MOEEBIUS

Modelling Optimization of Energy Efficiency in Buildings for Urban Sustainability

D4.5 MOEEBIUS District-Level Middleware

Version number: 1.0
Dissemination Level: PU
Lead Partner: HON
Interim Due date: 30/04/2017
Type of deliverable: Other
STATUS: Delivered

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 680517

Published in the framework of:

MOEEBIUS - Modelling Optimization of Energy Efficiency in Buildings for Urban Sustainability

MOEEBIUS website: www.moeebius.eu

Authors:

Jakub Malanik, Jiri Rojicek – HON

Borja Tellado Laraudogoitia – TECNALIA

Konstantinos Tsatsakis – HYP

With contributions from the project consortium as regards system deployment requirements to be met

Revision and history chart:

VERSION	DATE	EDITORS	COMMENT
1.0	29/04/2017	HON	Submitted to the EC

Disclaimer:

Table of content

List of tables

List of figures

Glossary

Acronym	Full name
Amazon AWS	Amazon Web Services
Amazon EC2	Amazon Elastic Compute Cloud
AMI	Amazon Machine Image
API	Application Programming Interface
BMS	Building Management System
CQL	Cassandra Query Language
CSV	Comma-separated values
DB	Database
EIP	Enterprise Integration Patterns
HTTP(S)	Hypertext Transfer Protocol (over TLP/over SSL/Secure)
IoT	Internet of Things
JMS	Java Message Service
JSON	JavaScript Object Notation
MOEEBIUS	Modelling Optimization of Energy Efficiency in Buildings for Urban Sustainability
MQTT	Message Queue Telemetry Transport
REST	Representation State Transfer
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
URL	Uniform Resource Locator
WSO2	Open source technology company developing service-oriented architecture middleware
WSO2 DAS	WSO2 Data Analytics Server
WSO2 DSS	WSO2 Data Services Server
WSO2 ESB	WSO2 Enterprise Service Bus
XML	eXtensible Markup Language

1 Executive summary

The aim of this deliverable is to describe the process of selecting a district MOEEBIUS middleware layer.

Following parts are dedicated to hardware choices, which are important to provide expected performance at reasonable cost. For some customers they can possibly differ from building level middleware, so we had to revisit in this document as well.

Similarly to D4.4, to explain the middleware expected use, the document then describes key middleware building blocks and their relationships. The most crucial part describes which functionalities developers can expect from the middleware and how they are expected to utilize it, however, the contents of this document should not be mixed with software development documentation. Finally, this part is accompanied by selected use cases.

Please note, that the key deliverable of the tasks 4.3 and 4.4 is not this document, but the middleware itself — this document describes the journey how it was achieved.

This deliverable is a continuation of D4.4 where middleware capabilities in terms of data-exchange and storage, have been described. The classical concept of middleware and its functionalities described in D4.4 will be extended in the current deliverable where the set of "tools" that will power the concept of the MOEEBIUS-Quest will be described.

MOEEBIUS project adopted this particularized middleware understanding in order to clearly describe the set of tools that give support to the MOEEBIUS-Pipe and MOEEBIUS-Quest layers.

2 Introduction

The classical concept of middleware is to introduce a layer providing services to software operations beyond those available from the operating system. It simplifies software development especially by unifying communication and data access in general.

Within MOEEBIUS framework the middleware serves two primary functions

- Data integration: unify data collection, i.e. offering pilot sites as well as future site owners support to easily send data to a common data store without developing any site-specific drivers;
- Application integration: allow analytics and workflows to easily hook complex data exchange processes and algorithms.

These core and fundamental requirements are also accompanied by others, also quite essential like security, authentication, scalability, messages managements, system monitoring, etc. — and such functionality should be transparent to end users.

This concept is valid regardless the abstraction level applied to the functionalities that want to be provided as a service.

2.1 Scope of this documentation

This document describes the WSO2 components that have been selected to implement the abstraction of the data analytics and process marshalling layers. The data-analytics and process marshalling activities are mainly linked to district level MOEEBIUS modules which are part of the MOEEBIUS-Quest.

Usually data analytics and workflow management are implemented as a core part of software developments and are tightly linked (hard coded) to the use case in which they are applied.

The MOEEBIUS project's concept for the middleware follows two main targets:

- Standardization: The use of a given set of tools (middleware) to implement the data analytic layer enables a common pattern to handle, MOEEBIUS-Quest level forecasting and optimization activities.
- Interoperability and Flexibility: MOEEBIUS-Quest requires to implement flows that collect data from many MOOEBIUS components. The data flows to implement would require several data transformations and manipulations which implementation and maintenance can be facilitated if they are implemented in a single layer.

The WS02 components selected to support the mentioned requirement are:

- WSO2 Data Analytic Server
- WSO2 Business Rule Server

The following chapters will describe the capabilities offered by each of the components mentioned above as well as other available alternatives, at the end of the document, the conclusions, will describe the rationale behind the WSO2 selection as data analytic and workflow middleware.

2.2 Document structure

This chapter focuses on motivation and general concept of middleware in the MOEEBIUS project context as well as describes the different available alternatives to fulfil the technical requirements imposed on the MOEEBIUS district level middleware.

Chapter 4 refers to MOEEBIUS middleware requirements, infrastructure platform choice, hardware selection and individual software components. Finally, Chapter 5 describes how the mentioned middleware functionalities fit some MOEEBIUS use cases.

2.3 Middleware concept

2.3.1 MOEEBIUS-Pipe VS MOEEBIUS-Quest

Middleware is software that functions as a conversion or translation layer. Middleware is also a consolidator and integrator. Custom-programmed middleware solutions have been developed for decades to enable one application to interface with another, which either runs on a different platform or comes from a different vendor. In particular, within European Projects several middleware solutions for buildings and smart grids have been reported. Some examples here include e.g. the IFC4-based middleware developed within BaaS fp7 project [7], the LinkSmart middleware developed within FP6 European Project HYDRA [8], the middleware developed within ENCOURAGE project of the ARTEMIS Industry Association call [9] and many others. The main downsides of these solutions are the following:

- they are heavily tailored towards the requirements of each umbrella project;
- their level of maturity, reliability, stability and security is below available commercial solutions;
- there is a huge lack of community support and public documentation in all these solutions.

Today, this effort can be reduced as there is a diverse group of commercially available middleware products which can be configured and tailored for specific needs.

Middleware supports and simplifies complex distributed applications. It may include web servers, application servers, messaging and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP, Web services, and service-oriented architecture.

Middleware often enables interoperability between applications that run on different operating systems, by supplying services so the application can exchange data in a standards-based way. Middleware sits "in the middle" between application software that may be working on different operating systems. It is similar to the middle layer of a three-tier single system architecture, except that it is stretched across multiple systems or applications.

2.3.2 Rapid Miner

RapidMiner is a centralized open source solution that enables users to create, deliver, and maintain predictive analytics.

RapidMiner suite of applications also comes with data integration, transformation, machine learning, and application integration. With such unified approach, RapidMiner expedites learning, improves standardization, and simplifies maintenance and extensibility, resulting to greatly boosted productivity and efficiency.

Featuring a powerful set of tools and functionalities, RapidMiner not only helps to understand and find value in data but enables but to create models and plans so that critical statistics can be extracted. RapidMiner implements also data exploration capabilities such as Graphs and Visualizations and Descriptive Statistics.

RapidMiner main features:

- Visual Workflow Designer
- Data Access and Management
- Graphs and Visualization
- Data Sampling, Partitioning, Replacement
- Similarity Calculation
- Clustering
- Bayesian Modelling
- Scoring

Additionally, the features of RapidMiner can be significantly enhanced with add-ons or extensions, many of which are also available for free. Its flexibility and

integrated GUI (graphical user interface) makes it suitable for pure data analytic activities but for the MOEEBIUS purposes it lacks the automation and integration capabilities required for further interoperability with third party tools.

2.3.3 H20

H2O is the outcome of a Start-Up launched by people in the orbit of Stanford University in 2011. Its development has been linked from the very bagging to the development of cloud based Big-Data platforms such as Hadoop. Originally written in Java and optimized for Key/Value type data structures handling, the algorithms are implemented on top of H2O's utilize the Java Fork/Join framework for multi-threading. The data is read in parallel and is distributed across the cluster and stored in memory in a columnar format in a compressed way.

The H20 implementation approach enables REST API access to all its capabilities an external program or script via JSON over HTTP. The Rest API is used by H2O's implement bindings for web interface (Flow UI), R (H2O-R), and Python (H2O-Python). The implementation of such bindings would enable the H20 integration in the MOEEBIUS framework, as substitute for the WSO2 DAS or as additional add-ons development framework.

H2O's main features:

- RandomForest, Regression models.
- Generalized Linear Modeling (GLM),
- Logistic regression, k-Means,
- High speed and scale for modeling and scoring over BigData
- Support for HDFS, S3, NoSQL, SQL
- Support for CSV format from local and distributed filesystems (nfs)

Taking into consideration the flexibility, interoperability and deep learning techniques modeling capabilities the adoption of H20 would not suppose any constraint or limitation for MOEEBIUS. Nevertheless, due to the availability, inside the WSO2 framework, of a native module (WSO2 DAS) for such implementation, in order to minimize the interoperability issues, the MOEEBIUS consortium decided to adopt it.

2.4 MOEEBIUS district level middleware requirements

The list of District Level Middleware as defined in D2.4 are presented in the following table:

Requirement	Priority
District Level Middleware	
District Middleware should provide access on high resolution data (15-minute granularity)	High
District Middleware should provide aggregated data (at building level) fully preserving privacy and security concerns	High
District Middleware should provide access about total building energy consumption (electricity/ heating) data	High
District Middleware should provide access about energy consumption data for each of the main device types examined in the project	High
District Middleware should provide access about total/ per device type demand flexibility data (aggregated data at building level)	High
District Middleware should provide access on real time building environmental conditions	High
District Middleware should provide access about real time building contextual data (occupancy data), fully preserving privacy concerns	Medium
Along with real time data, District Middleware will provide aggregated building energy performance simulation data as extracted from BEPS tool (several BEPS KPI parameters will be available at different spatial and temporal granularity)	High
Along with real time data, District Level Middleware will store historical data for further exploitation from aggregator side business applications	High
District Middleware should provide limited rule based process on raw data (spatial and time aggregation, etc…), by incorporating DIM (District Information Model) parameters to the dynamically retrieved raw data from premises	Medium
District Middleware should provide seamless interfaces to aggregator side business applications for retrieving real time and historical semantically enhanced	High

Table 1: Requirements for district level middleware.

Therefore, the role of Data Acquisition and Management Layer / District Level Middleware is to act as the building gateway to ensure communication with the Aggregator Hub. As with Building Level Data Acquisition and Management component, there are two main modules that consist of this software component. A proxy is hosted in the building environment and provides a bi-directional communication interface with the aggregator, acting that way as the gateway of the building. The gateway contains the knowledge to send and receive messages at an abstraction level higher than the level of information exchanged between the internal building system elements. Therefore, a certain level of aggregation is performed at the gateway, addressing that way privacy requirements by the end occupants in building premises (only aggregated information is available on DSM Aggregator Side).

Furthermore, the District level Data Acquisition and Management component acts as the district level middleware which enables semantic integration and orchestration of the different building types integrated at district level. The district level middleware establishes a transparent and homogeneous interface to building information, enabling the different business applications at aggregator side to access this information in a seamless way.

The selection of WSO2 for MOEEBIUS middleware does not imply that the middleware is finalized with no further work required. On contrary, WSO2 components have to be set up properly and configured for MOEEBIUS specific needs. Selecting WSO2 is rather like selecting an environment for building the middleware.

2.5 Infrastructure as a Service

After selecting WSO2 as the middleware platform, the next question concerns the deployment of the middleware solution itself. Here, the following three options have been considered:

- Custom-built platform;
- Microsoft Azure cloud;
- Amazon AWS.

Following, a short analysis on the advantages and disadvantages of each option are presented.

2.5.1 Custom-built platform

This option was soon eliminated as it requires a good IT support and infrastructure (internet access) without any restrictions and even though IT support was possible utilizing some partners' infrastructure (Honeywell, THN, Tecnalia), strict IT policies in all companies ruled out any deployment on their premises. Unlike for building

level middleware, this would require setting up a custom cloud based platform, which is a stretch goal considering the MOEEBIUS project scope.

2.5.2 Microsoft Azure

This is a viable option with no significant limitation; Azure however provides better support for Microsoft technologies.

2.5.3 Amazon AWS

Amazon was finally selected as the best option. It provides reliability, accessibility and pricing comparable to other commercially available platforms, but it also comes with good history of hosting WSO2 services — the latter implies rich community support.

2.6 Software and hardware

Software and hardware selection is exactly the same as for building level middleware, please refer to D4.4 for details.

2.7 Workflow

Workflow principles are exactly the same as for building level middleware, please refer to D4.4 for details.

3 WS02 analytic services

3.1.1 WSO2 Data Analytics Server (DAS)

WSO2 Data Analytics Server (WSO2 DAS) listens to a constant stream of events that represents the transactions and activities of an enterprise from different sources, processes them in real time and communicates the results in a variety of interfaces. This allows organisations to quickly respond to their environments, thus gaining an advantage over their competitors. In addition, WSO2 DAS combines real-time analytics with batch analytics (equipped with incremental processing), interactive and predictive (via machine learning) analysis of data into one integrated platform to support the multiple demands of Internet of Things (IoT) solutions, as well as mobile and Web apps. Table 2 lists features of WSO2 DAS.

Feature	**Description**
Communication	Possibility to create custom dashboards and gadgets that provide an at-a-glance view as well as an detail view. Detects conditions and generate realtime alerts and notifications (email, SMS, push notifications, physical sensor alarms etc.)

	Exposes event tables as an API via WSO2 API Manager and WSO2 Data Services Server.
Data aggregation	Receives data from event sources through Java agents (Thrift, Kafka, JMS), JavaScript clients (Web Sockets, REST), to IoT (MQTT), and also from WSO2 Enterprise Service Bus Connectors. Publishes events to one API for real-time, batch or interactive processing. Ability to access the analytics service via comprehensive REST API.
Extensibility using C-Apps	Industry/domain-specific toolboxes to extend the product for business use cases such as fraud detection, GIS data monitoring, activity monitoring etc. Ability to install C-Apps for each WSO2 middleware product, including the analytics functionality available with WSO2 API Manager.
High level language and data storage	Use of a structured easy to learn SQL-like query language. Develops complex real-time queries using SQL-like Siddhi query language. Scalable analytic querying using Spark SQL. Support for RDBMS (MSSQL, Oracle, MySQL) as data storages for low to medium scale enterprise deployments. Support for HBase and Cassandra as NoSQL storage for Big Data enterprise deployments.
Integrated, real-time, and batch analytics	Analyses both persisted and realtime data using a single product. Fast execution of batch programs using Apache Spark. Detects patterns (fraud detection) by correlating events from multiple data sources in real time using the high performing, open source WSO2 CEP engine powered by WSO2 Siddhi.
Interactive analytics and edge analytics	Searches for full text, complex query lookup, distributed indexing support using Apache Lucene for interactive analytics. Correlates/filters events at the edge for edge analytics.

Table 2: Features of WSO2 DAS

Figure 1 describes architecture of WSO2 DAS and Figure 2 illustrates simple event flow used in streaming analytics.

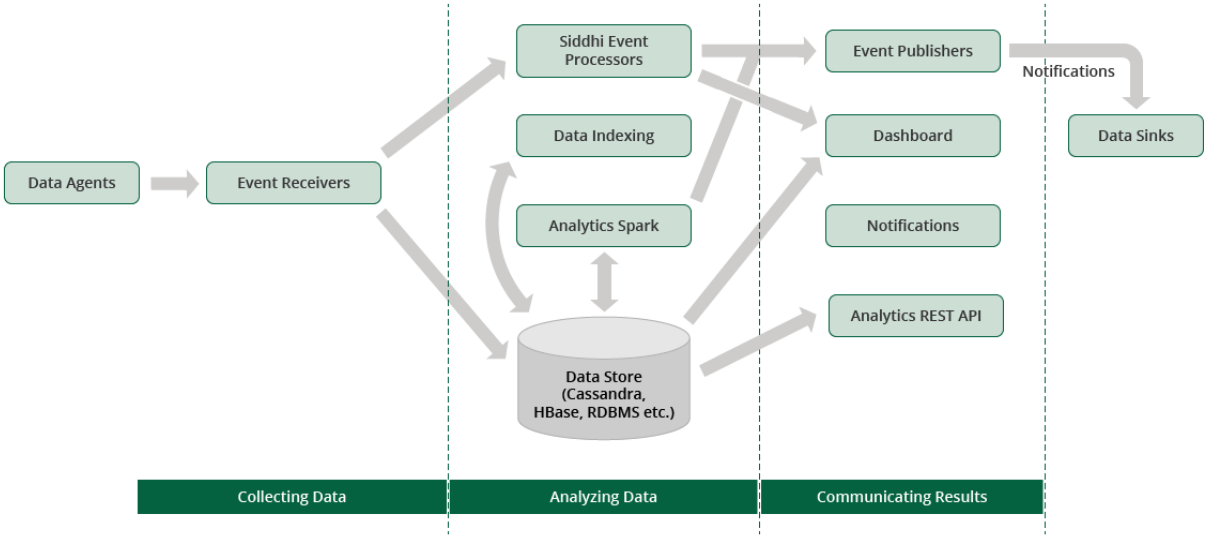

Figure 1: WSO2 DAS architecture.

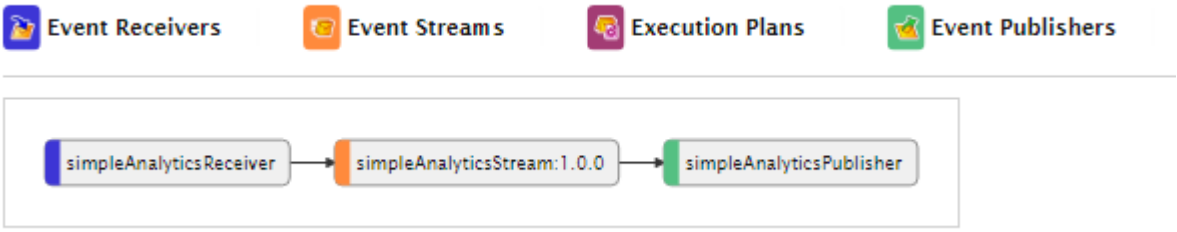

Figure 2: Example of simple event flow.

We have created a simple flow consisting of an Event Receiver, an Event Stream and an Event Publisher. This flow allows us to persist data coming from some specific sensors to a specialised DAS table. Custom scripts then can be set up to run against this table at specific time intervals given by a cron expression[1]. We also created a simple script which implements a moving averaging of the data for illustration purposes. Scrips will be set up for specific types of data points to aggregate the incoming data based on time, location, etc. These scripts can be executed daily or hourly to update tables containing aggregated values. Aggregated tables can be anonymised depending on our needs.

Tables containing aggregated data generated by the SparkSQL scripts are used for high level analytics of overall trends and total building energy consumption.

Follows example of a script used for calculation of averages of values from an analytics stream.

```
CREATE TEMPORARY TABLE avgTable USING CarbonAnalytics OPTIONS (tableName "simpleanalyticsstream");
```

[1] https://en.wikipedia.org/wiki/Cron

```
CREATE TEMPORARY TABLE avgTableResult USING CarbonAnalytics OPTIONS (tableName "avg_table_result", schema "pointname STRING, countryid STRING, buildingid STRING, avg_value DOUBLE");

INSERT OVERWRITE TABLE avgTableResult SELECT pointname, countryid, buildingid, avg(value) AS avg_value FROM avgTable GROUP BY pointname, countryid, buildingid;
```

3.1.2 WSO2 Business Rule Server (BRS)

WSO2 Business Rules Server is a lean, 100% open source, cloud-enabled Business Rule Management System (BRMS). It is offered under Apache Software License Version 2.0 which is one of the most business-friendly licenses available today. It provides capability for defining, deploying, monitoring, and maintaining an organization's business decisions, and exposing them as secure, reliable web services. Integration with WSO2 Enterprise Service Bus supports the incorporation of dynamic business decisions to your enterprise's application integration solution.

A BRMS is a software system used to define, deploy, execute, monitor and maintain the variety and complexity of decision logic that is used by operational systems within an organization or enterprise. This logic, also referred to as business rules, includes policies, requirements, and conditional statements that are used to determine the tactical actions that take place in applications and systems.

4 Selected MOEEBIUS-related use cases

This section aims to illustrate relevant MOEEBIUS use cases, especially with respect to the middleware layer – at district level. It does not aim to describe all considered use cases.

4.1 District level dynamic assessment framework for energy performance optimization

As one of the components of the district level assessment, the district Dynamic Assessment Engine will require of complex analytics implementation. The aim of the module is to calculate the different scenarios that ensuring each building's optimal management taking into account the business objectives as identified in WP2 (demand reduction for energy efficiency or Demand Response Strategies participation).

This is foreseen as purely as a combinational problem. The combinational type of problems is typical problems where all the power of data analytics can be exploited.

The combinational problems are NP-Complete problems; this means that their solution time doesn't follow polynomial time. There are several heuristic methods that solve this type of problems, Local Search or Genetic Algorithms are some of

them. Lineal Ordering Problem (LOP) and Number Partitioning problems are probably the most well-known problems of this type.

The MOEEBIUS modules involved in the current use case are:

- Demand Aggregation, Flexibility and Management Engine: as the "what if" simulation analysis components that facilitates the real time decision making by performing analytics over historical data.
- District Dynamic Assessment Engine: DAE's aim is to fine tune the energy demand/consumption forecasts done by the Building Performance Simulation Engine taking also into account the business scenarios and objectives defined by the external stakeholder (end user of the tool). DAE's activity can by split in two sub tasks.
    - Calibration: Executed offline, the calibration tunes district model parameters (energy consumption profiles, demand flexibility ratio…) in order to calculate correction factors related to them. *(training and fine-tuning towards allowing for enhanced demand/ supply predictions, monitoring of district heating status)*
    - Assessment: Once the model is calibrated the real time district operation is ensured → during the forecasting phase the "Assessment" task applies the appropriate correction factor.

Considering the high level functionality of this use case, towards real time optimization of buildings portfolio by taking into account business objectives, we define the role of District Level middleware as the software component that interfaces with:

- DIM server: Deployed over WSO2 DSS stores district topology repository and enables read/write operations.
- Pilot data server: Deployed over Cassandra framework stores the aggregated values gathered from pilot sites. Building level data from Building Middleware are available at district level for further analytics. The values stored are directly linked to the ones defined in the DIM server
- Building Performance Simulation Engine: Taking into consideration the BIM calculates the building level energy demand/consumption forecast in an aggregated way, making this information available at district level.

On the other hand, and following the monitoring functionalities, the District Level Middleware component enables the implementation of automated control actions deriving from the continuous evaluation of alternative demand response strategies. From a control facilitation point of view, the middleware enables the dispatch of optimum optimized control signals to the corresponding devices and systems. This information will be then communicated back to building-level and independent control systems.

The figure below describes the simplified version of the DAaF (Dynamica Aggregation and Flexibility) engine's activity diagram.

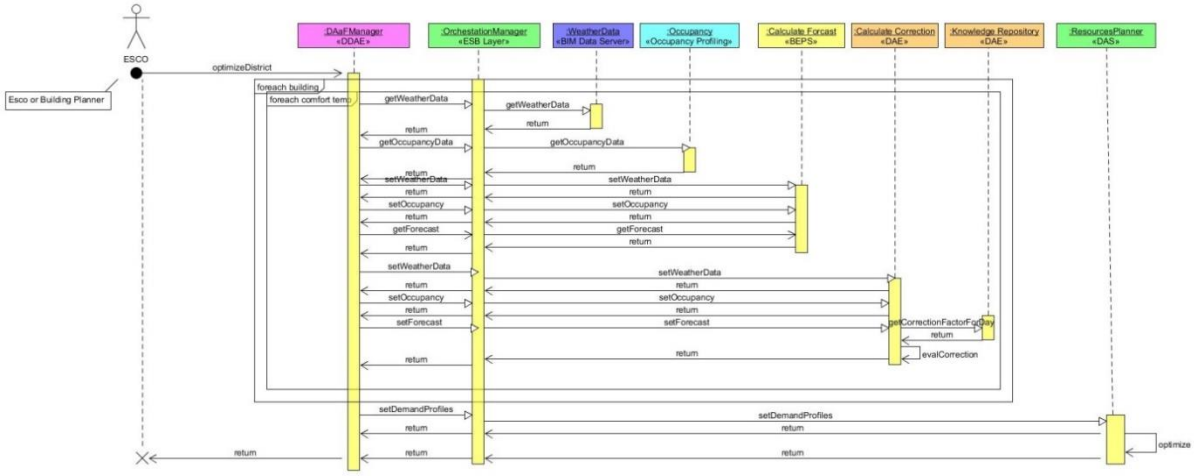

Figure 3: Simplified Dynamic Aggregation and Flexibility Engine activity diagram

The Resource Planner implemented in the WSO2 DAS (Data Analytic Server) will have as inputs a set of consumption profiles (n curves) delivered for a set of buildings (m buildings), this is in total $n_x m$ curves. Taking into consideration all the set of input curves the Resource Planner will find the subset of them (one per building) that minimizes the overall energy demand.

4.2 Visual Analytics on consumers' historical data

Apart from real time monitoring and control in tertiary building, one of the main objectives of the district level tool is to enable performing analytics over energy consumption and generation data addressing also contextual parameters of various consumers. Multi-parameter criteria analysis algorithms and tools will be utilized by a Data Analytics component which will provide visualization and interaction mechanisms to the Aggregators and ESCOs for multidimensional analysis, correlation and efficient management of prosumer profiles and prosumer flexibility. Therefore, an analytics tool is a main prerequisite for the business stakeholders towards facilitating the optimization of portfolio management.

It is clear that the analysis is performed over the streams of data retrieved from District Level Middleware, and thus we are highlighting the role of this component as part of the use case description. The MOEEBIUS modules involved in the current use case are:

- Demand Aggregation, Flexibility and Management Engine: as the analytics component that simulates the operation of the portfolio under different conditions and further facilitates the Aggregators and ESCOs towards the selection of the best fitted high level strategies.

Considering the high level functionality of this use case towards providing analytics over historical data retrieved from the different district end points, we define the role of District Level Middleware on accessing the aggregated information required for analysis:

- DIM server: Deployed over WSO2 DSS stores district topology repository and enables read/write operations.
- Pilot data server: Deployed over Cassandra framework stores the aggregated values gathered from pilot sites. Building level data from Building Middleware are available at district level for further analytics. The values stored are directly linked to the ones defined in the DIM server
- Building Performance Simulation Engine: Taking into consideration the BIM calculates the building level energy demand/consumption forecast in an aggregated way, making this information available at district level.

The figure below describes the simplified version of the DAFM (Demand Aggregation, Flexibility and Management Engine) engine's activity diagram as the core component of this use case.

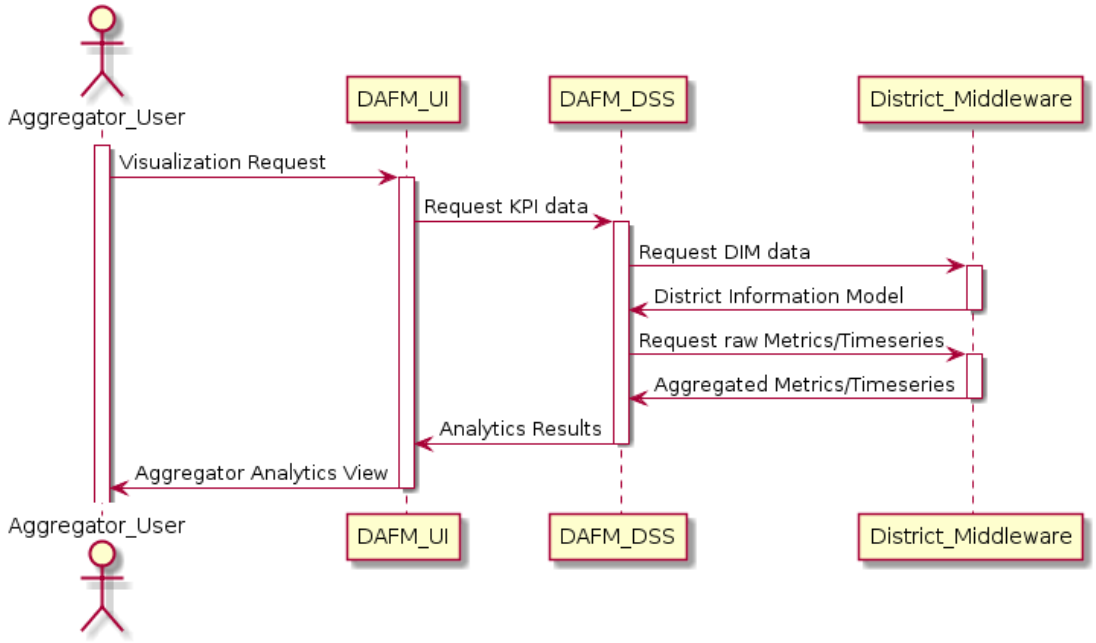

Figure 4: DAFM engine simplified activity diagram

The end user of the tool (DR Aggregator) interacts with the DAFM Engine via the associated GUI, setting the parameters for the analytics process.

The core engine of the tool (DAFM_DSS) triggers a request to District Level Middleware for retrieving raw streams of metrics time series required for the analysis.

The data time series are further associated and become semantically enhanced taking into account the District Information Model elements as defined in the DIM server (Internal component of District Level Middleware)

The static DIM information is also available to DAFM engine for further analysis.

The role of DAFM_DSS is to perform different types of analytics over the raw streams of data towards the extraction of knowledge that is useful for DR Aggregators as the main business stakeholders of the tool.

This processed information become available to the end users in a visually appealing way, by incorporating different visualization widgets to set the front end view of the DAFM platform (DAFM_UI).

The same workflow analysis is defined also for UC-06 – Hypothesis analysis for the optimal management of consumers on DSM strategies and thus the role of District Level Middleware remains the same.

It is clear from use case analysis that the core functionality of MOEEBIUS District Level Middleware remains the same, acting at the layer for retrieving information from the different end points (at district level) and providing a seamless interface layer for the business application at District Level Analysis Tools defined in the project.

5 Conclusion

After evaluation of the alternatives, the WSO2 platform was selected as a framework for the middleware, Amazon Web Services was chosen for hosting the middleware platform and Cassandra was selected as the data storage. We have registered MOEEBIUS Middleware AWS account and instantiated three EC2 instances for running required WSO2 components and Cassandra database cluster. Cassandra cluster was set up by interconnecting Cassandra databases on all three instances and tables for sensor data were created with parameters based on discussion with partners. Individual WSO2 components were deployed on the EC2 instances. Inbound and outbound ports were configured to allow communication with the middleware from outside of the cloud.

WSO2 Data Services Server (DSS) was configured to be able to access the Cassandra data store. For this purpose, a data service CassandraService was created and configured with Cassandra cluster details. Endpoints for reception of requests for data storage and retrieval were created. Currently there are endpoints which return either all data points or endpoints which expect query parameters constraining the scope of the data to be returned.

Several general APIs were created in WSO2 Enterprise Service Bus (ESB) to accommodate incoming requests for data input or data querying. These APIs contain sequences with optional conditional logic which can route or transform received messages based on their content. This allows us to truly separate the concerns of data acquisition, processing and analytics since we have the ability to dynamically change the shape and the destination of the messages according to our needs. Addition of new APIs, sequences and proxies is a live process. We can seamlessly implement new interface elements tailored to the needs of the new MOEEBIUS components.

WSO2 Data Analytics Server (DAS) was prepared for use by creating basic event receiver, event stream and event publisher. This event flow is needed to enable real-time streaming analytics. Data for streaming analytics can be routed to WSO2 DAS by WSO2 ESB according to some conditional logic. This also allows us to perform batch analytics, e.g. aggregation, on data stored in temporary tables. Results of this aggregation can be used to read total energy consumption for given building, whole district, etc.

Some partners already connected their services to the middleware and are sending data which are stored in the Cassandra data store. By the time of writing there is over 500,000 records in the database.

Future steps for the middleware development are linked with the development of other MOEEBIUS components. We will be cooperating with other partners developing the MOEEBIUS components to create tailored interfaces for these components to connect to the middleware and orchestrate the subsequent calls

with minimal effort. This documentation will evolve as new MOEEBIUS components will be connected to the middleware and new APIs will be deployed to the WSO2 ESB.

References

[1]. MOEEBIUS Description of Actions, MOEEBIUS Project (680517)
[2]. MOEEBIUS D2.1 End-user & business requirements, MOEEBIUS Project
[3]. MOEEBIUS D2.2 New Business Models and Associated Energy Management Strategies, MOEEBIUS Project
[4]. MOEEBIUS D2.3 MOEEBIUS Energy Performance Assessment Methodology, MOEEBIUS Project
[5]. MOEEBIUS D2.4 Functional and Non-functional requirements of the MOEEBIUS framework and individual components
[6]. MOEEBIUS D3.1 Framework Architecture including functional, technical and communication specifications
[7]. OPENSOURCE, MICROSERVICES [Online]. Available: https://opensource.com/resources/what-are-microservices
[8]. Fowler, A., MICROSERVICE [Online]. Available: https://martinfowler.com/articles/microservices.html
[9]. WSO2 [Online]. Available: http://wso2.com/
[10]. Martin, S., Hernandez, J., & Valmaseda, C. (2015, March). A novel middleware for smart grid data exchange towards the energy efficiency in buildings. In Networked Systems (NetSys), 2015 International Conference and Workshops on (pp. 1-8). IEEE.
[11]. Jahn, M., Eisenhauer, M., Serban, R., Salden, A., & Stam, A. (2012, July). Towards a context control model for simulation and optimization of energy performance in buildings. In 3rd Workshop on eeBuildings Data Models, Proc. of ECPPM conference, Reykjavik, Iceland.
[12]. Le Guilly, T., Skou, A., Olsen, P., Madsen, P. P., Albano, M., Ferreira, L. L.,... & Gangolells, M. (2016, September). ENCOURAGEing results on ICT for energy efficient buildings. In Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on (pp. 1-8). IEEE.
[13]. Hohpe, G., & Woolf, B. (2004). Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional.