failing jobs by user entropy: if in a certain time period, the failing jobs belong to a varied and large pool of different users, then the entropy measure for that time period is high; conversely, a low entropy value indicates that many jobs belonging to the same user (or a limited set of users) are failing at the same time. It is very probable that moments with low entropy correspond to batches of wrong jobs (e.g., a bug in the code, wrong specification of the job submitted to the supercomputer, attempt to access resources unavailable for the specific account, etc.); instead, job failures spikes with high entropy should be cause of concern for system administrators, as they underlie system-wide issues (e.g., a problem with the internal network).

Figure 3.12 displays the entropy indicator for the whole Marconi supercomputer in the period ranging from August 2019 to January 2020. The time is on the x-axis while the y-axis reports the entropy value; blue hues indicate low entropy; red hues indicate high entropy. The image on the bottom is a zoom-in of the first one, excluding exceptionally high job failure spikes. It can be seen that most of the failure spikes tend to be due to very few users (deep blue points), while relatively fewer (albeit not none) seem to depend on systemic issues, as highlighted by the lower number of bright red spots.

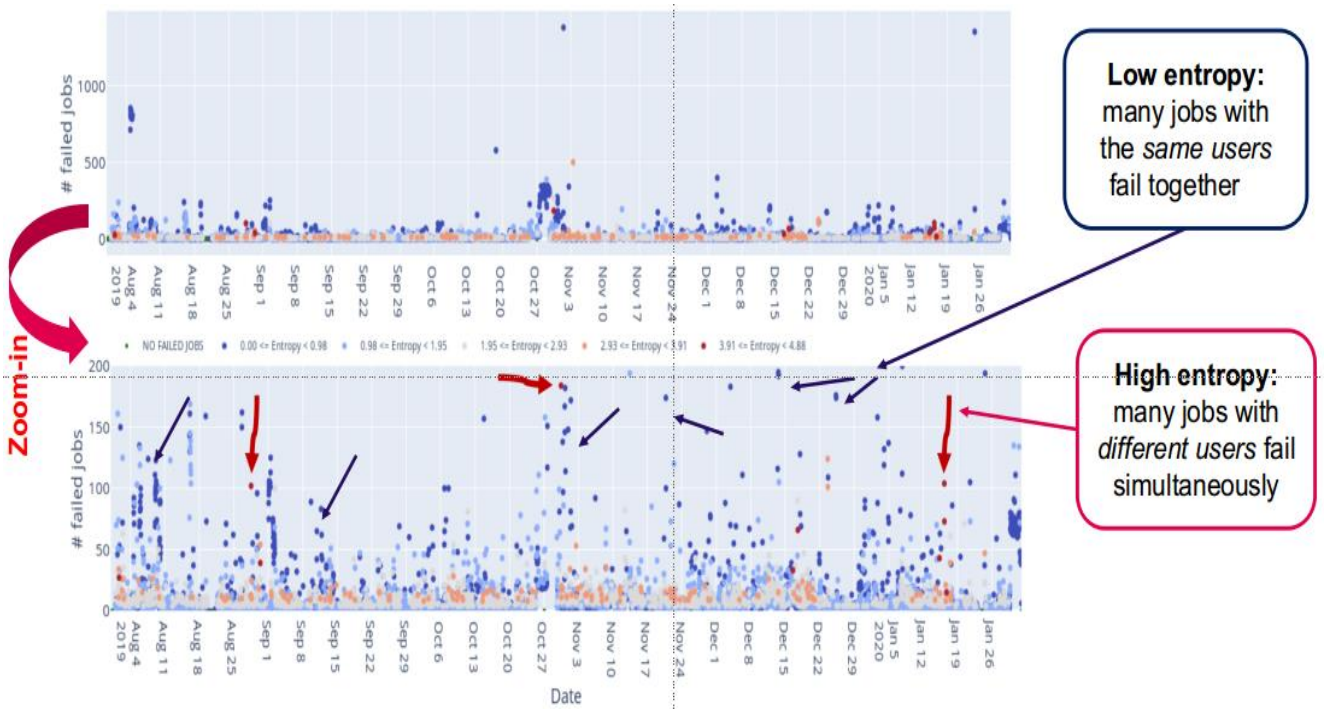


Figure 3.12 Job failures entropy-based indicator.

4 Testbed N7 - Smart grid - power quality management

4.1 General description of the system

IoTwins testbed 7 “Smart grid – power quality management” focuses on the detection of smart grid events - including faults, errors, and benign events - and estimations of missing data as close as possible to the data sources with input from higher levels for information that cannot be accessed locally (see use-cases described below). In smart grids, different information is available at different voltage levels – some is needed locally, some globally, or on the level of sub-systems. Data must be pre-processed locally in the context of the overall system state due to real-time requirements, high data volumes for selected types of sensors, and the spatial distribution of the system over a large area. In testbed 7 the IoTwins architecture is deployed to support the following use-cases by means of digital twins and Machine Learning models:

- Sensorless Control – Measurement Estimation
  In this use-case control and monitoring of network parts is enabled without installing additional sensors into the power grid. This can be realized by using smart meter measurements (which are received time-delayed) at the Cloud level to train Machine Learning models to estimate the measurements based on data that is available online (like the voltage and measurements at the transformer or the current weather data). Thus, even though the data from the smart meters is received (at Cloud level) with a time delay of 1 day and might not have the desired resolution, the Machine Learning models can be trained at the Cloud level to be used at the Edge level to calculate the estimated measurements at that point in the power grid.
- State detection
  The technologies that are added to the smart grid are added with a purpose, e.g., a battery and a peak shaving controller are added to keep the power consumption at a given point in the power network in a predefined range. The states of these systems must therefore be measured/derived and monitored. This use-case comprises the training and building of state and quality detection models on the Cloud level while the state monitoring can be done on the Cloud level and on the Edge level, so that data is only transferred in case of a state change.
- Anomaly detection and Root Cause Analysis
  For an optimal performance of the smart grid, events, errors, and faults must be detected and distinguished from each other in a fast and reliable way. Especially the distinction of events from faults and errors is important, since the user must be informed about every incident but should not be overloaded with unimportant events so that the important error or faults are overlooked.
  This use-case is closely related to the state detection use-case. For example, the monitoring of the performance of a battery system might also include the detection of battery maintenance events. However, event detection must be performed on streaming data so that the event can be detected while it is happening.

The use-cases are implemented using the algorithms that were developed in IoTwins task 3.3 and in 5.3 and the software architecture that was outlined in deliverable D2.2.

In the following subsections first the testbed itself is described (sub-section 4.2). Then the properties of the architecture implementation are described (sub-section 4.3). Then the functionality that is deployed in testbed 7 to detect events and estimate missing data is described (sub-section 4.4). This section closes with a description of how the root causes of smart grid events are derived in testbed 7 (sub-section 4.5)

4.2 Aspern Seestadt

4.2.1 Real Aspern Seestadt

The transformation of traditional passive distribution low voltage power grids towards smart grids in which most of the devices and participants are actively managed poses significant challenges. In addition, these grids must be able to cope with voltage fluctuations caused by the increasing decentralized generation of renewable energies, modern storage options and entirely new applications, such as e-mobility. The following objectives must therefore be supported: maximum efficiency, absolute supply security, and increasing the potential for environmental protection and reduced CO_2 consumption considering climate change.

The smart grids of the future will connect all players in the energy system via communications networks, thus enabling timely, bidirectional, and cost-efficient communications between grid components, producers, storage facilities and consumers. To achieve this, the Aspern Smart City Research GmbH & Co KG (ASCR)[30] is conducting research in the development areas Aspern Seestadt in Vienna, Austria on 12 grid stations, 24 transformers of various types and five grid storage systems.

Once the sensor technology has been expanded, the use of software will make it possible to record and monitor factors that were previously hidden, such as grid capacity utilization. The precise network data allows the infrastructure to be used closer to its physical limits and an early warning signal is sent in the event of an impending boundary breach – no active intervention is necessary. This is especially significant considering increasing e-mobility and decentralized energy generation.

The next step is an active intervention in the grid infrastructure. Increasing automation means that the grid storage systems can be switched over and activated, and buildings can be actively used as energy storage systems. The smart grid can also shut buildings and photovoltaic systems off from the grid during periods of overcapacity.

4.2.2 Virtual Aspern Seestadt

A virtual version of the testbed Aspern Seestadt, based on the multi-modal simulation tool Bifrost[31], was introduced to compensate for restricted access to the real Aspern testbed because of the Covid-19 situation. In contrast to the real testbed, parameters like battery capacity, number of concurrently charging vehicles etc. can be changed freely, and operator actions as well as interrupting events can be performed without interfering with the end-customer. This also supports verification and testing of the deployed architecture.

Bifrost is a persistent, shared design tool and simulation environment for Smart Cities, with a strong focus on power grid infrastructure. Backed by a reactive server backend and powerful simulation engine, a browser-based, 2.5D user interface empowers researchers, network operators and planning experts to construct and analyse situations revolving around power grid operations. The internal engine state representing the simulation world, including all physical dynamics and structures, is fully exposed to external applications, such as control algorithms or time series analysis tools. Thus, Bifrost can be employed as a virtual testbed for Smart Energy System installations, allowing for the evaluation of scenarios which would be hard or impossible to stage in-field.

Bifrost is designed as a single-instance, concurrent multi-user environment, which means that clients can edit and view its structure and manipulate the simulation of the same settlement in real-time. The single source

[30] https://www.ascr.at/
[31] https://bifrost.siemens.com

of truth for a settlement is its representation in the state. A theoretically unlimited number of settlements can be maintained in parallel in the state, each outfitted with its own simulation control.

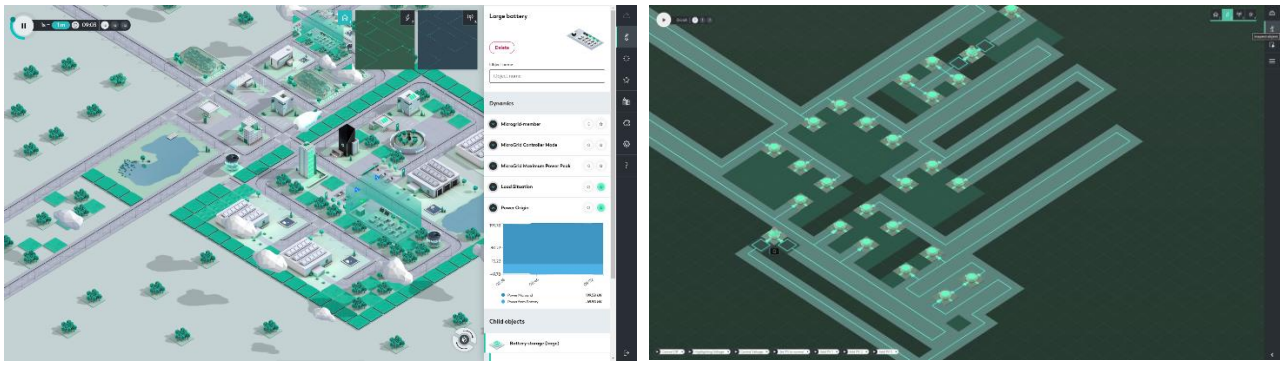


Figure 4.1: Screenshots of the multi modal simulation tool Bifrost which is used to create a Virtual Aspern Seestadt Testbed (VAST).

Internally, the state is maintained as a plain JSON object. The Bifrost engine uses Redux[32] as a modelling/control flow paradigm/library. Redux mandates that the state be immutable. The control flow is unidirectional: actions (themselves plain JSON objects) introduce changes to the state via reducers, which return a novel state object manifesting the requested changes. In Bifrost, actions can come from a variety of sources, including the simulation (e.g., to update the current in-world time), the browser client (e.g., to build a new structure) or external software (e.g., to affect the position of a power switch).

Figure 4.1 shows how Bifrost and a set of external modules is used to create the Virtual Aspern Seestadt Testbed (VAST). The data that is created by the power grid simulation is thereby transferred to the Edge devices via a Bifrost module that simulates a SEM3[33] device which provides the measurement values via Modbus. Thus, the Edge devices can be integrated into VAST as a Hardware in the Loop (HIL) simulation as they are integrated in the Real Aspern Seestadt Testbed (RAST) without the need of adapting the software.

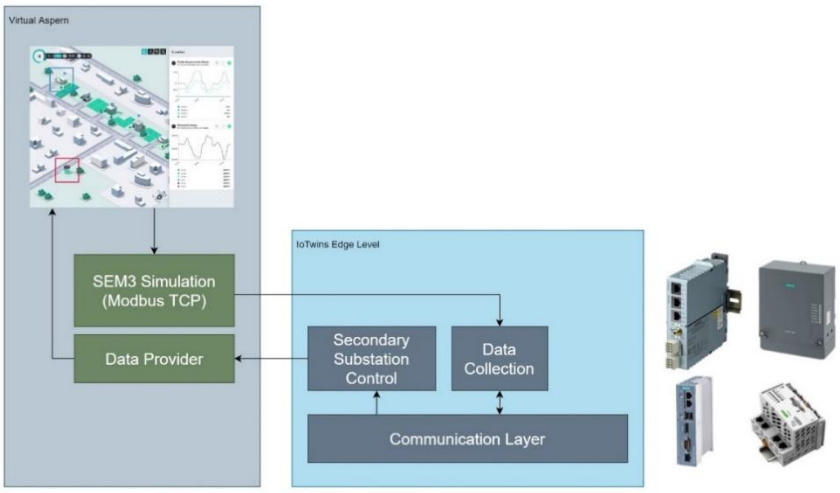


Figure 4.2: Integration of diverse Hardware into the Virtual Aspern Seestadt Testbed. The software stack at Edge and Cloud Level is the same as in the Real Aspern Seestadt Testbed.

[32] https://redux.js.org/
[33] https://www.siemens.com/download?BTLV_43061

4.3 Software and Hardware Architecture

The use-cases that were described in the previous section were implemented using different instantiations of the architecture that was proposed in deliverable D2.2. This section describes the basic software stack that was used to implement the architecture itself. For a description on how the use-case specific functionalities were implemented using this stack, see the following sections. This also includes specifics of the setup into which the IoTwins Architecture is integrated, since some of the functionality is to be integrated into an already existing hardware- and software environment. This provides the opportunity to not only evaluate the IoTwins architecture for usage in completely new setups but also to evaluate its flexibility when it comes to migrating existing solutions towards the IoTwins architecture.

Figure 4.3 and Figure 4.4 show the Cloud level and the Edge level architecture based on the IoTwins reference architecture described in IoTwins deliverable D2.2. In the next subsections the following software components and their usage in the testbed are described:

- Management of functionality:
  Wherever possible the used software components are deployed as containerized applications (following the OCI standard[34]). These applications are stored in a central repository. Thereby also the Machine Learning models are stored in containers so that only one storage is needed, and the same management components can be used for managing the applications and the models (see sub-section 4.3.1).
- Service Orchestration:
  For testbed 7 the central aspect of the IoTwins architecture is the integrated management of services at the Cloud level and the Edge level. To implement the central components for this management the service orchestrator Kubernetes[35] and the KubeEdge[36] solution (as described in deliverable D2.2 sub-section 2.4.1) are used. This software stack supports flexible setups for the different use-cases and the development environment (see the following section). The Kubernetes systems can be integrated with the OpenID Connect based INDIGO-IAM service that is provided by WP2 (see also deliverable D2.2 sub-section 2.4.5).
- Communication:
  For the communication between sensor devices and the Edge level, Modbus and IEC61850 protocols are used, while communication between the Edge level and the Cloud level is either based on OPC-UA PubSub or MQTT. On the Cloud level and the Edge level nats.io and MQTT are used for internal communication (see IoTwins deliverable D2.2 sub-section 3.3).

[34] https://opencontainers.org/
[35] https://kubernetes.io/
[36] https://kubeedge.io/

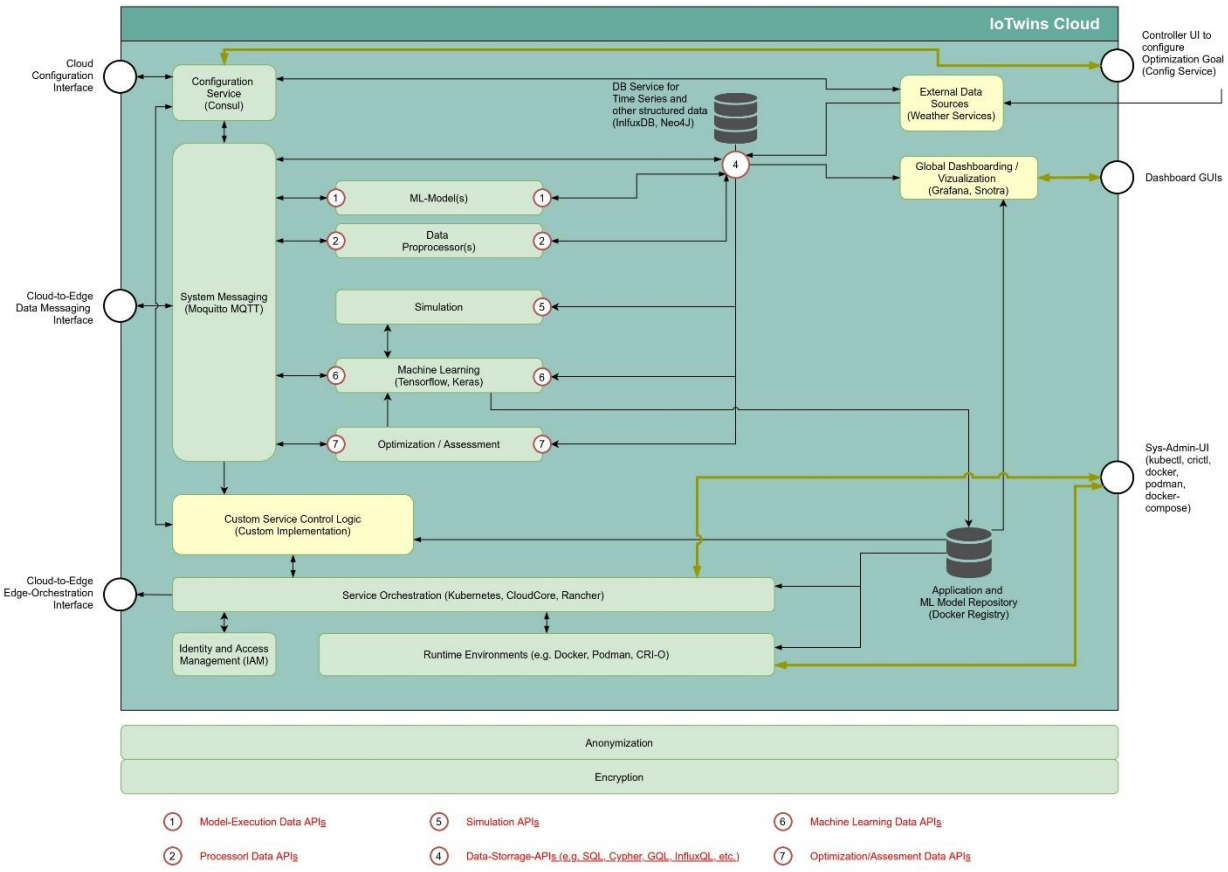


Figure 4.3: Cloud level architecture used in testbed 7, based on Figure 10 in IoTwins deliverable D2.2.

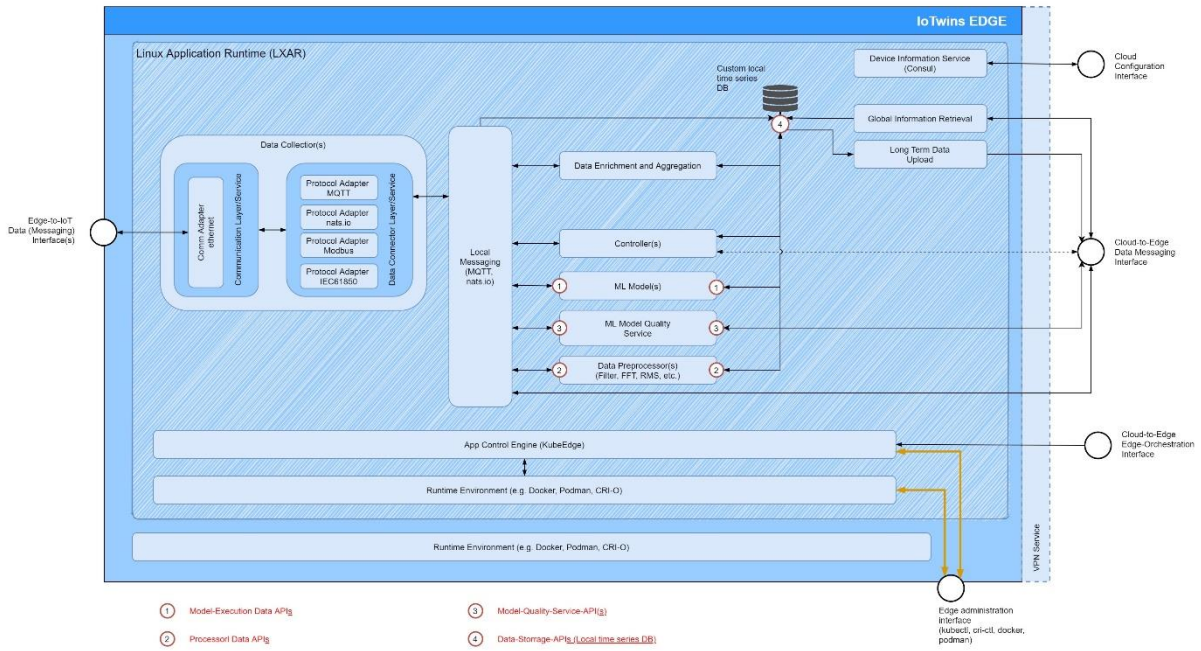


Figure 4.4: Edge level architecture in testbed 7, based on Figure 9 in IoTwins deliverable D2.2. As an extra security measurement, the data services as well as the orchestration services are running inside of a container (the LinuX Application Runtime LXAR).

4.3.1 Containerization of applications and ML models

The services at both the Cloud level as well as at the Edge level are managed using OCI compatible software containers (see IoTwins deliverable D2.2 subsection 2.4.1). This includes pure software components like applications that collect data from different data sources or filter this data as well as applications that contain Machine Learning models.

There are two options when it come to the deployment of trained Machine Learning model to both the Cloud and the Edge:

- Deployment via the data channel:
  In this case the container contains the application and all the libraries needed for running the application. The Machine Learning model is transferred to the application via a separate channel, for example via a REST interface, from a central model store.
- Deployment via the application channel:
  In this case the container contains the application, the libraries, and the model as a top layer in the container's file system (see Figure 4.5). Thus, when a new model is trained and is to be rolled out, a new container image is built, whereby only the top layer is updated and deployed via the standard application channel (which then only downloads the top layer).

The deployment via the data channel has the advantage that switching between two models is very fast and does not need restarting the application, but all applications must then implement this mechanism for updating and hot switch over and the tasks of managing the software and processing the data are mixed in the same application (which violates the principle "separation of concerns"[37]).

By using the deployment via the application channel, the hot standby and switch over mechanisms of the container manager and the orchestrator can be used to speed up switching between applications. In all cases the layers in the containers are to be planned carefully to only update the necessary components.

For testbed 7 the model deployment via the application channel was chosen. A central image registry thereby provides the images for starting and managing both the services and models. By providing base images (those images to which the model is added) for different hardware platforms (e.g., AM32v7, AMD64, and ARM64) it becomes possible to exchange hardware on the fly without the need to change any configuration.

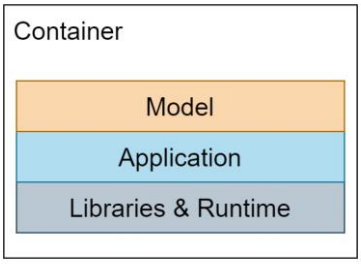


Figure 4.5: Structure of a container that contains the basic Libraries and Runtime components on a base level, the Application that manages the input and output and the Machine Learning Model on the top layer.

[37] https://en.wikipedia.org/wiki/Separation_of_concerns

4.3.2 Scalability of the Solution at Cloud Level and at Edge Level

Scalability is a core feature for testbed 7 for both the data services and for the orchestration of these services. The Kubernetes open-source system for automating deployment, scaling, and management of containerized applications was used in testbed 7 to implement the orchestration at Cloud level and at Edge level in accordance with IoTwins deliverable D2.2 (sub-section 2.4.1). Thereby the Edge devices are attached to the backend using the KubeEdge solution which supports also resource restricted devices that are not always connected to the backend (see also IoTwins deliverable D2.2). The Kubernetes solution provides the possibility to run the backend using different implementations and setups while the API and thus the provided functionality stays unchanged. Thus, the backend can be realized using at least these options (see also Figure 4.6):

- In-container (on-premise)
  The Rancher Kubernetes backend packs all central Kubernetes components (including the Rancher GUI) into a container that can be started using the simple docker compose file show in Listing 1. This solution can be used for managing single nodes on-premise or in the development and test environment (see Figure 4.6 (a)).
- Single Dedicated Server (on-premise and cloud)
  A Kubernetes backend can be set up on a dedicated Server (VM or bare metal) by installing the container manager (e.g. CRI-O[38]) and Kubelet[39] as well as the management tools Kubeadm[40] and Kubectl[41]. In this setup all central services like the API server and the cluster storage are running on the same central server (see Figure 4.6 (b)).
- High Availability Servers (on-premise and cloud)
  In case the central Kubernetes components are hosted on a single node (see "Single Dedicated Server") the failure of this server would lead to a service outage. Therefore, the central services themselves can be managed by the cluster itself and moved to spare node in case of a failure available[42] (see Figure 4.6 (c)). In this setup a separate solution for moving the public IP address of the CloudCore (part of KubeEdge) is needed. This can be realized using the keepalive tool[43].
- Dedicated Services (cloud)
  The big Cloud providers provide managed Kubernetes solutions and provide solutions for easily setting up and extending the clusters. Some examples of these managed Kubernetes services are:
    - Amazon Elastic Kubernetes Services (EKS): https://aws.amazon.com/eks
    - Microsoft Azure Kubernetes Service (AKS): https://azure.microsoft.com/en-us/services/kubernetes-service
    - Google Kubernetes Engine (GKE): https://cloud.google.com/kubernetes-engine
    - Digitalocean Kubernetes: https://www.digitalocean.com/products/kubernetes/
    - Ubuntu Managed Kubernetes: https://ubuntu.com/kubernetes/managed

[38] https://cri-o.io
[39] https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
[40] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/
[41] https://kubernetes.io/docs/tasks/tools/
[42] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/ha-topology/
[43] https://github.com/kubeedge/kubeedge/tree/master/build/cloud/ha#the-ha-of-cloudcoredeployed-in-k8s-cluster

In IoTwins for testbed 7 the in-container solution is used for the test- and integration-environment, the single server instance is currently used for the Real Aspern Seestadt Testbed and the Amazon EKS service is used to deploy Machine Learning models in the Virtual Aspern Seestadt Testbed.

This way, it was shown that the IoTwins architecture and its implementation scales up to the needs in testbed 7 and can be adopted to the customer's needs and preferences.

As in case of the Kubernetes solution there are different options to implement the image registry in which the container images for the services and models are stored. For testing purposes, a simple registry can be started inside of a container (see Listing 2) while a high availability solution like JFrog Container Registry[44] can be used for production environments.

```
version: '3'
services:
  rancher:
    image: rancher/rancher:latest
    container_name: "rancher"
    privileged: true
    ports:
        - "80:80"
        - "443:443"
    volumes:
        - ./rancher:/var/lib/rancher
```

Listing 1: Example docker-compose file to start Rancher Kubernetes in a container.

```
version: '3'
services:
  registry:
    image: registry:2
    container_name: "image-registry"
    ports:
        - "5000:5000"
    volumes:
        - ./registry:/var/lib/registry
```

Listing 2: Example docker-compose file for starting a simple imgae registry (e.g., to be used in a test- or integration environment).

[44] https://jfrog.com/container-registry/

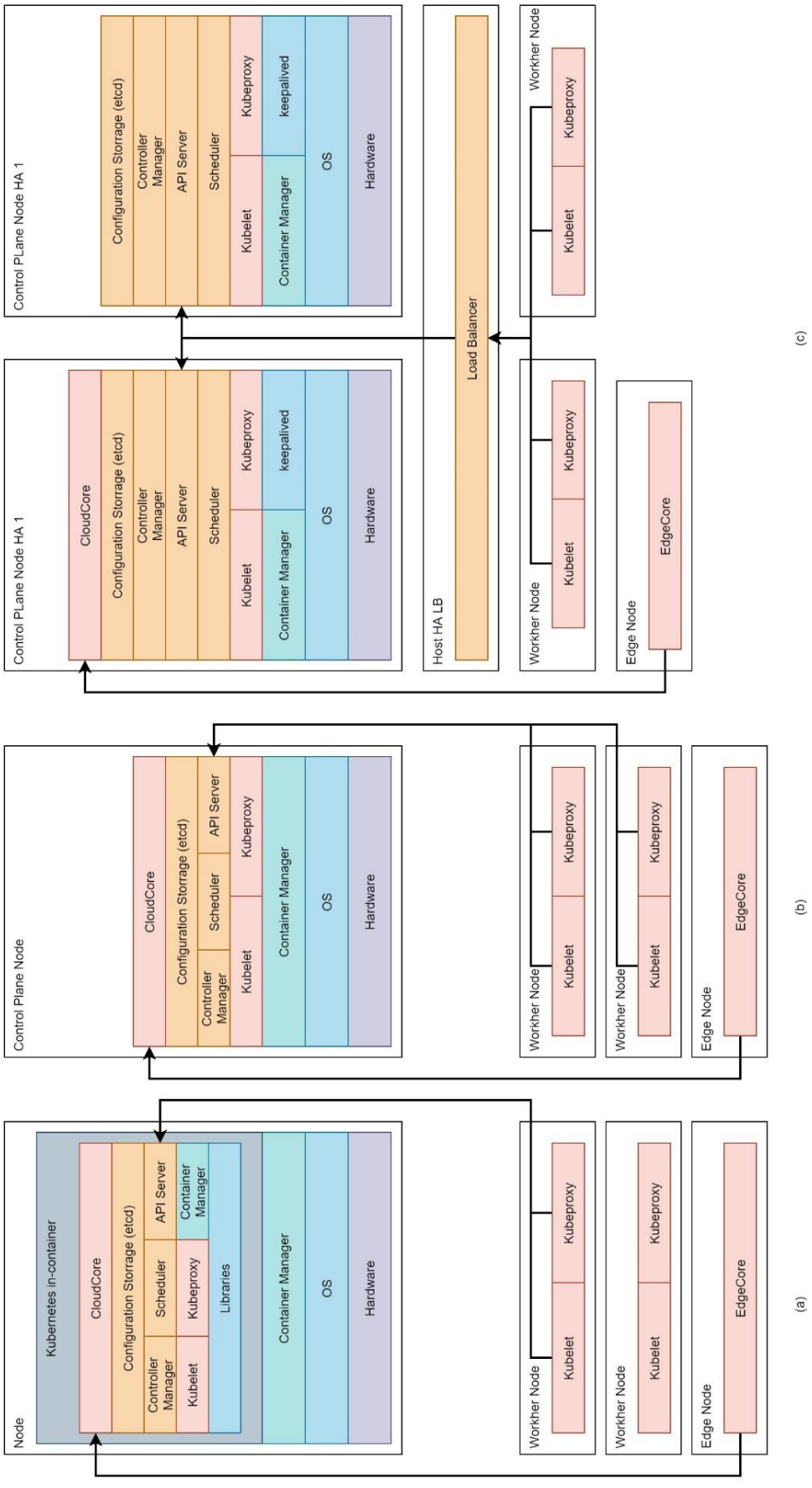


Figure 4.6: Different instantiation of the Kubernetes software stack providing the same functionality but with different levels of availability.

4.3.3 Access and Identity Management

The Kubernetes software for service orchestration on both the Cloud and the Edge level provides its API via the Kubernetes API Server[45]. To access this API there is no need to log in into the system and no session handling is supported. As documented in the according Kubernetes documentation[46] every API request is unique and must contain everything that is needed to authenticate and authorize the request[47].

Thus, every request must assert an identity to Kubernetes in one of multiple possible methods. Thereby Kubernetes never itself authenticates users; it validates assertions. Kubernetes does store service accounts, which however do not represent people. Instead, they represent anything that interacts with something else in Kubernetes. Since a service account has no expiration, if one is leaked and no one knows, it can be abused continuously until discovered. Thus, service accounts should never be used to identify users.

Kubernetes does not itself check any identity. However, Kubernetes provides mechanisms to check the validity of an identity. One of the mechanisms uses Open ID Connect[48] to authenticate users and actions. The INDIGO-IAM, which was introduces in IoTwins deliverable D2.2 (page 33) is based on Open ID Connect (which in turn is built on top of OAuth2.0[49]). The supported authentication method works like this:

1. The user logs in to IAM.
2. IAM generates an `id_token` and a `refresh_token`.
3. The `id_token` is used to assert the user's identity to Kubernetes.
4. When the `id_token` has expired, the `refresh_token` is used to generate a new `id_token`.

An `id_token` is a JSON Web Token (JWT) that contains the following information:

1. Who the user is.
2. What groups the user is a member of (optionally).
3. How long the token is valid.
4. It contains a digital signature to validate that the JWT hasn't been tampered with.

The JWT is passed on every request from the command line tool `kubectl` to Kubernetes. The `id_token` is called a Bearer Token because it grants the bearer access without any additional checks. This means if a system in the flow of an API call—such as a service mesh proxy, validating webhook or mutating webhook—were to leak this token, it could be abused by an attacker. Because these tokens are so easily abused, they should have very short life spans (e.g., one minute).

[45] https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
[46] https://kubernetes.io/docs/reference/access-authn-authz/authentication/
[47] see also https://www.linuxjournal.com/content/kubernetes-identity-management-authentication
[48] https://openid.net/connect/
[49] https://oauth.net/2/

4.3.4 Communication

To retrieve data from sensor devices in testbed 7 both protocols IEC61850[50] and Modbus[51] are used. At the Edge level the data is shared between the local services using the publish subscribe communication systems nats.io[52] and MQTT[53] (for details and a comparison with other techniques see also IoTwins deliverable D2.2). When legacy components or of-the-shelf software is introduced into an existing architecture (like databases or the configuration solution Consul[54]), the APIs of these services is used directly or adapted to the Edge communication system (nats.io). Since all pre-existing software integrated on used in testbed 7 provide a REST API, they are used directly by the implemented custom services without any adapter software in-between.

To transfer the data from the Edge devices into the backend both OPC UA PubSub and MQTT are used, depending on the setup. For establishing the secure connection with the Mindsphere Cloud the dotnet[55] based library implementation[56] is used which was made public as an open-source component during the project by project-partner SAG. At the Cloud level the communication systems that are provided by the according Cloud provider are used. In case this communication can be freely configured MQTT is used.

4.4 AI-based Operation of Smart Grids

4.4.1 Sensorless Control – Measurement Estimation

As described above, the goal of this use-case is to use the IoTwins architecture to enable control and monitoring of parts of the power grid without installing additional sensors. Smart meter measurements, which are received time-delayed at the Cloud level, are used to train Artificial Neural Networks (ANN) to estimate the measurements based on data that is available online (like the voltage and currents measurements at the transformer or weather data).

Thus, even though the data from the smart meters is received with a time delay of 1 day at the Cloud level and might not have the desired resolution, the ANN can be trained at the Cloud level to be used at the Edge level to calculate the estimated values at that point in the power grid. Figure 4.7 shows the components that were used to implement the scenario. The architecture was deployed in the Virtual Aspern Seestadt Testbed (see also Figure 4.2 above).

To enable the Training, Rollout and Update Workflow the following preparations have to be performed:

1. Cross-Compile all python libraries for all supportes CPU architectures and make them locally available using the *Python Library Storrage*.
2. Build base OCI images containing the basic components that take the data from Edge devices message bus, preprocesses them (e.g. scaling to acceptable range), puts them into a model and writes back the output of the model to the local message bus. This part of the Cross Build Service takes a prepared base image (which contain the base Python system), the application code, and the

[50] https://webstore.iec.ch/searchform&q=61850
[51] https://modbus.org/
[52] https://nats.io/
[53] https://mqtt.org/
[54] https://www.consul.io/
[55] https://dotnet.microsoft.com/
[56] https://github.com/siemens/opc-ua-pubsub-dotnet

python libraries from the *Python Library Storrage* to build the *Base Model Execution Image* (one for each CPU architecture). These images are stored in the *Image Registry* (see also section 4.3.2).

The basic training, rollout and update workflow is as follows:

1. On the Edge level the custom *Data Collection* service collects the measurements from the sensor devices using the Modbus protocol. In the Virtual Aspern Seestadt Testbed scenario the Modbus device is a software component that receives its measurement values from the multi modal simulation environment Bifrost. The *Data Collection* service puts the received data on the local message bus (in this use-case nats.io[57] is used) and pushes it to the data backend (in this scenario MQTT is used to easily integrate with already implemented components, while the afore mentioned UPC UA pub sub implementation[56] is used in the Real Aspern Sesstadt Testbed). The *Data Collection* service is the only service on the Edge Device that has to be adapted to the device settings (Modbus, UPC UA, register addresses). All other components on the Edge level use the message bus for communication and are thus device independent.
2. At the Cloud level the data that was pushed by the *Data Collection* service is stored to an NO-SQL time series database (in this setup InfluxDB) for later comparison with estimated data. In addition the database is used to store the measurement data that was received time delayed (and possibly in a non-sufficient resolution) via the custom *Data Importer* service.
3. The *Machine Learning Core* takes the historical data from the Measurement DB (original measurements from the Edge level and time delayed data from the Cloud level) and trains an ANN to generate estimations of the time delayed data using the Edge level measurements as an input.
4. The *Deployment Manager* forwards the resulting model to the *Container Build* service which takes the *Base Model Execution Images* (see above) and generates *Model Specific Execution Images* by adding the model in an additional layer on top of the *Base Model Execution Images*. The resulting images are again stored in the *Image Registry*.
5. Once the *Model Specific Execution Images* are created, the *Deployment Manager* invokes the deployment via the Kubernetes services (using the Kubernetes API) to the Edge node for which the model was trained. The Kubernetes components (see section 4.3.2) communicate the information to the Edge device via the EdgeCore components. The CRI-O component on the Edge device downloads the *Model Specific Execution Image* to the Edge device and starts it. In case there alreay is a *Model Specific Execution Image* running on the Edge device only the model layer of the image is to be downloaded which significantly increases the download speed. In addition no extra data channel is to be set up for managing the models.
6. Once the model is used to estimate the measurements, these estimations are transferred back to the data backend where they are used to measure the quality of the model and trigger a re-training if necessary.

The components that were implemented to realize the described processes were integrated using the implementation of the IoTwins architecture described above. The IoTwins architecture proved to fulfill the needs for the use-case Sensorless Control and to be useful in guiding the implementation process for the use-case specific custom software components.

[57] https://nats.io/

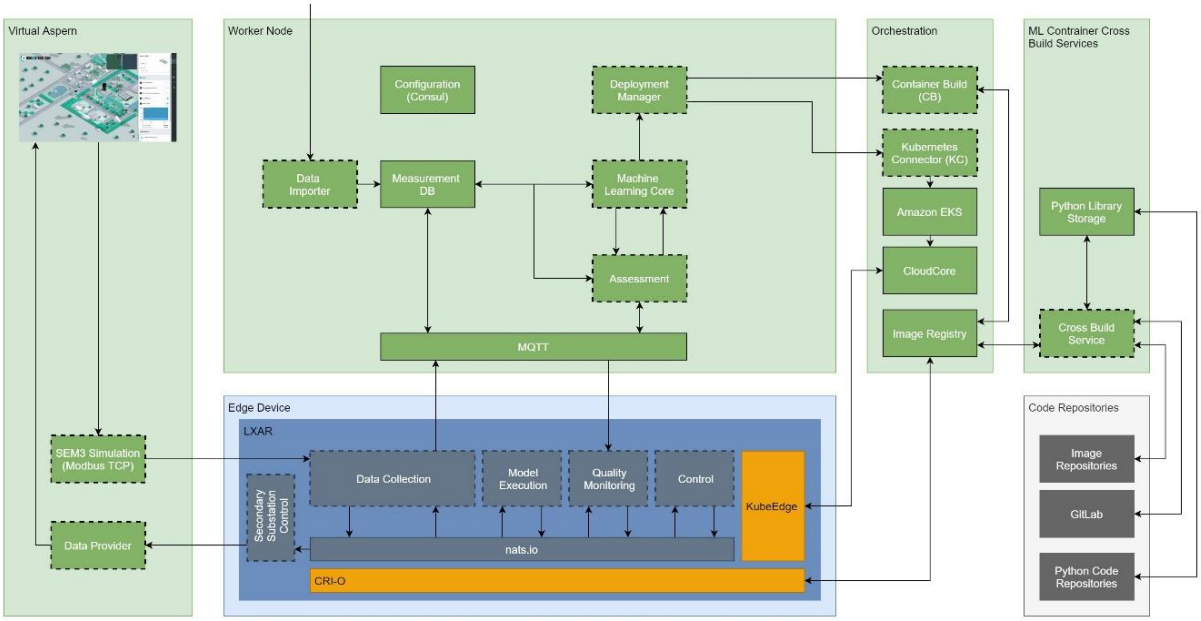


Figure 4.7: IoTwins architecture instance to implement use-case Sensorless-Control

4.4.2 Pattern detection on Cloud and Edge Level

The goal of this use-case is to monitor the state of the smart grid using means to detect and monitor patterns in the measured data. Therefore, in a first step the historical data (see IoTwins deliverable D5.1) was analysed (see the following sub-sections). In a second step ML algorithms were designed that automatically detect patterns from which the system state can be derived.

While the analysis of the data by the domain expert is a manual task, the training of ML algorithms can be performed automatically on the Cloud level where the data is collected. Online monitoring of the states during operation can be done on the Edge level, so that data is only transferred in case of an (approaching) state change.

4.4.2.1 Grid data

The Real Aspern Seestadt Testbed is divided into 12 measurement fields, whereas each field is dedicated to a distribution substation. To be able to successfully monitor and analyse the testbed, Grid Monitoring Devices (GMD) are installed in the individual measurement fields. These devices provide measurements of the grid values with a resolution of 2.5 min and are in this case available for a period of several years (2016 – 2019, see also deliverable D5.1).

The available grid measurements at each GMD are related to a 3-phase low voltage grid section and include the Root Means Square (RMS) values of voltages (U1 - U3), currents (I1 - I3), active powers (P1 - P3), and reactive powers (Q1 - Q3). Figure 4.8 shows several example timeseries measurement of a distribution substation for 24 hours. The voltage in this case is fluctuating in a range of 232V to 238V. The reactive power is mostly negative, which indicates that the transformer behaves in a capacitive way within this time window. One can also see that the voltage measurement at 12:00 and the reactive power measurement are correlated.

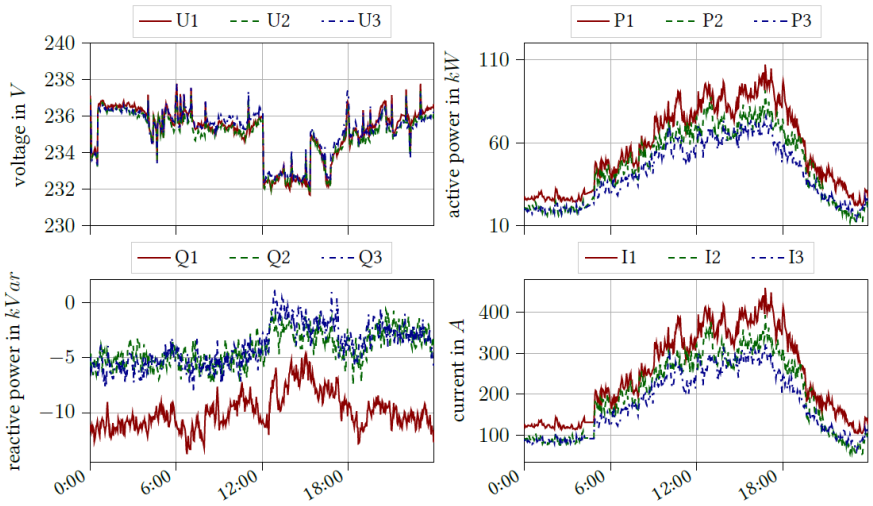


Figure 4.8: Example recordings of voltage level, active-power, reactive power and current for one day in testbed 7.

The active power measurement plotted over the duration of the day in Figure 4.8, clearly shows a recurring day profile. In the early hours from 06:00 onwards, the power increases steadily, whereas the inversion point is reached shortly before 18:00 and then the power decreases again with a negative slope. It is also worth mentioning that there is a significant difference in the rising and falling slope of the day profile. Since the voltage level is relatively stable the active power and the current are highly correlated.

4.4.2.2 Environmental data

To improve the detection of patterns and performance related information the available heterogeneous data should be included in the analysis process. This data can be segmented into external and internal information, whereas the external data consist of:

- **Weather data** (temperature, solar radiation, amount of daylight)
  The temperature and solar radiation are observed from local sensors in the testbed and are available with a sampling rate of 1 hour. The data about the daily amount of daylight is gathered with the help of the python library 'suntime'. The therefore used coordinates (latitude: 48:22573692335911_, longitude: 16:508141942206418_) are those of the Aspern Seestadt.
- **Social behaviour related data**
  Data that is very easy to acquire but can have a big impact on the Smart Grid is the distinction between weekends, workdays, and holidays. This information does have direct influence on the way people are producing and consuming energy. In this case the data is obtained with the python library 'holidays'[58].

One essential point for extracting patterns is the knowledge of the internal structure of the grid. This includes the components, energy consumers, energy producers and the geographic distribution of the individual parts. This so-called domain knowledge for this testbed is listed below:

- **Battery storage systems** - Prevention of load peaks or storing the over-production of energy
- **Photo-voltaic systems** - Renewable energy source
- **Heating systems** - Temperature controlled heat pumps
- **Energy consumers** - Industry and commercial buildings, residential areas, schools, car parks
- **Geographic information** - Latitude and longitude coordinates of the distribution substations
- **Grid plan** - Description of the grid measurement concept with the information on how the individual substations are interconnected and distributed through the grid.

[58] https://pypi.org/project/holidays/

A summary of the available dataset which is used for the analysis and pattern extraction is illustrated in the following table:

data	type	sample rate	components	unit	source
grid data - 12 distribution substations	time series	2.5 min	active power $P1 - P3$ reactive power $Q1 - Q3$ voltage $U1 - U3$ current $I1 - I3$	kW Var V A	influxDB
weather	time series	1 min 1 min 1 day	temperature solar radiation suntime	$^{\circ}C$ W/m^2 h	csv - file csv - file python package
social behaviour	categorical	–	"weekend", "working day", "holiday"	–	python package

Table 4.1 Historical heterogeneous Smart Grid dataset

4.4.2.3 Time Series visualization for data exploration

To visualise how the data and especially the time series are changing over the different days a day profile animation using the python library plotly[59] is implemented. Figure 4.9 shows the active power of all three phases plotted over the minutes since midnight. By pressing the play button, one can monitor the animation of all the lined-up days (30 days in this case), one after the other. This tool turned out to be highly useful to quickly get an overview about the development of the timeseries data and it also provides a useful way to visually identify irregularities or anomalies.

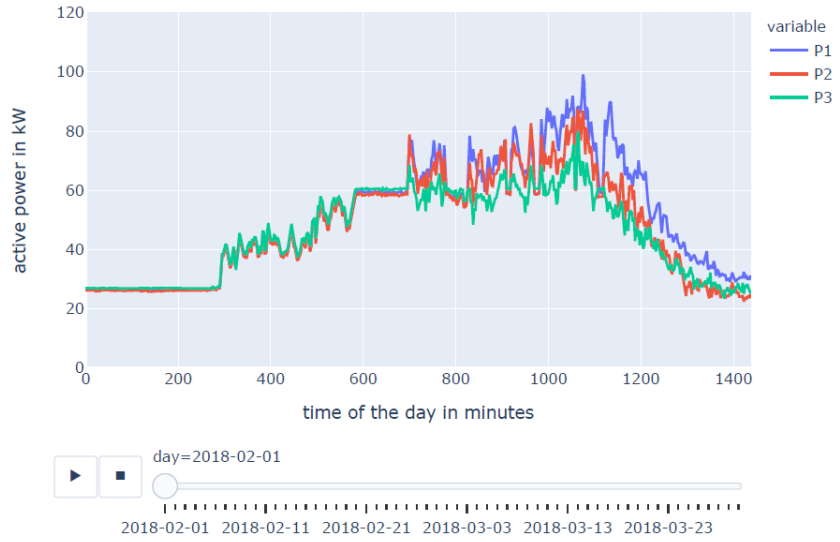


Figure 4.9: Animation displaying the change in active power of all three phases

To be able to analyse the distribution of power consumption in the power grid a tool was implemented to visualize the temporal changes in the spatial energy consumption. The spatial information was chosen accordingly to the position in the grid plan. However, using the same principle this could easily be replaced by geographic coordinates. Figure 4.10 illustrates an example where one can monitor the different bubbles which correspond to the power consumption in the according substations. The size of each bubble correlates with the daily mean active power consumption at the substation. When the play button is pressed the animation shows the power consumption of the different lined up days (30 days in this case) and thus depicts the change in for each measurement field.

[59] https://plotly.com/

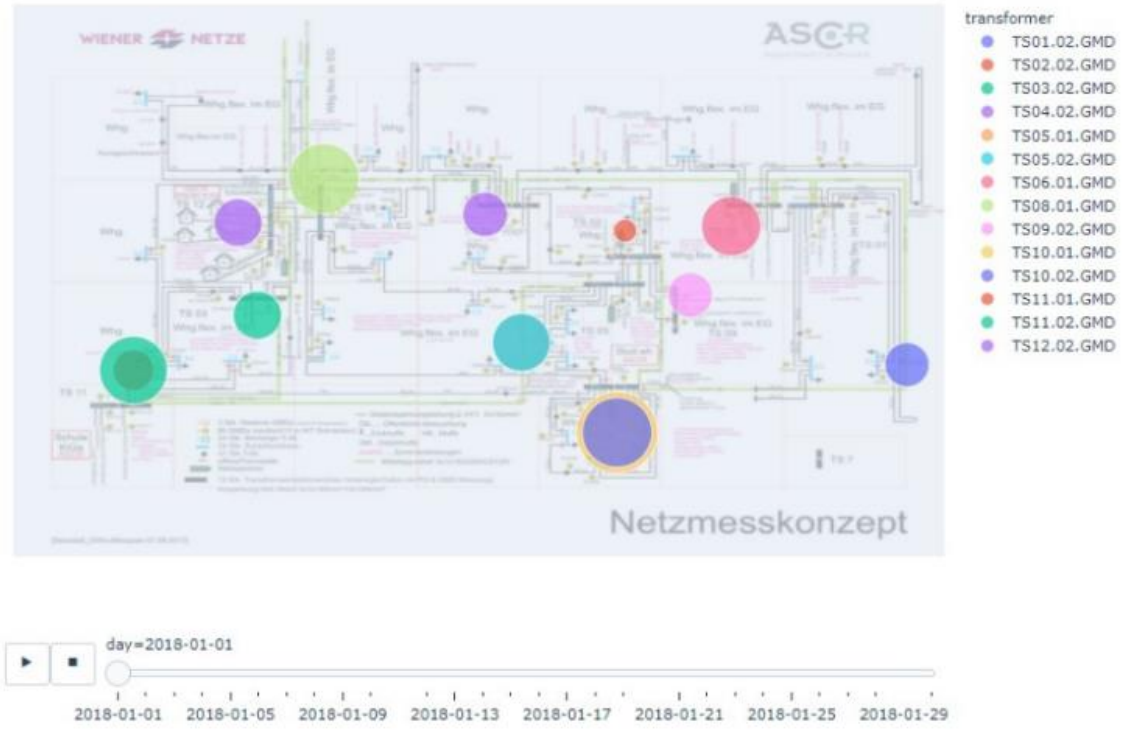


Figure 4.10 Example for an animated buble diagram showing the change of power consumption in the substations

4.4.2.4 Historical heterogenous data analysis

Based on the manual analysis described in the previous sub-sections, the behaviour of the distribution substations was compared and patterns from these substations were extracted from the recorded historical data for 2 years-time-window (start date: 2017-01-01, end date: 2018-12-31).

The following visualizations/tables were created for each distribution substation (an example plot/table is provided for each):

- Scatter plot of the grid measurements (Figure 4.11)
- Scatter plot of temperature, solar radiation, and the main grid measurements (Figure 4.12)
- Time Series plots of heterogenous data (Figure 4.13)
- Correlation matrix for the heterogeneous measurements (Table 4.2)

After the plots and tables described in the previous section were generated, a domain expert analysed the data for meaningful patterns which then were labelled accordingly. These patterns can then be used to design and train Machine Learning models to detect these patterns even in different environments (see following sub-sections). Once the patterns are automatically detected they can be used to estimate the quality of the systems performance.

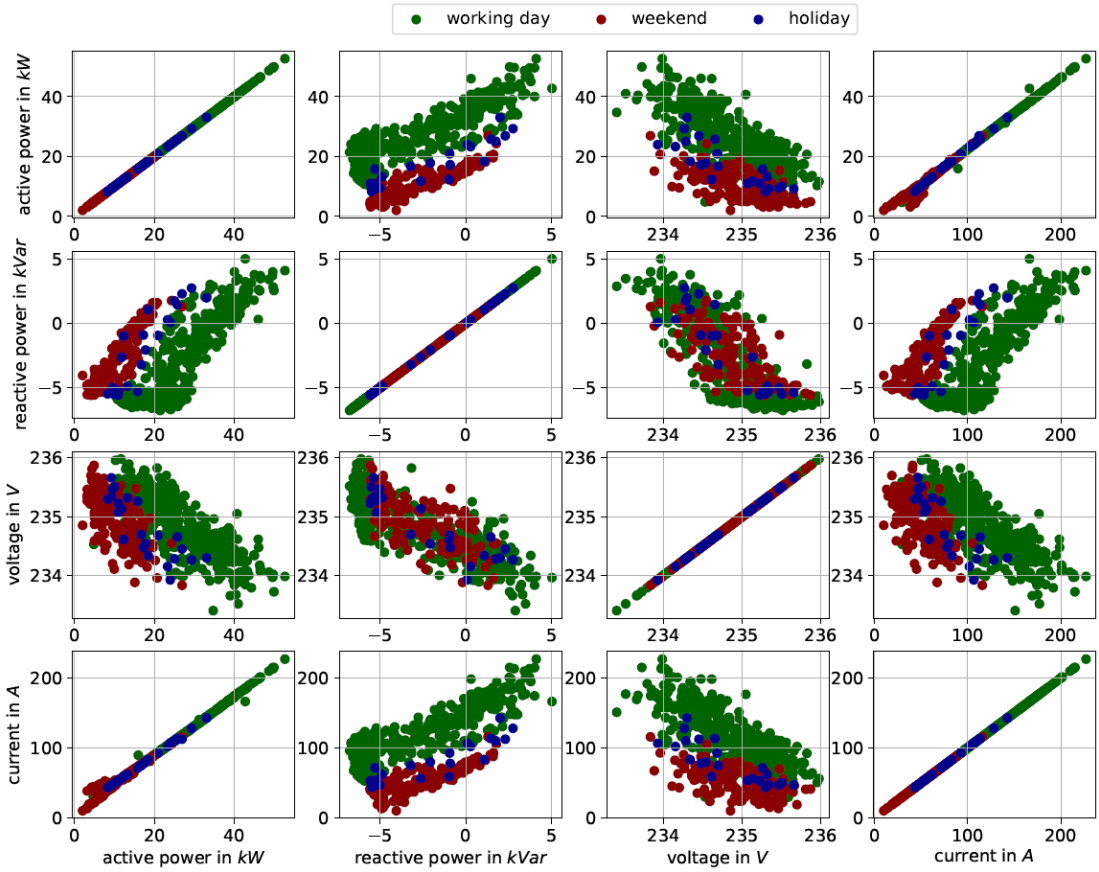


Figure 4.11: Example for a scatter plot of the power grid measurements

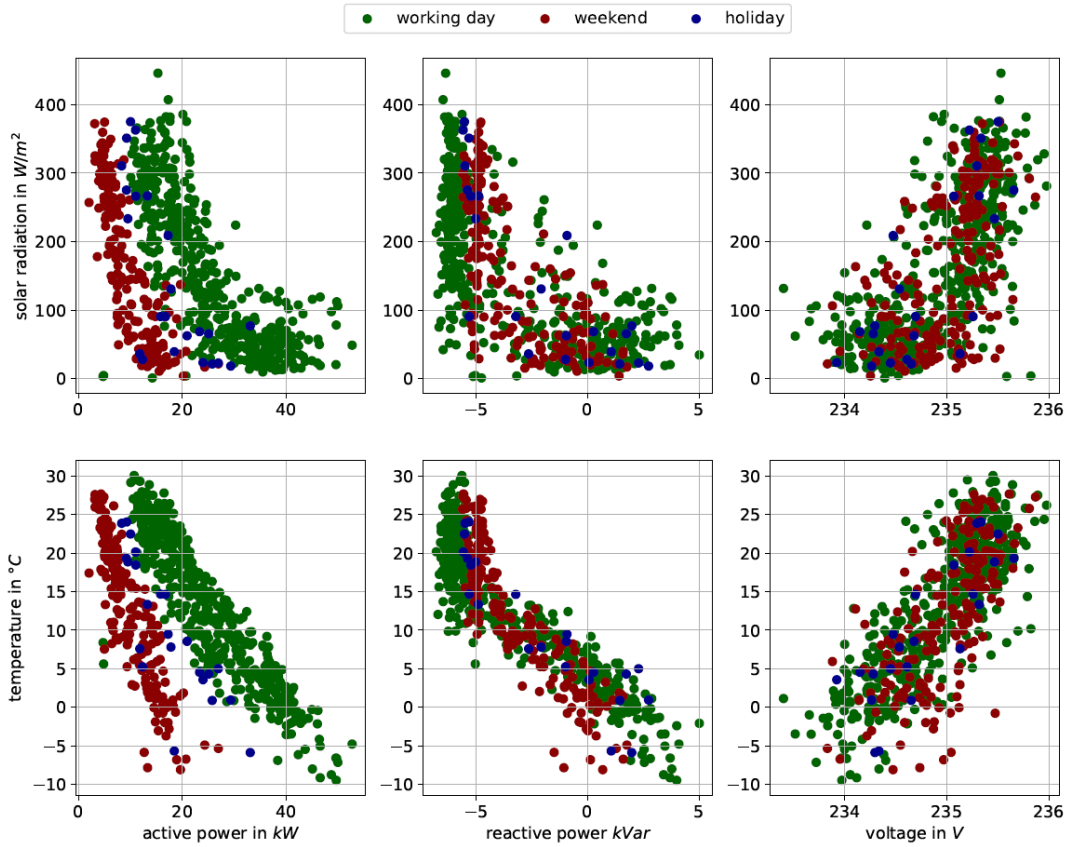


Figure 4.12 Example for a scatter plot f the temperature, solar radiation and the main grid measurements

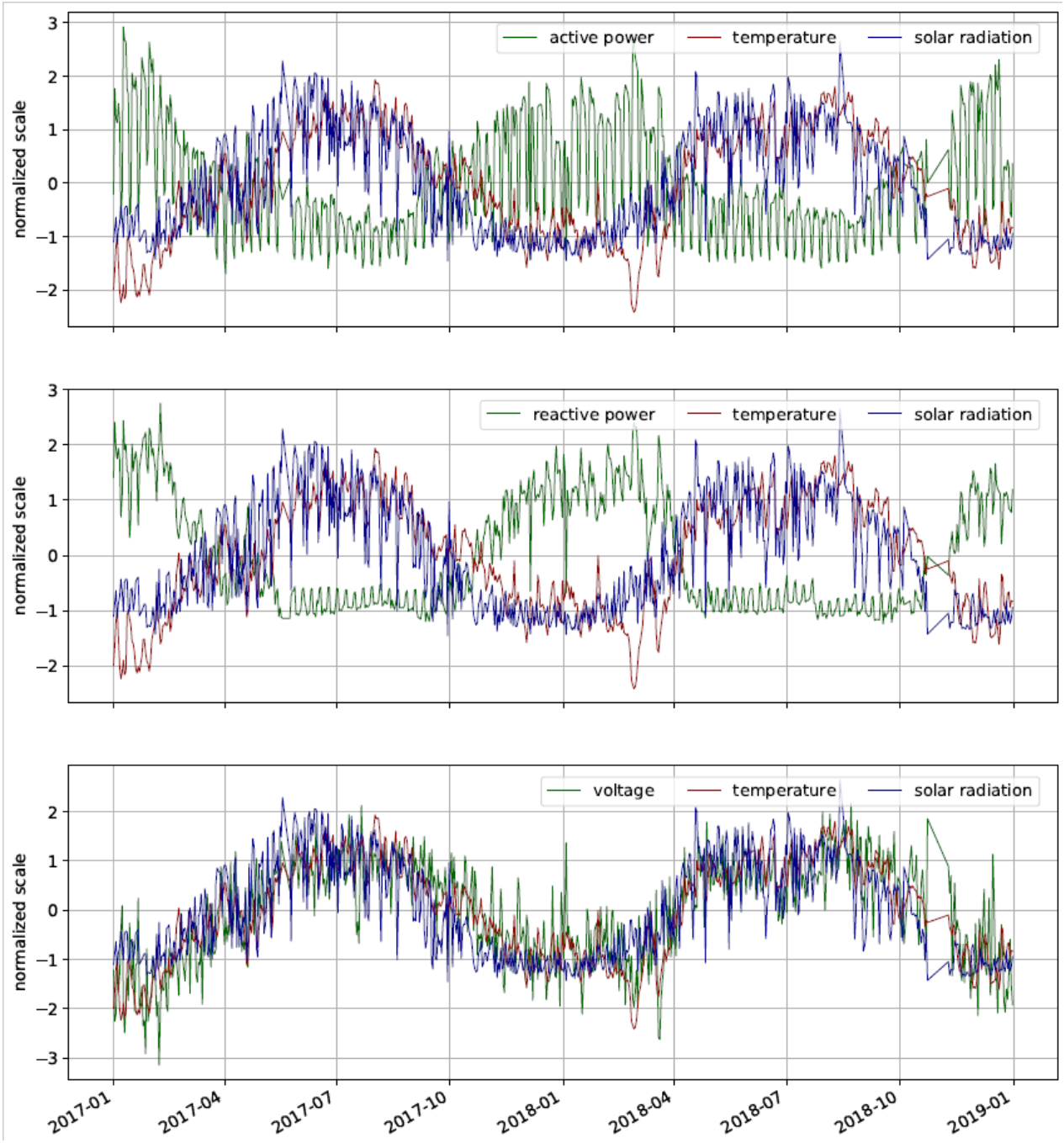


Figure 4.13: Example for a time series plot of the heterogeneous data

	I_mean	P_mean	U_mean	Q_mean	Temperature	Solar Radiation	Suntime
I_mean	1.000	0.997	-0.658	0.692	-0.668	-0.582	-0.607
P_mean	0.997	1.000	-0.662	0.693	-0.675	-0.591	-0.613
U_mean	-0.658	-0.662	1.000	-0.833	0.787	0.656	0.693
Q_mean	0.692	0.693	-0.833	1.000	-0.902	-0.712	-0.791
Temperature	**-0.668**	**-0.675**	**0.787**	**-0.902**	1.000	0.788	0.831
Solar Radiation	-0.582	-0.591	0.656	-0.712	0.788	1.000	0.839
Suntime	-0.607	-0.613	0.693	-0.791	0.831	0.839	1.000

Table 4.2: Example for the correlation matrix between the heterogenous measurements

4.4.2.5 Extracted patterns and trends in the daily mean

In this section the patterns that were found in the daily mean values are described.

Pattern 1: Power consumption difference between working days and weekends

The scatter plot between the active power and the reactive power depicted in Figure 4.10 shows that there is a unique distinction between the power consumption during the week (marked in green) and during the weekend (marked in red). Additionally, one can see the power consumption on holidays (marked in blue) which is also dedicated to lower values compared to the values during a normal working week. When checking the topological data for this substation it turns out that the main consumers are a school and a kindergarten. So, there are naturally less consumers present on the weekends. This knowledge about the switching behaviour in the power consumption can be very useful when trying to optimize modern energy communities.

Pattern 2: Temperature dependent active power consumption

This pattern is visible in Figure 4.13 and focuses on the relation between the active power and the temperature. In addition to the distinction between weekdays and weekends one can see a negative correlation between the temperature and the power consumption. In this special case the correlation coefficient equals r = 0.675. This inverse behaviour can also be recognized by focusing on the time series representation of the active power and the temperature (see Figure 4.12). This pattern can be caused by heating systems (e.g., heat pumps) and the correlation is especially high for substations where there is no battery system installed. The installed battery systems cover up this pattern, as they are supposed to smooth out such fluctuations and especially grid peaks. The knowledge about the temperature dependent power consumption can be used to make predictions based on temperature forecasts, or if this temperature dependent behaviour is not welcome, this information about the correlation can be used to install and design battery systems that compensate this behaviour in an efficient way.

Pattern 3: Temperature dependent reactive power consumption behaviour

After analysing the substations, three further reactive power consumption patterns were identified in addition to the case where there is little or no correlation between the reactive power and the other features:

- Temperature dependent power consumption clusters
- Negative correlation between the temperature and the reactive power
- Positive correlation between the temperature and the reactive power (see Figure 4.13)
  While the first two patterns can be related to temperature-depending heating pump activities, the third one could be caused by a cooling system.

Pattern 4: Solar radiation dependent active power consumption

When comparing the correlation coefficients of substations where a PV system is installed, there is always a stronger correlation between the solar radiation and the active power. This therefore also causes a reduction of the temperature dependent active power consumption and flattens the consumption behaviour of the grid.

Anomaly 1: Outlier in the feature space

Considering the feature space X= [I_mean, P_mean, U_mean, Temperature, Solar Radiation] and observing the extracted visualizations of all the different substations one can see outliers and different clusters. These outliers can be related to anomalous behaviour of the grid. The reasons for these outliers are hard to determine but possible causes could be, e.g., system failures or missing values in the time series.

4.4.2.6 Extracted patterns and trends in the high-resolution time series

Pattern 5: Different day profiles because of the environmental influences

As mentioned in the description of the grid data, one can observe characteristic day profile curves in the active power consumption during the day. In addition to the classic behaviour, where the consumption increases steadily in the early hours and decreases again in the evening after the change point, there are also different effects on the day profiles due to the external or internal domain-specific influences on the grid. Investigating the substations and their heterogeneous environment the following influences on the day profiles were identified:

- Social behaviour influence (working day, weekend)
- Heating system influence (heat pumps)
- Battery storage influence
- Photo-voltaic influence

Anomaly 2: Irregular behaviour in grid time series

The presented visualization tools for animating the different time series also revealed the following anomaly in the data:

Active power time series anomaly - at substations with installed battery storage systems there exists a battery maintenance event, where the battery is fully discharged and charged during the day (cause by a manual reset by an operator at the battery itself). This event occurs at a few random days in the year.

4.4.2.7 Power consumption prediction based on temperature

For distribution substations where there is a strong correlation between the temperature and the active power consumption, one can create a function which allows to predict the power consumption based on temperature forecasts. This ML concept is therefore related to **Pattern 2: temperature dependent active power consumption**. We can define the underlying problem in this case as approximating a function

$$\bar{P}(\bar{T}) = g(\bar{T}, \mathbf{w}),$$

which allows us to calculate the daily mean power consumption P, based on the daily mean temperature T and the model coefficients *w*. After the necessary data was collected (see above) the next step is to choose an appropriate model class for solving the task. In this case, as this problem is related to a two-dimensional space (P, T) a linear neuron, which can also be described in form of a polynomial regression, is an appropriate choice. The coefficients and features can directly be reformulated to:

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & w_2 & \dots & w_k \end{bmatrix}^T, \quad \mathbf{x} = \begin{bmatrix} 1 & \bar{T} & \bar{T}^2 & \dots & \bar{T}^k \end{bmatrix}^T.$$

The feature matrix **X** is constructed as

$$\mathbf{X} = \begin{bmatrix} 1 & \bar{T}_1 & \bar{T}_1^2 & \dots & \bar{T}_1^k \\ 1 & \bar{T}_2 & \bar{T}_2^2 & \dots & \bar{T}_2^k \\ . & . & . & \dots & . \\ 1 & \bar{T}_n & \bar{T}_n^2 & \dots & \bar{T}_n^k \end{bmatrix} \quad \in \quad \mathbb{R}^{n \times (k+1)},$$

and the target values **y** as

$$\mathbf{y} = \begin{bmatrix} \bar{P}_1 \\ \bar{P}_2 \\ . \\ \bar{P}_n \end{bmatrix} \quad \in \quad \mathbb{R}^{n \times 1}.$$

The variable *n* denotes the number of investigated days and *k* the order of the polynomial. Using this concatenated form of the daily mean power consumptions and the polynomial expansions of the daily mean temperatures we can fit and select the model by obtaining the optimal parameter in the following way:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

Results for an example substation:

The result for one example substation is visualized in Figure 4.14. The investigated time period for this analysis corresponds to two years: 2017-01-01 – 2019-01-01 and the examined substation shows a negative linear correlation between active power and temperature in lower regions of the temperature and a positive linear correlation if the temperature reaches higher values. As already visible by just observing the datapoints, there seems to be an overall quadratic relationship. Once we have implemented the algorithm for fitting the polynomial function to the corresponding data, one can simply choose different orders of *k* and visualize the results.

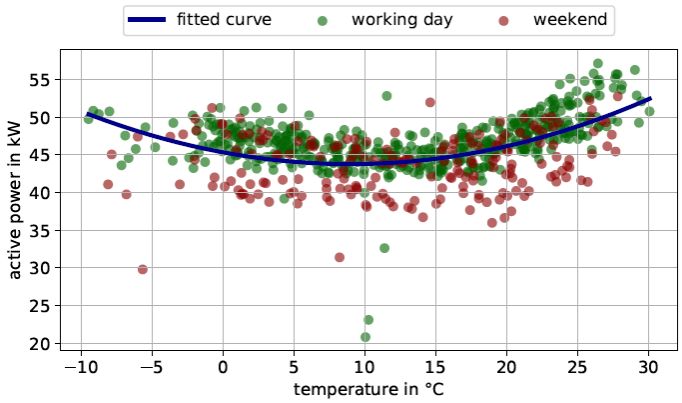


Figure 4.14: Example for active power consumption prediction based on temperature

In this case the order of the polynomial is chosen as k = 2 and we therefore acquire the following function for the daily mean power consumption prediction:

$$\bar{P}(\bar{T}) = g(\bar{T}, \mathbf{w}) = 45.35 - 0.35\,\bar{T} + 0.02\,\bar{T}^2.$$

4.4.2.8 Unsupervised day profile clustering

As shown in the detailed exploratory data analysis, there is a very strong influence of topology related factors (e.g., battery storage systems, photo-voltaic systems, heating systems) on the recorded data. Therefore, it is of interest to find out how these external factors are influencing the behaviour of the grid or the respectively substations. Regarding the undesired system states it is beneficial to know the root causes (see section 4.5) of the changes in the system states. However, the first step is to identify the different system states.

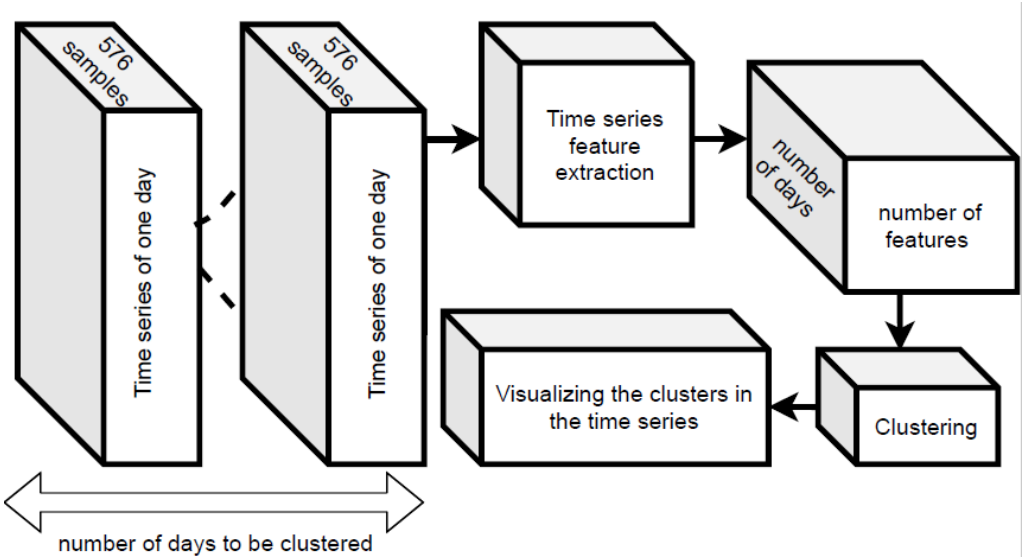


Figure 4.15: Machine Learning concept for day profile clustering

The following Machine Learning concept (see Figure 4.15) is related to **Pattern 5: different day profiles because of the environmental influences**. This unsupervised approach demonstrates how information about the different system states can be extracted in an unsupervised manner, i.e., without any prior knowledge or labelling. In this special case the system state is defined as a defined power consumption day profile. In the next section a concept is introduced which uses a clustering algorithm that processes features extracted from the daily time series. After the day profile clusters are evaluated and visualised a polynomial regression approach is used to fit a function which can then be used as a representative for the respective day profile class.

Feature Extraction:

For this use-case the time series is filtered and dived into windows of 24 hours, whereby midnight is always used as a boundary between two windows. The feature extraction step is now performed on each daily time series. When selecting the features, it is important to consider which information one wants to extract from the time series, as the features are representing the problem. If the features are not chosen properly, one might not be able to solve the problem, or simply produce weak results. The Time Series Feature Extraction Library TSFEL[60] which is used in this work, makes it possible to choose between the following feature domains:

- Temporal: e.g., entropy, auto-correlation, peak to peak distance
- Statistical: e.g., histogram, interquartile range, maximum, minimum
- Spectral: e.g., FFT coefficients, Maximum frequency

[60] https://tsfel.readthedocs.io/

In this use-case the goal is to generate as much knowledge as possible about different existing day profiles. Thus, all three domains are considered for the feature extraction and clustering.

Clustering algorithm:

Once the features are extracted, which are now the representatives for each day profile, this information can be used, and an unsupervised clustering algorithm can be applied. One of the most used unsupervised clustering approaches is k-means. One of its features is, that it is quite well suited to spherical cluster shapes. But as in any case when one applies an unsupervised approach in a highly dimension feature space, one does not have lots of prior information about the cluster shape. Therefore, different approaches in addition to the k-means algorithm are tested and the one which best solves the problem is selected. The tested unsupervised clustering methods thus comprise k-means, hierarchical clustering, and mixture of gaussians. The k-means algorithm has revealed most of the information within the clusters. One disadvantage of the k-means clustering is, that one must choose the number of clusters k. To get a good estimation for the appropriate number of clusters, one can use the so-called elbow method. This method calculates the sum of squared distances between the data points and the cluster centroids. By monitoring this measure and increasing k one will observe a shape which resembles the form of an elbow. The case where one reaches the bend point of the elbow is simply the point where one does not significantly acquire more information by adding more clusters. Therefore, the reached number can be considered as an optimal number of clusters.

Day profile curve fitting - polynomial regression:

Once the different day profile clusters are derived, one can use this information to label the original time series data. The labelled time series can now be used to fit a polynomial function for each time series cluster.

By modifying and applying this supervised ML concept we can reformulate the goal towards fitting a function for each cluster to

$$P(t) = g(t, \mathbf{w}),$$

which describes the active power consumption depending on the time *t* and the polynomial coefficients ***w***. When the coefficients and features are defined as

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & w_2 & \dots & w_k \end{bmatrix}^T, \quad \mathbf{x} = \begin{bmatrix} 1 & t & t^2 & \dots & t^k \end{bmatrix}^T,$$

One can construct the feature matrix **X** as

$$\mathbf{X} = \begin{bmatrix} 1 & t_{c_1,1} & t_{c_1,1}^2 & \dots & t_{c_1,1}^k \\ 1 & t_{c_1,2} & t_{c_1,2}^2 & \dots & t_{c_1,2}^k \\ . & . & . & \dots & . \\ 1 & t_{c_1,n} & t_{c_1,n}^2 & \dots & t_{c_1,n}^k \\ 1 & t_{c_2,1} & t_{c_1,1}^2 & \dots & t_{c_1,1}^k \\ . & . & . & \dots & . \\ 1 & t_{c_n,n} & t_{c_n,n}^2 & \dots & t_{c_n,n}^k \end{bmatrix} \quad \in \quad \mathbb{R}^{(n \cdot c) \times (k+1)},$$

and the target values **y** as

$$\mathbf{y} = \begin{bmatrix} P_{c_1}(t_1) \\ P_{c_1}(t_2) \\ . \\ P_{c_1}(t_n) \\ P_{c_2}(t_1) \\ . \\ P_{c_n}(t_n) \end{bmatrix} \quad \in \quad \mathbb{R}^{(n \cdot c) \times 1}.$$

In this *c* denotes the number of days in the investigated cluster and c_i corresponds to the respective day profile. The variable *n* denotes the number of samples in each cluster and *k* corresponds to the order of the polynomial. Using this concatenated form of the different day profiles for one cluster we can obtain the optimal polynomial parameter in the following way,

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

Results for the example substation:

As described, the different day profiles result from environmental influences and the installed technologies. Especially a battery storage system in a substation strongly influences the daily power consumption profile. The most common objective for installing such systems is to prevent load peaks. However, these systems also can cause undesired behaviours, e.g., load steps because of discharged batteries, battery maintenance events, or disturbance because of battery failures. These events can be detected using the described clustering approach. The investigated period for this analysis is one year: 2018-01-01 – 2019-01-01

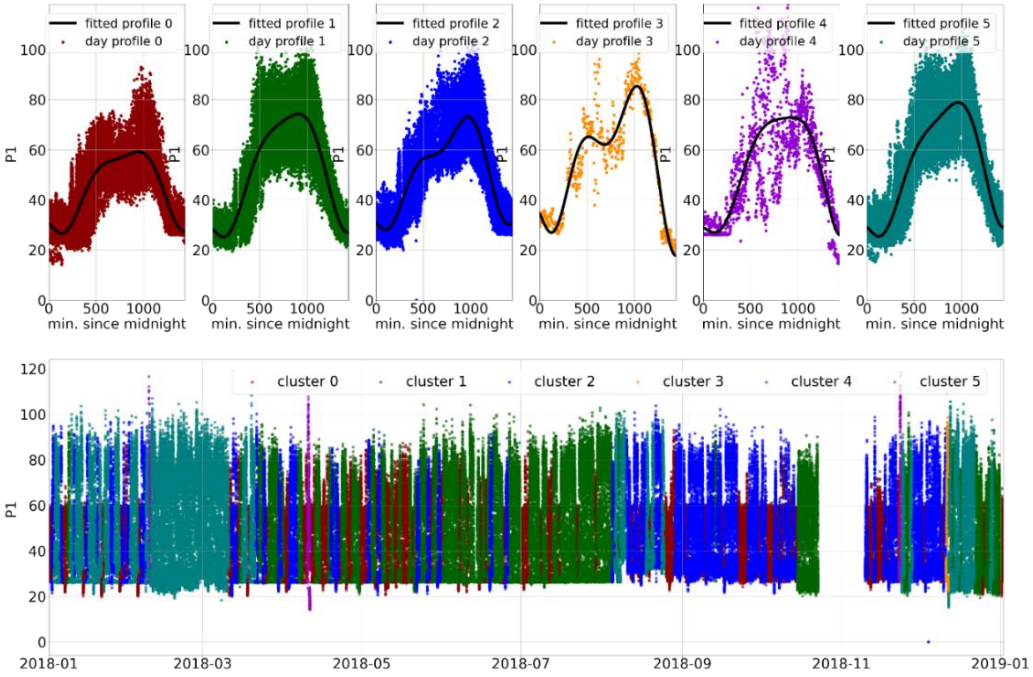


Figure 4.16: Example for resulting day profile classes/clusters

Figure 4.16 illustrates the results for the day profile clustering approach for an example substation. The number of applied clusters is determined using the elbow method and results in k = 6. Investigating the fitted profile of each cluster, one can see that the desired behaviour (which in this case is preventing load peaks over 60kW) is only reached on days which fall are grouped in cluster 0. Cluster 2 is related to the case where the battery can cover up the load peaks partly during the day, while in the evening ours when the battery is discharged, the measurements directly represent the consumption by the consumers. Cluster 1 and Cluster 5 contain all days where there is no peak shaving active (possible reasons therefore are e.g., battery failure, peak shaving deactivated, discharged battery). Cluster 3 corresponds to an anomalous behaviour of the battery. Cluster 4 contains all day profiles which correspond to battery maintenance events. Such battery maintenance events are initialized by field engineers on-site and cause a fully discharging and charging of the battery storage system.

Table 4.3 shows the occurrence frequency of each cluster in absolute numbers and in %. The system is only working in the desired system state (cluster 0) in 24.5% of the investigated period. Cluster 2 is occurring 24.8%. This means the peak shaving system is active in only ≈ 50% of the time. During the rest of the time, the controller is inactive or shows anomalous behaviour (approximately one percent of the time).

	cluster 0	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5
absolute occurrence	85	103	86	1	3	69
occurrence in %	24.5	29.7	24.8	0.3	0.9	19.8

Table 4.3: Results for the occurrence of the different day profiles in the example substation

4.4.3 Online state detection

To be able to detect events not only retrospectively but also while they are happening (at Cloud level and at Edge level), ML algorithms were trained based on data sets that were created using the multi modal simulation framework Bifrost.

4.4.3.1 Battery Maintenance Event Classification

As illustrated in section 4.4.2.8, one can classify the historical time series into a set of day profiles. After applying this classification to the data recorded for a substation with an installed battery storage system, one can identify the four different classes shown in Figure 4.17, for which the occurrence in percent is listed in table 4.4.

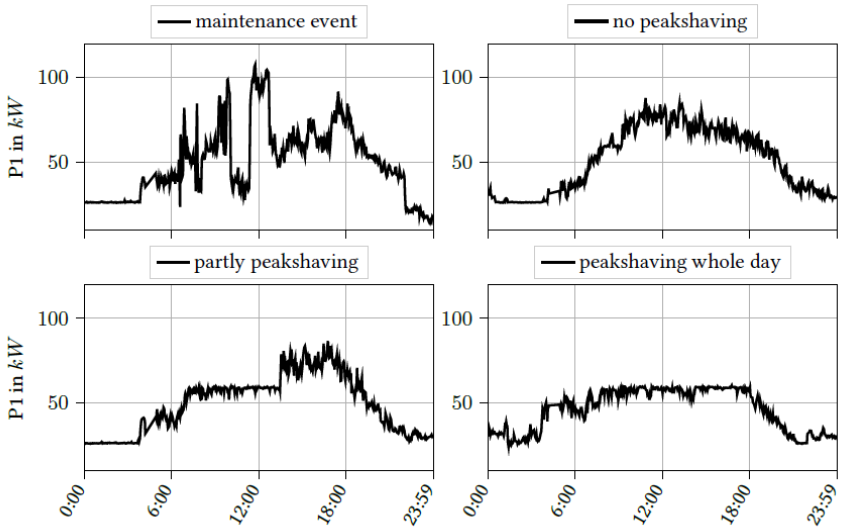


Figure 4.17: Example for typical consumption day profiles of an investigated transformer

day-profile	maintenance event	no peakshav.	partly peakshav.	whole day peak-shav.
appearance	1%	50%	24.5%	24.5%

Table 4.4: Occurrence of typical day profiles

Depending on the use-case and ML algorithm, the classification of the time series as described in the previous sub-section can directly be used as input for ML training methods. However, the clustering shows that the class "maintenance event" only occurs with 1% in this data set. To train neural networks to recognize this event in the real-world grid data, significantly more training samples are required.

To reproduce the battery maintenance event in the multi-modal simulation framework Bifrost, a concept for re-enacting this maintenance event was created which was realized by implementing a module which reconstructs the root cause (triggering on site) and parameters (storage size of the battery and the maximum charging and discharging power) of the event.

Figure 4.18 illustrates the created/simulated normalized training data p1, where the battery maintenance event is at least triggered once per day. The second half of this graph contains two example day profiles. The corresponding labels are automatically extracted from the simulation and mark the discharging and charging period of the battery.

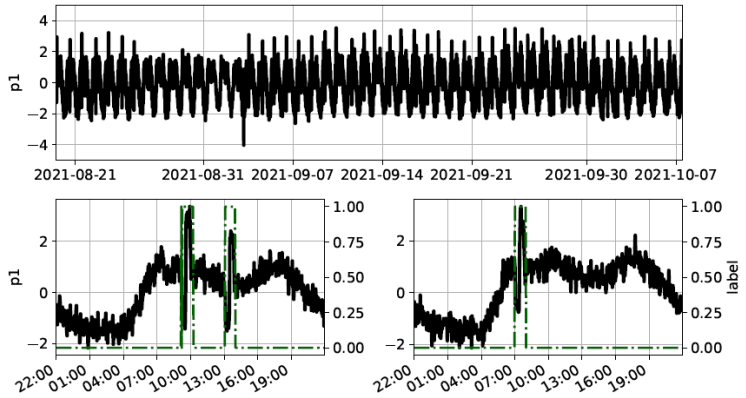


Figure 4.18: Result of the co-simulation for generating ML training data

A common technique for analysing/classifying time series are Long-Short-Term-Memory[61] (LSTM) networks. They can be used to learn patterns in time series and thus fit the battery maintenance use-case. In testbed 7 the Keras[62] framework was used to implement the LSTM. Table 4.5 lists the implemented layers as well as those parameters, that differ from the standard implementation. Additionally, a dropout layer (dropout rate of 0.1) was inserted between the individual layers to avoid over-fitting.

layer	L1: LSTM	L2: LSTM	L3: Dense
parameter	units = 264 rec.act. = "tanh"	units = 64 rec.act. = "tanh"	act. = "tanh"

Table 4.5: Neural network architecture for battery maintenance event detection

[61] https://en.wikipedia.org/wiki/Long_short-term_memory
[62] https://github.com/keras-team/keras

The network is trained with a sequence length of 25 (which corresponds to approx. 1 hour at a sampling rate of 2.5 minutes). The input to the network is defined as

$$\mathbf{X}_{train} = [\mathbf{p1}, \dot{\mathbf{p1}}, sin(\omega \mathbf{t})] \in \mathbb{R}^{25\times 3},$$

where **p1** is the normalized (zero mean, unit standard deviation) time series of the power consumption and $\dot{\mathbf{p1}}$ its derivative. To let the network also recognize/learn relative temporal relationships, the third feature vector consists of the relative time $sin(\omega \boldsymbol{t})$, where $\omega = \frac{2\pi}{24\cdot 60\cdot 60}\frac{rad}{s}$.

After training the LSTM network (at Cloud Level) with 30 epochs and a batch size of 200 samples, a training accuracy of 98% was achieved. This shows that the network can detect the event in the simulated data. The results are then to be applied to the real grid data. Figure 4.19 shows the prediction of the network on the historical grid time series for the year 2018 (with a recording gap due to malfunctioning sensors during autumn). The network detects 5 events, which fits to the result of the day profile clustering (battery maintenance events appear 1% of the time) and the results of the manual analysis of the data. In contrast to the day profile clustering approach, the LSTM network can be used to detect the event while it happens.

Figure 4.20 illustrates four detailed day profiles and their battery maintenance event prediction. In all four cases anomalous behaviour is prevalent and the battery controller can be the root cause for this behaviour. However, the two day-profiles on the right-hand side of the figure show that on 2018-04-10 the maintenance event is detected in addition to two other anomalies while on 2018-11-23 the event is carried out twice in a row and is detected on both occasions.

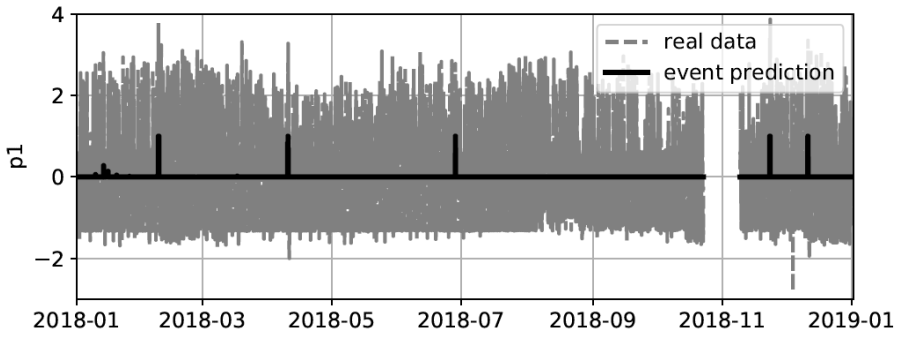


Figure 4.19: Overall battery maintenance event classification

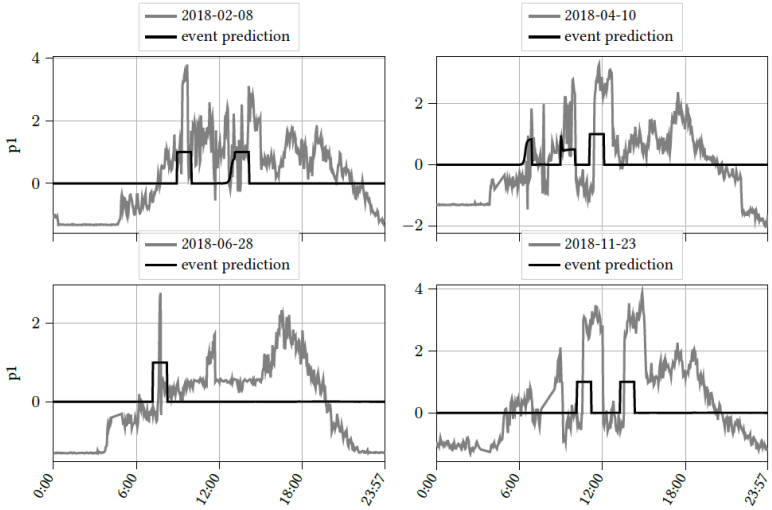


Figure 4.20: Daily based battery maintenance event classification

4.4.4 Anomaly detection using Machine Learning / Power Quality App

To prevent damages in power grids and attached devices, anomalies such as short circuits must be detected and reacted to quickly, e.g., by switching off the affected parts of the grid. The detection of anomalies is typically accomplished by analysing key parameters such as voltage and current using signal processing algorithms. However, classical methods are restricted to pre-defined signal patterns and may fail in case of unforeseen events. To solve this problem, more flexible approaches based on artificial intelligence have been developed that aim to cover a wider spectrum of faults. This includes slowly developing anomalies that may arise, for example, when trees grow into power transmission lines.

As part of IoTwins, Siemens is working on the implementation of such approaches using neural networks. These networks are supposed to run on the Edge, i.e., close to the field devices that generate the data. Note that the maximum time span to react on a fault may – depending on its type – be in the range of milliseconds. Hence, the neural networks must be executed in real-time. A further challenge stems from the fact that the target devices are usually not very powerful for cost reasons. To address these and further challenges, Siemens started with an elucidation of the key requirements. Moreover, a software stack has been defined that permits real-time processing of sensor data (time series) obtained from the field devices. This includes the ability to pre-process the data before it is fed into a neural network. The architecture also comprises a database for long-term data storage and visualization via a dashboard. Figure 4.21 gives a high-level overview of the prototypical architecture.

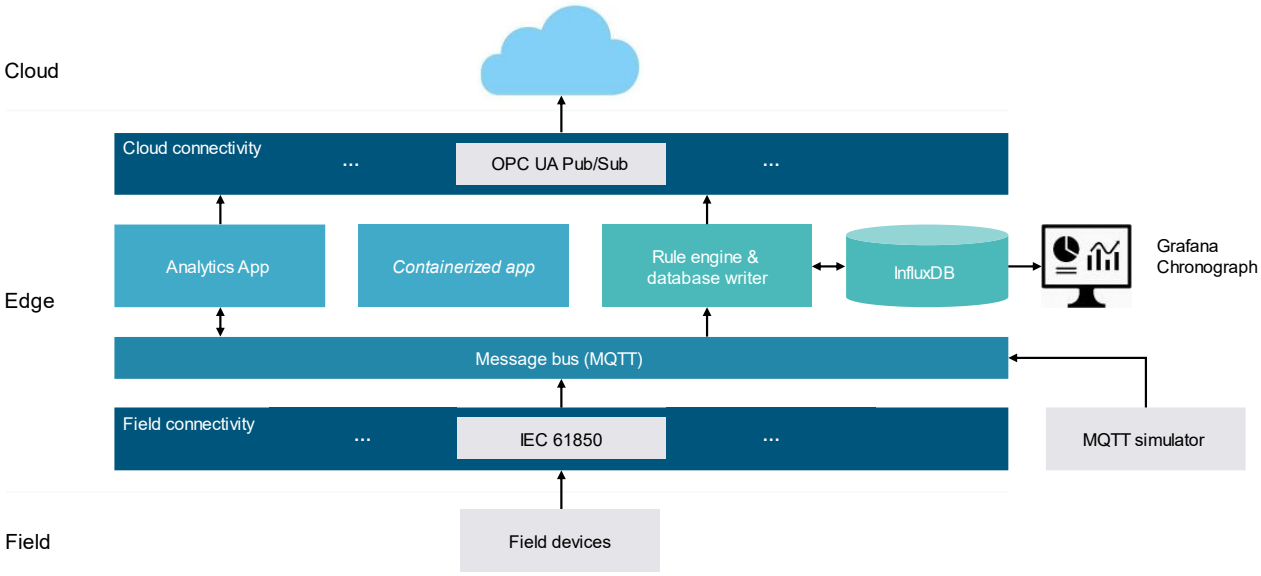


Figure 4.21: High-level software architecture of Edge-based anomaly detection for power grids

The network, which has been used for IoTwins, analyses the incoming data and – in case an anomaly occurs – reports its type. The training was done using simulations based on electrical models of typical faults. To cope with the timing constraints and the limited compute power at the Edge, the neural network has been optimized extensively. For example, quantization and model reductions have been applied for faster inference. By combining multiple approaches, the inference time could be reduced significantly and matches the required frequency of 4kHz.

To evaluate our approach, we developed a demonstrator that covers all aspects from collection of power grid data over AI-based analysis to visualization. Moreover, the neural network was successfully evaluated using recorded real-world data. While being trained with simulated data, it shows high accuracy, making it

suitable for practical application. For testing purposes and improved customer experience, both the inputs and the output are visualized using a Grafana-based dashboard (see Figure 4.22).

Figure 4.22: Dashboard for visualising the results of anomaly detection in power grids

4.5 Root Cause Analysis (RCA)

As Cyber-physical Systems (CPS) like the Smart Grid are becoming more complex and dynamic the need to explain, why certain system events happen becomes more and more important. Thereby it is important to incorporate the cross-domain characteristics of systems which dealing with large number of heterogeneous data sources. Solutions must thereby also consider that there are diverse end-users for which these explanations must be compiled. In addition, when a root cause is identified, the information can be used to recreate the event to create data for the event detection using ANNs (see section 4.4.3.1).

This section describes an approach selected to address these challenges in testbed number 7, based on the approaches for event detection in describe din previous sections (see for example section 4.4.3.1). In the case of the battery maintenance event the event label is already associated with the root cause “battery maintenance was activated” of this event. However, events are not always directly related to a root cause. The underlying cause, why the clustering of the day profiles at a given substation results in a certain cluster, can also be related to a change in the production of power by the installed PV systems which in turn is related to weather events. The indirect relation of weather data and power profiles was already discussed and implicitly handled in section 4.4.2.7. The Root Cause Analysis (RCA) tries to make these relations explicit to be able to influence or even control them. The presented approach thereby integrates the result generated in a research projects with research partners TU Wien and the nationally funded project PoSyCo[63]. For testbed 7 the approach based on semantic web technologies was selected and is described below in more detail since it perfectly integrates with the IoTwins architecture. As described above, the data acquisition and

[63] https://projekte.ffg.at/projekt/3036508

event detection which is the basis for the process is already implemented. The algorithms for the integration with domain knowledge provided by a domain expert is currently being implemented.

Figure 4.23 shows the abstracted RCA process in smart grids as a chain of four steps, which are implemented by the two approaches:

- Data and Knowledge Acquisition
- Event and Causality Detection
- Root Cause Identification
- Root Cause Explanation

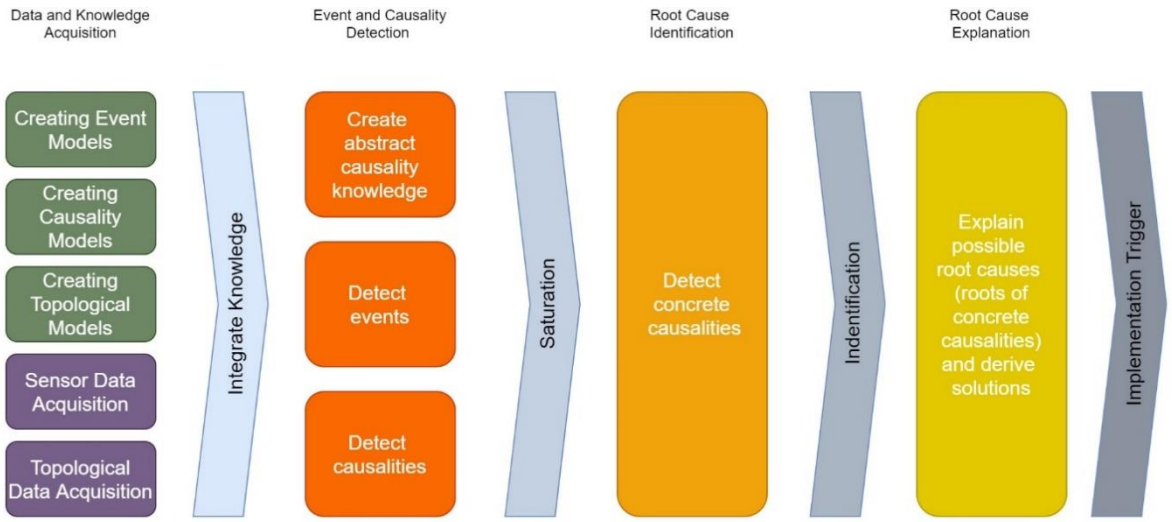


Figure 4.23: Root Cause Analysis process to support the deduction of concrete causalities in smart grid events.

Thereby in accordance with the publication on "Explainable Cyber-Physical Energy Systems based on Knowledge Graph"[64] an *abstract causality* is a relationship that is known to exists between two abstract or physical properties based on general knowledge such as how solar intensity causes power to be produced in a solar panel. The *concrete causality* is a relationship between the actual devices that observe or actuates the properties in the system. For example, a faulty sensor causes the Edge device not to function properly. The topology of the system contextualizes abstract causality to specify concrete causality.

The approach chosen to implement the abstract RCA process shown in Figure 4.23 uses semantic web technologies. This work was presented in the MSCPED paper "Explainable Cyber-Physical Energy Systems based on Knowledge Graph"[64]. How the architecture of such a framework fits to the IoTwins architecture (see deliverable D2.2) is shown in Figure 4.24. The data sources that are used for the Data and Knowledge Acquisition and require the interaction with a user are depicted on the right side ("causality modelling",

[64] Peb R. Aryan, Fajar J. Ekaputra, Marta Sabou, Daniel Hauer, Ralf Mosshammer, Alfred Einfalt, Tomasz Miksa, Andreas Rauber.: "Explainable Cyber-Physical Energy Systems based on Knowledge Graph". 8th workshop on Modeling and Simulation on Cyber-Physical Energy Systems, 2020 http://semsys.ifs.tuwien.ac.at/presentation-at-mscpes-2020-simulation-support-for-explainable-cyber-physical-energy-systems/

"event modelling", and "topology modelling"), while the data acquisition with respect to measurements (voltage levels, power consumption etc.) is depicted on the bottom in the Edge node and on the top when it comes to gathering data from external data sources. The arrows indicate the flow of information from one source to another. Coloured boxes represent the process that can be manual or automated and grey boxes represent data structure that represent information that is generated by the processes. One of the central components is the data storage used to persist the knowledge graphs that are enriched during the Event and Causality Detection Phase by the Event Detection (see also section 4.4.3.1) and the Causality Detection processes.

The processed information is formalized and stored using the Explainable Cyber-Physical Systems Ontology[65]. Initially, the required information is identified and organized into three groups: ExpCPS (Explainable CPS) topology, events, and core. The main classes in these groups are shown in Figure 4.25. The topology group is based on Semantic Sensor Network Ontology[66] (SOSA) for modelling sensor and actuation data along with the feature and property where the data acquisition take place. *ExpCPS event* uses function ontology to define event detection from data generated in the *ExpCPS topology*. *ExpCPS core* contains essential concepts needed to model causality for generating explanation from events.

Both the Event Detection and the Causality Analysis operate on the information that is stored in the central data storage and the data that is retrieved directly from the Cyber Physical System (which includes power measurements, topological data about the power grid and weather data) and external data sources. Knowledge about the events that might happen in the system are acquired from domain experts as described in the previous sections. Additionally, the domain experts also provide knowledge about abstract causality or causal relationships that is needed to reason about how the system works. The causality knowledge is used to understand how things work in general such as the underlying relationship of physical properties.

The research presented in the cited MSCPES paper proposes to represent causalities as IF-THEN rules. Concrete causalities can then be derived from the stored data. The domain expert is required to specify two properties that are connected by causal relation and specify the condition under which the relation should hold. The condition can be formulated as a SPARQL query that can be executed on the knowledge graph and the concrete causality can then deduced based on this query. The proposed causality deduction algorithm then works by joining two sets of records: the set of device pairs that interacts with the properties involved in the rule and the set of device pairs that fulfils the causality condition query.

[65] http://pebbie.org/expcps
[66] https://www.w3.org/TR/vocab-ssn/

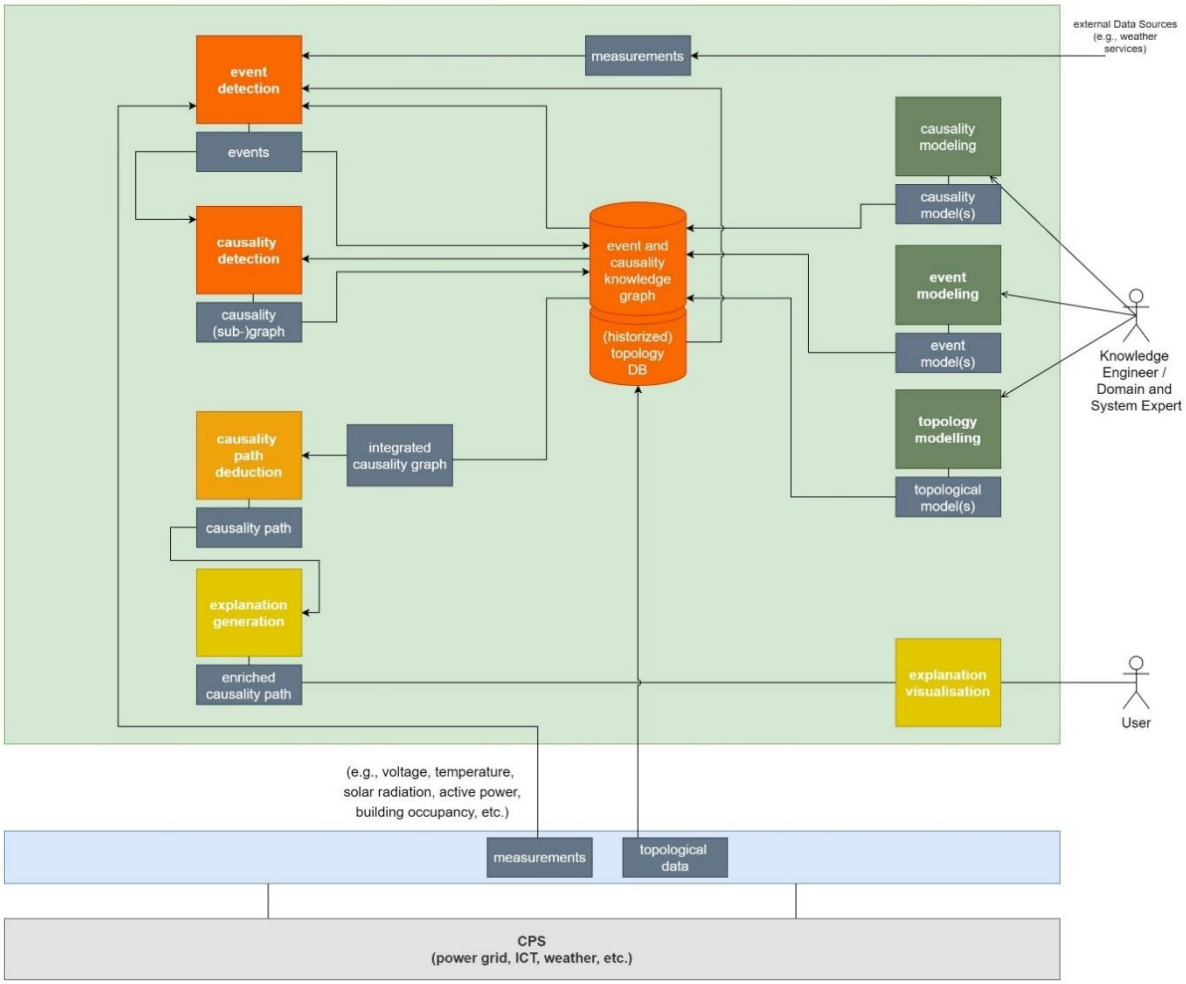


Figure 4.24: Architecture to implement the Root Cause Analysis process for the CPS Smart Grid using knowledge graphs in the IoTwins architecture (based on Figure 2 in the MSCEPS paper on "Explainable Cyber-Physical Energy Systems based on Knowledge Graph"[64], see also D2.2 for the IoTwins architecture).

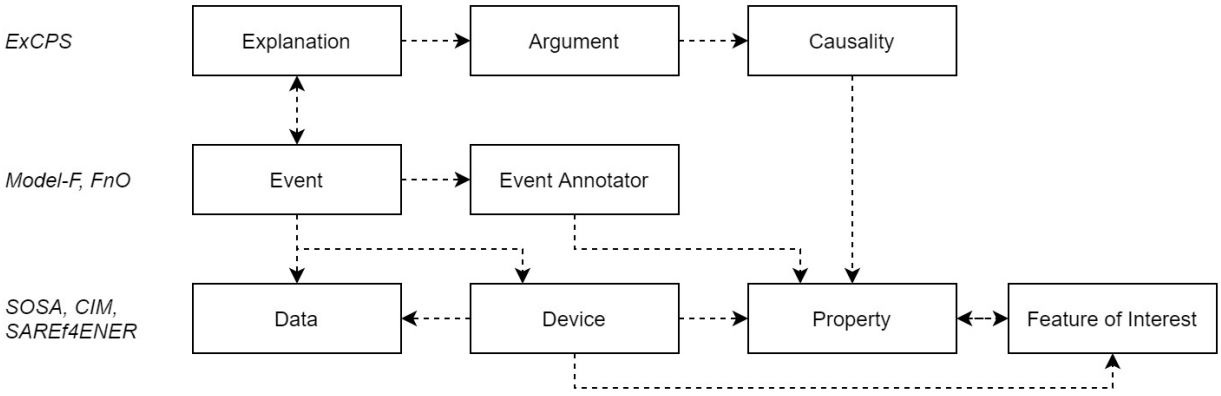


Figure 4.25: Classes in the ExCPS ontology (see http://pebbie.org/expcps) which is used to store and process system knowledge to derive causality paths in the Root Cause Analysis process (see Figure 3 in the MSCEPS paper on "Explainable Cyber-Physical Energy Systems based on Knowledge Graph"[64]).

The explanation for an event can then be represented as a graph where each node is an event and a pair of two nodes are related by causal relationships. The explanation generation algorithm is then formulated as a

graph construction algorithm based on the knowledge graph. Each event is associated with the device that generates data points (sensor observation or actuation log) which are the basis for the detected event. Each device is associated with the properties which have a causal relation to each other. To derive the relationship between two events the concrete causality derived previously is utilized. Thus, the resulting traversing path through this graph is the searched explanation. Even though the algorithm refers to the devices and the topology on the Edge and IoT level, the processes for RCA integrate knowledge on the Cloud level.

The explanation tree that the approach generates may contain many details and must be adapted to the user's needs and must be presented accordingly ("explanation visualisation", see Figure 4.24). An end customer, who is interested in why his/her car is currently being charged with less than the maximum power is not interested in the detailed information about all the event's assets but in the abstracted-out explanation to get a summary of the situation. A maintenance operator on the other hand might need to know the complete deduction chain as well as the root causes (which in terms of the explanation tree refers to the leaves).

5 Conclusions

We reported here the workings of the first version of the digital twins geared towards facility management. The three testbeds described here have addressed the full IoTwins digital twin architecture, to varying degrees depending on the specifics of each application. We have shown that there are positive outcomes and actionable information coming from each system, as a whole and from individual components or subsystems.

While the testbeds were affected differently by the COVID pandemic, we have also learned that adapting the systems for the platform and infrastructure is a complex process, where all services need review processes by different teams, from technical teams who must understand the implications and consequences of change, to legal, security, and operations teams that must ensure that regulations, data anonymisation, and overall safety are being followed correctly.

Future work will include an improved second version of the digital twins, and the transfer and replicability to other testbeds in WP6. One reflection from the followed approach is that in some testbeds the opposite order of tasks could have been more efficient: start from a smaller facility/problem, and later scale up the system to cover the larger, more ambitious problem.

Nevertheless, the second version of the digital twins will build upon the results reported here not only by addressing issues or performance, but also with added functionality, improved communication, and usage of the IoTwins platform.