A novel smart grid architecture that facilitates high RES penetration through innovative markets towards efficient interaction between advanced electricity grid management and intelligent stakeholders

H2020-GA-863876

Data Model of FLEXGRID architecture

Deliverable D6.1

Document Information

Scheduled delivery	31.03.2021
Actual delivery	24.03.2021
Version	Final
Responsible Partner	ICCS

Dissemination Level

PU Public

Contributors

Prodromos Makris (ICCS), Dimitrios Vergados (ICCS), Konstantinos Steriotis (ICCS), Nikolaos Efthymiopoulos (ICCS), German Martinez (ETRA), Elena Leal Lorente (ETRA), Maria-Iro Baka (UCY), Christina Papadimitriou (UCY), Marios Kynigos (UCY), Stylianos Loizidis (UCY), Andreas Kyprianou (UCY), George Georghiou (UCY), Domagoj Badanjak (UNIZG-FER), Hrvoje Pandzic (UNIZG-FER), Lars Finn Herre (DTU), Elea Marie Prat (DTU), Spyros Chatzivasileiadis (DTU), Robert Gerhcke (NPC), Matin Bagherpour (NPC), Gesa Milzer (NODES), Tonci Tadin (HOPS), Malte Thoma (bnNETZE)

Internal Reviewers

Elena Leal (ETRA), German Martinez (ETRA), Emmanouel Varvarigos (ICCS)

Acknowledgements

The research leading to these results has received funding from the EC Framework Programme HORIZON2020/2014-2020 under grant agreement n° 863876.

Glossary of Acronyms

Project management terminology

Acronym	Definition
D	Deliverable
HLUC	High Level Use Case
MS	Milestone
TRL	Technology Readiness Level
UCS	Use Case Scenario
WP	Work Package

Technical terminology

Acronym	Definition
AC-OPF	Alternating Current Optimal Power Flow
AFAT	Automated Flexibility Aggregation Toolkit
API	Application Programming Interface
ATP	Automated Trading Platform
B2B/B2C	Business to Business / Business to Consumer
BM	Balancing Market
BRNN	Bayesian Regularized Neural Network
BRP	Balance Responsible Party
B-RTP	Behavioral Real Time Pricing
BSU	Battery Storage Unit
CAPEX	Capital Expenditures
DAD	Day Ahead Dispatch
DA-EM	Day Ahead Energy Market
DA-RM	Day Ahead Reserve Market
DB	Data Base
DC-OPF	Direct Current Optimal Power Flow
DER	Distributed Energy Resource
DFA	Distributed Flexibility Asset
DLEM	Distribution Level Energy Market
DLFM	Distribution Level Flexibility Market
DR	Demand Response
DSM	Demand Side Management
DSO/TSO	Distribution/Transmission System Operator
ELM	Extreme Learning Machine
ESP	Energy Service Provider
FMCT	Flexibility Market Clearing Toolkit
FMO	Flexibility Market Operator
FSP	Flexibility Service Provider
FST	FlexSupplier's Toolkit
GUI	Graphical User Interface
ICT	Information and Communication Technology

I-DLFM	Interactive Distribution Level Flexibility Market
KPI	Key Performance Indicator
LMP	Locational Marginal Price
MO	Market Operator
MTU	Market Time Unit
NWP	Numerical Weather Prediction
OPEX	Operational Expenditures
OPF	Optimal Power Flow
PCC	Point of Common Coupling
P-DLFM	Proactive Distribution Level Flexibility Market
RPC	RES Production Curve
R-DLFM	Reactive Distribution Level Flexibility Market
RES	Renewable Energy Sources
SLFN	Single Layer Feed Forward Network
SoC	State of Charge
SOCP	Second Order Cone Programming
S/W	Software
TER	Transmission level Energy Resource
VPP	Virtual Power Plant

Table of Contents

List of Figures and Tables

List of Figures

List of Tables

Document History

This deliverable includes the research output of task 6.1. It includes a detailed data model for the S/W development of FLEXGRID S/W platform. High-level description of the interaction with existing S/W platforms is also provided.

Table 1: Document History Summary

Revision Date	File version	Summary of Changes
30/11/2020	v0.1	Draft ToC circulated within all consortium partners
07/01/2021	v0.2	All partners commented on the draft ToC structure.
13/01/2021	v0.3	Final ToC version has been agreed and writing task delegations have been provided to all involved partners.
26/02/2021	v0.4	All partners contributed their 1st round inputs and first draft version has been reviewed by ETRA.
03/03/2021	v0.8	ICCS integrates text, addressed comments from all research partners and produced the pre-final version for internal review.
17/03/2021	v0.9	ETRA reviewed the pre-final version and provided comments for changes/enhancements.
22/03/2021	v0.95	ICCS addressed all comments from the internal review process and forwarded the final version to the coordinator.
24/03/2021	v1.0	Coordinator (ICCS) made final enhancements/changes and submitted to ECAS portal

Executive Summary

This report is an official deliverable of H2020-GA-863876 FLEXGRID project dealing with all the data modelling work of FLEXGRID's S/W architecture. It includes the research output of task 6.1 and thus a detailed data model, which is a pre-requisite for the development of FLEXGRID S/W platform.

D6.1 elaborates on almost all previous FLEXGRID deliverables[1] as follows:

- D2.1: For all High Level Use Cases (HLUC) and Use Case Scenarios (UCS) that have been defined in D2.1, this report specifies the exact data model that will be used in order to fulfill the system-level and user requirements that have been analyzed in the first months of the project.
- D2.2: The FLEXGRID data model has been developed following up all the major architectural decisions that have been made and have been documented in D2.2. More specifically, this report elaborates on the draft data model for AFAT, FST and FMCT algorithms that has already been described in a quite abstract way in chapter 7 of D2.2.
- D3.1: Based on chapter 6 of D3.1 and the description of the WP3 mathematical models and algorithms, this report provides the final data model for the interaction between the AFAT's frontend (i.e. ATP GUI) and backend services.
- D4.1: Based on chapter 8 of D4.1 and the description of the WP4 mathematical models and algorithms, this report provides the final data model for the interaction between the FST's frontend and backend services.
- D5.1: Based on chapter 5 of D5.1 and the description of the WP5 mathematical models and algorithms, this report provides the final data model for the interaction between the FMCT's frontend and backend services.
- D8.1: The final data model that is presented in this report has been developed in a way that is totally in line with the initial market analysis, business modeling and the long list of value propositions presented in chapters 1 and 2 of D8.1.

Chapter 1 of this report describes the main steps of the S/W implementation methodology that has been followed by the entire consortium as well as the interactions with the research methodology that has been adopted at the early stages of the project. FLEXGRID S/W architecture is modular-by-design providing thus flexibility and means for efficient collaborative work and exploitation after the end of project's lifetime. A subset of the most important UCS have been selected to be integrated in the FLEXGRID ATP (TRL 5), while the residual ones are expected to be developed until TRL 3 within WP3, WP4 and WP5 and respective high-quality scientific papers are expected to be published in prestigious scientific journals and conferences. Moreover, **a subset of the most important functionalities per selected UCS have been chosen to be integrated in FLEXGRID ATP, because the focus of WP6 work is not on scientific excellence (like in WP3, WP4 and WP5), but on the potential impact, meaning the demonstration of proof-of-concept results in a real-life S/W platform, which can be used directly by all interested market stakeholders and will also boost the FLEXGRID's communication, dissemination and exploitation activities**.

[1] https://flexgrid-project.eu/deliverables.html

Chapter 2 presents detailed sequence diagrams for the proposed x-DLFM architectures, which are actually holistic energy market architectures that integrate the novel concept of "Distribution Level Flexibility Market - DLFM". The goal of these sequence diagrams is to depict all existing markets and how these may interact with the proposed DLFM and also how the different sequence of markets affects the performance of certain KPIs. **In each sequence diagram, various building blocks are illustrated, which represent the advanced mathematical models and algorithms that have been developed within WP3, WP4 and WP5 and will be integrated in FLEXGRID ATP (WP6). Moreover, exchange of information is illustrated between the various building blocks and thus among the various market stakeholders. For each one of these arrows and building blocks, detailed data models are provided in chapters 3-5.**

In Chapter 3, we present the detailed data model for the FMO and DSO users based on the WP5 research work. **All algorithmic inputs and outputs are described for UCS 1.1, 1.2 and 1.3 together with UML and sequence diagrams**. Based on this data modelling work provided by DTU, ETRA will implement the FMCT frontend and respective APIs for the communication between the FLEXGRID ATP, the central database and the FMCT backend (i.e. algorithms).

Chapter 4 presents the detailed data model for the ESP user based on the WP4 research work. **All algorithmic inputs and outputs are described for UCS 2.1, 2.2, 2.3 and 4.4 together with UML and sequence diagrams**. Based on this data modelling work provided by UNIZG, ICCS and UCY, ETRA will implement the FST frontend and respective APIs for the communication between the FLEXGRID ATP, the central database and the FST backend.

Chapter 5 presents the detailed data model for the aggregator user based on the WP3 research work. **All algorithmic inputs and outputs are described for UCS 4.1, 4.2 and 4.3 together with UML and sequence diagrams**. Based on this data modelling work provided by UCY and ICCS, ETRA will implement the AFAT frontend and respective APIs for the communication between the FLEXGRID ATP, the central database and the AFAT backend.

Conclusively, in the following months, FLEXGRID consortium will **elaborate on the data modeling work to deploy the FMCT/FST/AFAT frontends in FLEXGRID ATP as well as the Application Programming Interfaces (APIs) for the integration of the respective backend intelligence into the FLEXGRID ATP**. Finally, the central database will be developed, populated with real-life historical data together with the APIs for the information exchange between the database and FLEXGRID ATP frontend and backend modules. Finally, it should be noted that although the data modeling work has finished in M18, an iterative S/W development process will be followed and thus the final data model may be slightly different from the one reported in this deliverable. Therefore, the final FLEXGRID data model will be delivered at the end of the project's lifetime via D6.3.

1 Introduction

1.1 Purpose of the document

This report aims at defining the detailed data model of FLEXGRID architecture. **With the term "data model", we mean the exact information exchange and communication protocols based on which the various market stakeholders interact with each other via the proposed FLEXGRID Automated Trading Platform (ATP).** We also mean the algorithmic inputs and outputs of all FLEXGRID processes, which aims at optimizing the benefits/interests of each market stakeholder or the system as a whole.

We have made a clear categorization of the various FLEXGRID data models throughout the report. One major categorization is between the proposed x-DLFM architectures. We propose three architectures, namely: i) Reactive DLFM (R-DLFM), ii) Proactive DLFM (P-DLFM), and iii) Interactive DLFM (I-DLFM). We also assume the "No-DLFM" architecture as a benchmark in the sense that no DLFM exists in today's EU energy markets. R-DLFM takes place after day-ahead energy (DA-EM) and day-ahead reserve markets (DA-RM) operated by the Market Operator (MO) and Transmission System Operator (TSO) respectively. P-DLFM takes place before the aforementioned existing markets in order to deal proactively with possible distribution-level congestion management and voltage control problems. Finally, I-DLFM assumes an iterative information exchange process between the MO and FMO (i.e. energy markets) and the TSO and DSO (i.e. reserve and balancing markets).

Another major categorization is between the FMO, DSO, ESP and aggregator users. In particular, **we provide a detailed data model for each one of the aforementioned market stakeholders**. For the FMO and DSO users, there is a data model for the FMCT frontend (i.e. what the FMO/DSO users can fill in in their GUIs in order to perform their market and business logic operations) and the FMCT backend (i.e. what the FMO/DSO users can visualize as algorithmic results in order to make further managerial actions). Following the same rationale, there is a data model for the ESP user and the aggregator user (cf. chapters 4 and 5 respectively).

1.2 Scope of the document

This document presents the results of the FLEXGRID's data modelling work in the context of Task 6.1, which took place between M13 and M18. As shown in the figure below, task 6.1 work has been based on previous work and respective deliverables that took place within the first twelve months of the project's lifetime as follows:

- D6.1 elaborates on the high-level specifications and draft data modeling work that took place within WP2 as well as the business cases that have also been defined at the early stages of the project.
- Detailed data models are also defined for all the mathematical models and algorithmic solutions that have been defined in the first phase of research work within WP3, WP4 and WP5. In other words, all research partners (i.e. ICCS, UCY, DTU, UNIZG-FER) have identified the most important UCS functionalities that will be integrated in the FLEXGRID ATP and collaborated closely with ETRA in order to design all the GUIs

(i.e. ATP frontend services) as well as the APIs for the exchange of data with ATP backend services.

- Finally, data modeling work was based on the FLEXGRID's data management plan (DMP), which includes all types of real-life and historical datasets that will be available by FLEXGRID's industrial partners (i.e. NODES, NPC, BNNETZE, HOPS, SIN) in order to test and validate the proposed mathematical models and algorithms. Moreover, based on the industrial partners' expertise, an initial market analysis and business modeling work took place that gave important feedback to the research partners towards defining the final version of the data models.

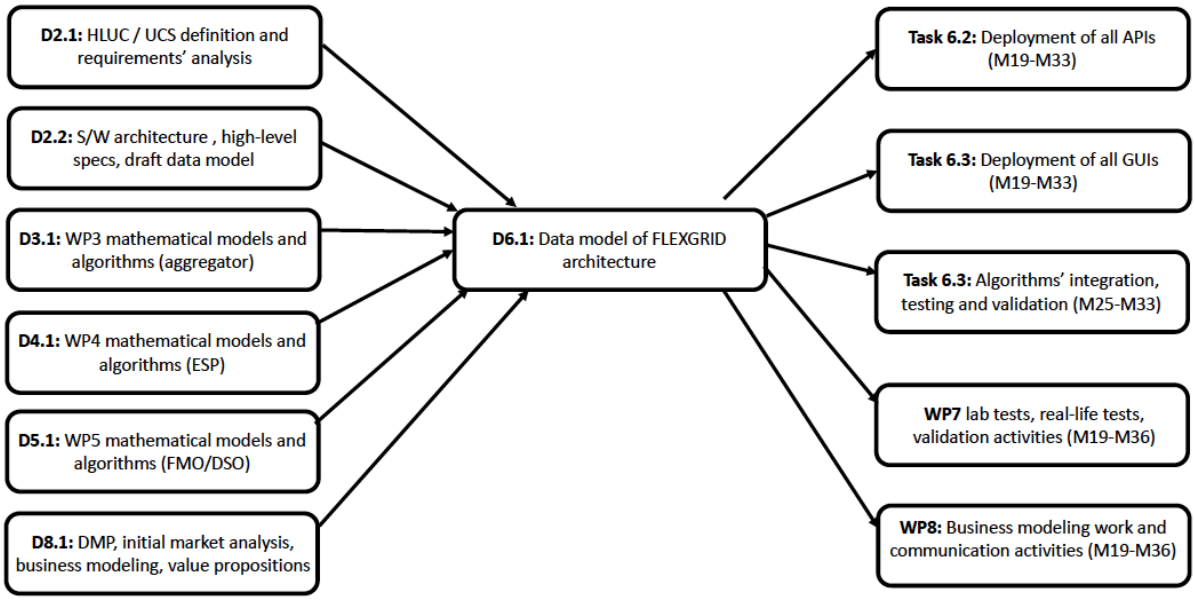

Figure 1: Placement of data modeling work within FLEXGRID project's context

After the end of Task 6.1 and delivery of D6.1 in M18, a set of specific activities will take place in order to elaborate on data modeling work's results as follows:

- As all data models will be translated in json format and respective swagger[2] files will be ready for use, the deployment of all APIs and GUIs will start in M19.
- Another major task will be the integration of all selected algorithms in the FLEXGRID ATP as well as the testing and validation activities that will take place during the Period 2.
- Based on the data modeling work presented in chapter 2 of this report, various holistic FLEXGRID energy market architectures will be developed and compared within the lab testing work of WP7.
- Finally, as all data models are organized in swagger files and are also available in FLEXGRID GitHub area[3], the consortium's communication activities will be more targeted and efficient towards identifying interesting real-life business cases in EU area with targeted customer segments according to the ongoing FLEXGRID's business modeling work, too.

[2] Swagger is a set of open source software tools for designing, building, documenting and using RESTful web services - https://swagger.io/
[3] https://github.com/FlexGrid

1.3 Research and S/W implementation methodology

FLEXGRID research partners have already defined a clear research methodology plan in Task 2.1. ETRA has also made a clear S/W implementation plan in Task 2.4, which is based on a modular-by-design approach. In the figure below, the data modelling methodology is illustrated. Five main steps were followed by all research partners in close collaboration with ETRA, which is responsible for the S/W integration and FLEXGRID ATP frontend services (i.e. GUIs). These five steps were followed for each individual UCS and thus proposed mathematical model and algorithm solution as follows:

- **Step 1:** For every UCS that has been short-listed to be integrated in FLEXGRID ATP, the algorithmic solution is clarified in order to fit the business needs of a real-life market stakeholder (i.e. FMO, DSO, ESP, aggregator) that will utilize the FLEXGRID ATP. This means that some assumptions need to be made in order to transform the rather complex mathematical formulations of the research WPs (i.e. WP3, WP4 and WP5) into more realistic and closer to business- and regulation-related constraints.
- **Step 2:** The algorithmic inputs are defined in great detail after consultation with industrial partners. Moreover, ETRA collaborated closely with research partners in order to design the FLEXGRID ATP frontend (GUI) services in a user-friendly manner.
- **Step 3:** The algorithmic outputs (i.e. results) are defined in great detail after consultation with industrial partners. Moreover, ETRA collaborated closely with research partners in order to illustrate interesting views to the users that will actually help them in their real-life business.
- **Step 4:** After the initial GUI designs and user views have been agreed, respective UML and sequence diagrams were designed in order to facilitate the S/W implementation at a later stage. The UML diagrams (one for the algorithmic inputs and another one for algorithmic outputs per selected UCS) will help in the development of the central FLEXGRID database. The sequence diagram (one per HLUC) will help in the development of the APIs and the exact information exchange between the FLEXGRID's database, frontend and backend services.
- **Step 5:** In the last step of the data modelling work, the RESTful APIs have been developed. This means that the raw data models (i.e. in the form of tables as shown in chapters 3-5 below) have been transformed into a json format that can be easily read by REST API servers and clients. These RESTful APIs are also available in .yml files, so that it can be easily visualized and potentially used by interested developers via online swagger editors (https://editor.swagger.io/).

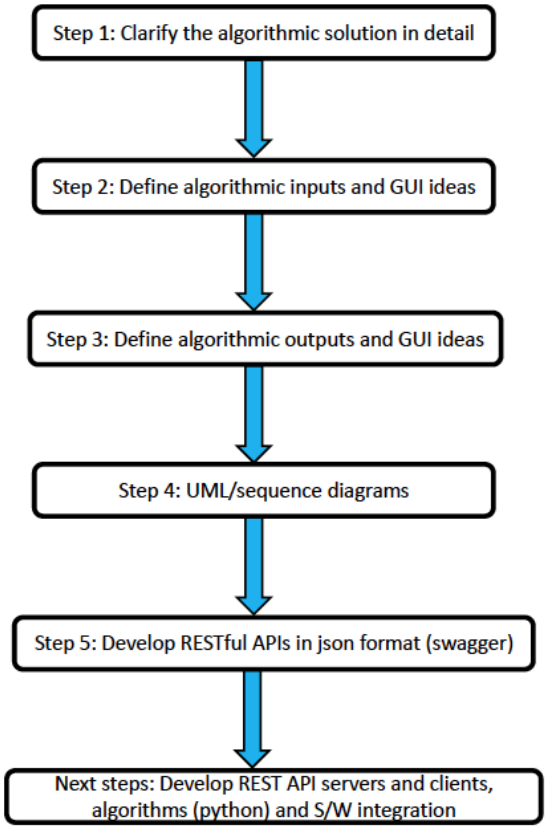

Figure 2: Data modelling methodology

On top of all this data modeling work, there is a holistic energy market architecture based on which the various market stakeholders interact with each other via the proposed FLEXGRID ATP. The current EU regulatory framework adopts the "no-DLFM" architecture and thus FLEXGRID uses it as a benchmark. This means that nowadays there is no real-life Distribution Level Flexibility Market (DLFM) in Europe. According to FLEXGRID, we introduce the Reactive DLFM (R-DLFM) architecture, which is quite close and compatible with the existing regulation. **Within the WP6 context, FLEXGRID develops the ATP taking into consideration the R-DLFM model. However, FLEXGRID ATP could also support other energy market architectures like P-DLFM and I-DLFM in the future**. Extensive simulation and emulation tests in a laboratory environment (i.e. TRL 4) that will compare the various x-DLFM architectures will take place in the context of WP7.

More details about the data model of the various x-DLFM architectures are provided in chapter 2 of the current report. Chapters 3-5 describe the data model for each one of the selected UCS (or else algorithms and functionalities that will be integrated in FLEXGRID ATP). **As a first S/W integration step, FLEXGRID will integrate the first algorithm (UCS 2.3) and have a live demonstration during the Period 1 Review meeting. Right after, all other UCS will be integrated following a similar S/W integration plan.** More technical details about this S/W integration process are provided in D6.2 (M18).

2 FLEXGRID x-DLFM architectures

As already mentioned, in WP6 work (i.e. TRL 5), we focus on the Reactive Distribution Level Flexibility Market (DLFM) architecture, which is the most compatible with the today's EU regulatory framework. However, in the next chapters, it is also explained under which assumptions could FLEXGRID algorithms be applicable for all x-DLFM architecture variants. **Before analyzing the data model per UCS, it is important to describe how all the proposed FLEXGRID mathematical models and algorithms are placed in a single energy market architecture. For example, the timing (i.e. when?) that each algorithm runs, within which energy market context it runs and what is the sequence of markets that is being assumed are important considerations that directly affect the data model.**

In the following sections, we provide detailed sequence diagrams that explain the following holistic energy market architectures:

- No-DLFM architecture
- R-DLFM architecture
- P-DLFM architecture
- I-DLFM architecture

2.1 No-DLFM architecture - benchmark

In section 2.4 of D5.1[4], we provided an extensive summary of the proposed FLEXGRID x-DLFM architectures. Leveraging on the NPC's expertise in energy markets' design, as Nord Pool is one of the most prestigious Market Operators (MOs) in Europe, we have made the following realistic assumptions about the organization of FLEXGRID x-DLFM architectures:

- The Market Operator - MO (e.g. Nord Pool) operates day-ahead and intra-day energy markets at the transmission network (TN) level.
- The Flexibility Market Operator – FMO (e.g. NODES) operates day-ahead and intra-day energy markets at the distribution network (DN) level. This entity may also be called Local Market Operator (LMO).
- The Transmission System Operator – TSO operates the day-ahead reserve and balancing energy markets at the TN level.
- The Distribution System Operator – DSO operates the day-ahead reserve and balancing energy markets at the DN level.

Within the FLEXGRID project, we assume the sequence of the 3 following markets: i) day-ahead energy market, ii) day-ahead reserve market, and iii) near-real-time balancing energy market. Finally, we assume that this sequence of 3 markets may also take place for the distribution network level, too.

In the sequence diagram below, the baseline architecture that represents the today's regulatory framework (i.e. without any DLFM) is illustrated. In the horizontal axis, all basic energy market stakeholders are depicted, namely:

- Market Operator (MO)

[4] https://flexgrid-project.eu/assets/deliverables/FLEXGRID_D5.1_final_03122020.pdf

- Transmission System Operator (TSO)
- Energy Service Provider (ESP) that uses the FLEXGRID's FST services
- Aggregator that uses the FLEXGRID's AFAT services
- Flexibility Market Operator (FMO) that uses the FLEXGRID's FMCT services[5]
- Distribution System Operator (DSO) that uses the FLEXGRID's FMCT services

In the vertical axis, the temporal sequence of markets is illustrated. For example, in the no-DLFM architecture, where there exist no distribution-level markets, we assume 3 main markets, while in Reactive DLFM architecture, we assume one more market (i.e. DLFM), which takes place after the day-ahead energy and reserve markets and before the near-real-time balancing market.

There are also several colored boxes, which represent the FLEXGRID mathematical models and algorithmic solutions, whose data models are described in detail in chapters 3-5 of this document. More specifically, orange boxes represent WP3 algorithms (i.e. aggregator-related), blue boxes represent WP4 algorithms (i.e. ESP-related) and purple boxes represent WP5 algorithms (i.e. related with FMO and DSO). There are also several black boxes that represent algorithms and processes that are out of FLEXGRID's scope. This means that FLEXGRID does not make any novel scientific contributions in these processes, but we rather assume some state-of-the-art implementations in order to develop the holistic energy market architecture. The dotted arrows represent the results from one process, which are communicated to another market actor in order to serve as an input to another process.

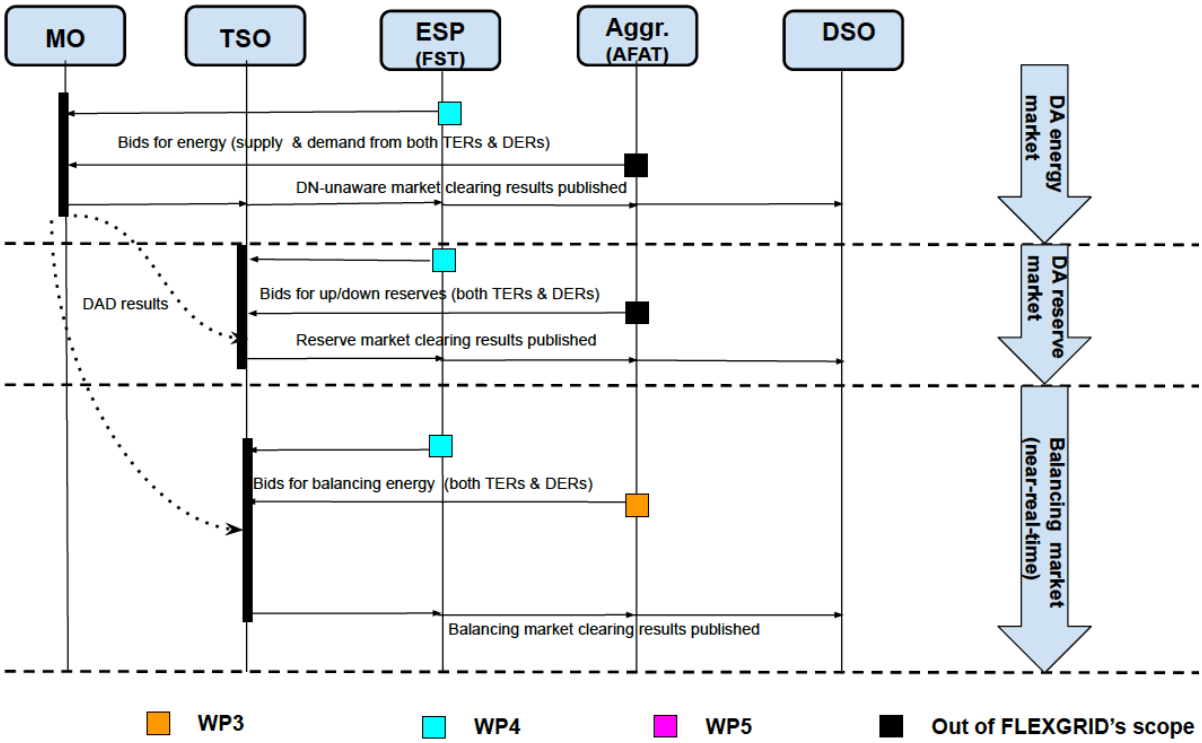

Figure 3: No-DLFM architecture representing the today's EU regulatory framework

[5] FMO is not present in the no-DLFM architecture. It is a new market entity that is introduced within FLEXGRID project.

As shown in Figure 3, the three blue boxes represent the mathematical model and algorithmic solution developed within UCS 2.3, whose data model is extensively analyzed in section 4.3 below[6]. These blue boxes also involve the market price forecasting algorithms that are developed within UCS 4.4, whose data model is extensively analyzed in section 4.4. The orange box of Figure 3 represents the mathematical model and algorithmic solution developed within UCS 4.3, whose data model is extensively analyzed in section 5.2.

Within WP6, we will integrate the above-mentioned algorithms. The ESP and aggregator users will be able to fill in input parameters and configure a new simulation scenario in the FST/AFAT frontend (GUI), the respective algorithm will run in the FST/AFAT backend and finally the results will be visualized by the user in the FLEXGRID ATP.

2.2 Reactive DLFM (R-DLFM) architecture

As already mentioned, the proposed Reactive DLFM architecture is compatible with the existing EU regulatory framework. This is the reason why we decided to implement it until TRL 5 via the deployment of FLEXGRID ATP. The respective data models of the short-listed UCS are extensively described in chapters 3-5.

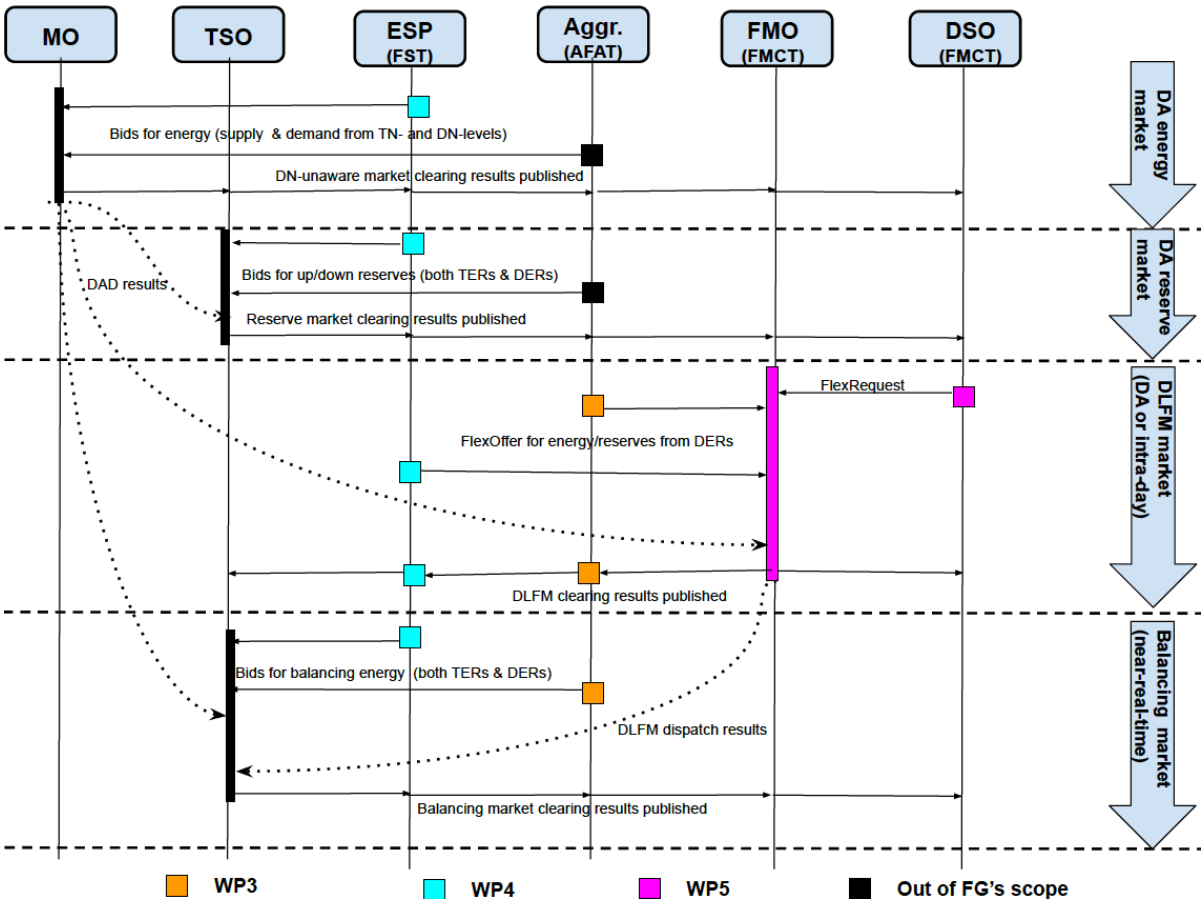

Figure 4: Reactive DLFM architecture (DLFM follows DA-EM and DA-RM)

[6] In no-DLFM architecture, the ESP participates in the three existing markets, while in R-DLFM the ESP can also participate in the new DLFM introduced by FLEXGRID.

In Figure 4, the WP5 processes are also illustrated. The purple box represents the algorithm that dynamically generates a FlexRequest for the DSO user. This FlexRequest is a price vs. quantity curve for a given timeframe and location that represents the various price/quantity tuples that the DSO requests for a flexibility service in order to be able to deal with imminent local congestion and voltage control issues at a specific geographical location of its distribution network. The elongated purple box represents the market clearing process (auction-based or pay-as-bid) that should be run by the FMO. This market clearing process tries to match the DSO's FlexRequests with the FlexOffers made by the ESPs and aggregators. This can be done via two main algorithms, namely: i) auction-based market clearing (i.e. the algorithm runs once after the gate closure), and ii) pay-as-bid continuous market clearing[7]. We also assume that day-ahead dispatch (DAD) results have already been published by the MO, so the energy "positions" of all players are known and thus are used as inputs to the FMO's network-aware market clearing algorithm. Another important assumption is that the DLFM clearing results are used as input for the clearing of the near-real-time balancing market operated by the TSO. This implicitly means that the ESPs/aggregators will have to pay/get paid for the possible imbalances that they incur due to the change of their day-ahead energy "positions", because of the fact that they had to provide their flexibility to the DSO. Extensive details about the exact data model of the various market clearing algorithms for the various flexibility products (i.e. energy, active power reserve, reactive power reserve) are presented in sections 3.1, 3.2 and 3.3 below.

There are also four blue boxes that represent the optimal bidding algorithm for the ESP user that is analyzed in UCS 2.3 in section 4.3. The only difference with the no-DLFM architecture presented earlier is that there is one more optimized FlexOffer made by the ESP to the FMO. These four bids made by the ESP are co-optimized in order to maximize the ESP's revenues. There is one more blue box that takes place after the DLFM clearing results are published and represents the ESP's optimal scheduling algorithm in order to minimize its OPEX. The detailed data model for this algorithm is presented in section 4.1 and corresponds to the UCS algorithm 2.1.

Regarding the aggregator user and the respective AFAT services, there are three orange boxes, which correspond to two main algorithms. The first orange box represents the optimal FlexOffer made by the aggregator to the FMO in the context of DLFM, while the third orange box represents the optimal aggregator's bidding in the near-real-time balancing market operated by the TSO. The data model of this bidding process is analyzed in section 5.2 (UCS 4.3). The second orange box represents the mathematical model and algorithmic solution of UCS 4.1, whose data model is analyzed in section 5.1. In particular, once the DLFM clearing results are published, the aggregator is informed about the FlexRequest schedule that has to execute. Thus, it applies a scheduling algorithm to optimally manage the FlexRequest in a way that maximizes the aggregator's profits and does not violate the end user's constraints that are explicitly stated and agreed in the FlexContracts.

[7] The pay-as-bid market clearing algorithm is implemented by NODES platform. FLEXGRID extends this approach by applying a network-aware market clearing approach.

2.3 Proactive DLFM architecture

The basic characteristic of P-DLFM is that distribution network (DN) level markets are cleared before the TN-level ones, so the 3 types of DLFMs operate proactively and thus based on their results, the TN-level markets follow. This process can also be seen as a "DN feasibility check" in order to mitigate the main drawback of the aforementioned R-DLFM model, which is the difficulty to manage an infeasible or expensive TN-level dispatch schedule. We assume:

- one day-ahead distribution level energy market (DA-DLEM) that takes place before the existing DA energy market (transmission level), and
- one near-real-time balancing market at distribution network level that takes place right before the existing balancing market operated by the TSO.

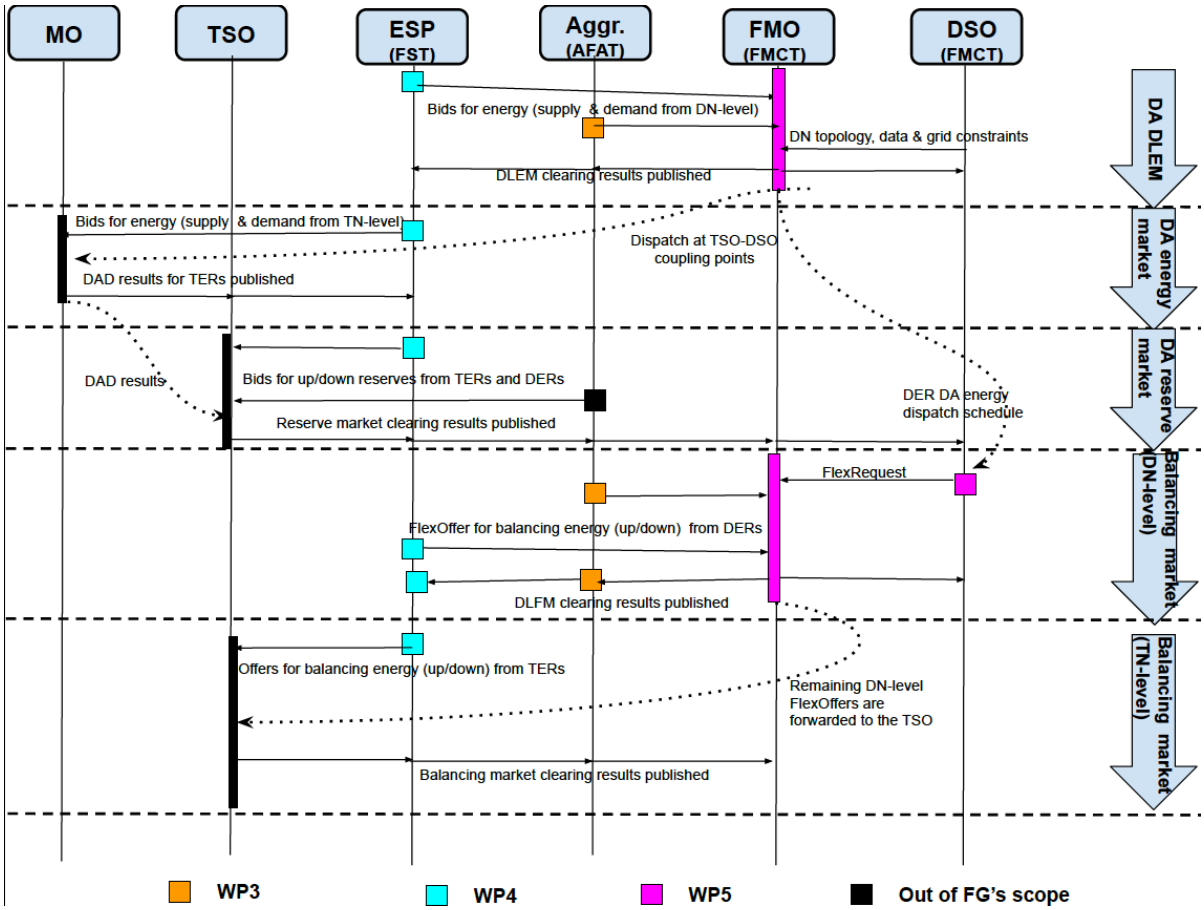

Figure 5: Proactive DLFM (DLFM precedes DA-EM and DA-RM)

The P-DLFM sequence diagram is illustrated in Figure 5. Regarding the DLFM clearing process (cf. two long purple boxes), the major difference compared to R-DLFM architecture is that the energy product is traded and not up/down reserve products. More details about the data model of this market clearing process is presented in section 3.1 (UCS 1.1). Correspondingly, the bidding algorithms and FlexOffers can also be slightly changed in order to serve the purpose of the energy products that are traded in both day-ahead and near-real-time timeframes.

Note: There are several variants of P-DLFM architecture that may be considered and are extensively described in section 2.4 of previous D5.1. Within the WP6 context, only the R-

DLFM architecture will be implemented, while one basic P-DLFM variant will also be supported. More specifically, the assumption is that residual DN-level FlexOffers that have not been accepted in the DN-level markets can be automatically forwarded to the TSO via a dedicated API. This means that the FLEXGRID ATP can serve as a gateway (or else intermediary platform) that can redirect local flexibility to the transmission level in order to help the TSO to deal with system-level imbalances. This novel functionality is also supported by NODES platform in a couple of real-life pilots within the EU area.

2.4 Interactive DLFM architecture

In the I-DLFM architecture model, we consider an iterative process that takes place between the MO and FMO and between TSO and DSO until they converge to an optimal dispatch schedule for both TN and DN levels. In the two figures below, the sequence diagrams for the MO-FMO coordination and TSO-DSO coordination are illustrated.

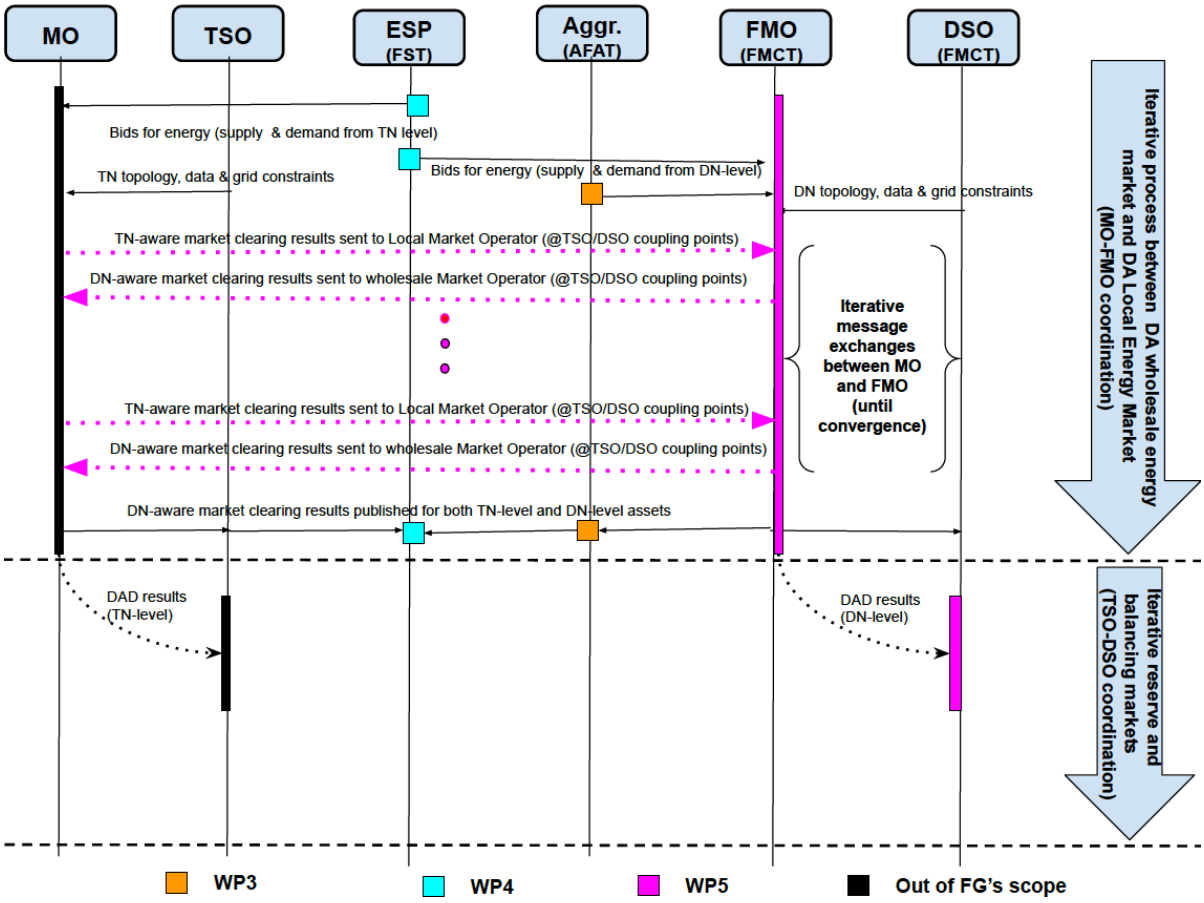

Figure 6: Interactive DLEM (iterative message exchanges between MO and FMO until convergence)

As shown in Figure 6, in the day-ahead energy market context, the MO initially runs an instance of its market clearing problem at the TN level and sends the results to the FMO. Then, the FMO takes as input the MO's results and runs its own market clearing problem at the DN level. The respective results (e.g. Lagrange multipliers) are sent back to the MO, who runs another round of the TN-level market clearing. Of course, the dispatch schedules that are decided in each round of algorithm's execution are virtual and are not actuated in reality.

After several algorithmic iterations (i.e. several message exchanges between MO and FMO), the process converges to an overall dispatch schedule (i.e. at both TN and DN levels) that maximizes the social welfare[8].

A similar iterative process shown in Figure 7 may take place for day-ahead reserve markets and near-real-time balancing markets (cf. TSO-DSO collaboration). We assume that day-ahead energy dispatch results are sent by the MO to the TSO and by the FMO to DSO. It should be noted that the I-DLFM architecture will not be developed at TRL 5 in the context of WP6, but rather at TRL 3 in WP5 and at TRL 4 in WP7 (cf. lab tests using AIT's large research infrastructure). I-DLFM is quite futuristic approach and is also incompatible with the existing EU regulation, even though it can theoretically achieve better social welfare results. However, the FLEXGRID data models that are followed in all I-DLFM processes are similar to the ones presented in the previously mentioned architectures.

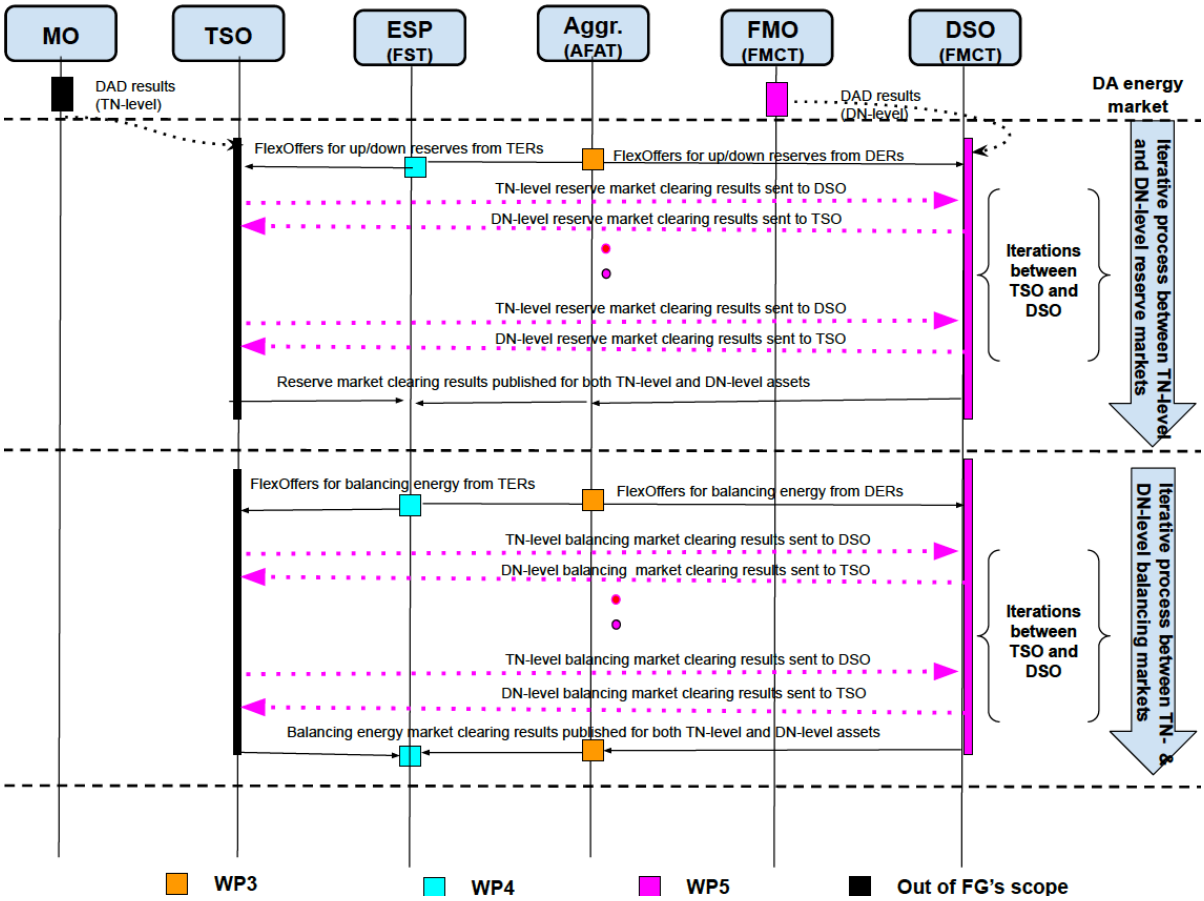

Figure 7: Interactive DLFM (iterative message exchanges between TSO and DSO until convergence)

[8] By the term "social welfare", we mean market efficiency and it is generally defined as the sum of all suppliers' profits and the "profits" from the demand side (i.e. consumers' utility minus costs).

3 Data model for the FMO and DSO user's frontend and FMCT backend

3.1 UCS 1.1 – DLFM clearing for the active power (energy) product

In this UCS, we consider a Flexibility Market Operator (FMO), who clears a **local energy market** after (i.e. R-DLEM) the transmission level commitments have been cleared. This means that some of the local generators and loads may already have committed parts of their energy to the wholesale transmission level (i.e. day-ahead energy market). The FMO runs a continuous pay-as-bid market, where FlexRequest from the DSO (i.e. FlexDemand side) and FlexOffers from ESPs (i.e. FlexSupply side) are continuously accepted and added to the orderbook. When the prices match, a network check is performed in order to ensure that no distribution network constraint is violated. Without loss of generality and within FLEXGRID's context, we assume that the full network model of the DSO is known to the FMO, as well as the active and reactive power setpoints committed in the wholesale transmission level market. The aim of the FMO is to maximize social welfare by matching all bids that result in feasible power flows. An auction-based market clearing algorithm (i.e. pay-as-clear) will also be available.

The novelty of the FLEXGRID's algorithmic approach is that the FMO clears the market continuously (or on an auction basis) and under full consideration of network constraints, i.e., including line and transformer ratings, reactive power limits, and voltage bands.

3.1.1 Proposed algorithm to integrate in FMCT

Technical details about the mathematical model, algorithmic solution and initial performance evaluation results are provided in chapter 2 of D5.2 (i.e. TRL 3). Within WP6 context (i.e. TRL 4-5), our main goal is to demonstrate that the FMO user visualizes in ATP the FlexRequest and FlexOffers that were accepted, and those that are rejected.

This algorithm could also be applicable for all other x-DLEM architecture variants; for P-DLEM, if all setpoints from the wholesale market are set to zero, and for I-DLEM if the setpoint of active and reactive power exchange at the Point of Common Coupling (PCC) are iteratively exchanged between wholesale (or else transmission network level) and local (or else distribution network level) market levels.

We distinguish two main operation modes for the FMO's GUI, namely:

- **Online operation:** The FMO user has the initiative. It accepts FlexOffers and FlexRequests and matches them in a continuous fashion whenever a new bid arrives. These cleared bids should also be made visible for the ESP user (i.e. FlexSeller) and DSO user (i.e. FlexBuyer).
- **Offline operation:** The FMO user runs various "what-if" simulation scenarios (using various OPF formulations and market clearing algorithms) to identify how it can achieve maximum expected social welfare. Only the FMO user will be able to visualize the results.

The market clearing process of this UCS has the purpose to clear energy on the day-ahead, intra-day, or in real-time.

- This traded product is defined as **active power per time unit, i.e., energy** in MWh/h.
- With a short lead time, pay-as-bid is recommended, with a long lead time, auctions (i.e. pay-as clear) are recommended.

In order to integrate the proposed mathematical model and algorithm at TRL 5, we have made the following assumptions:

- We clear only active energy per time unit, in e.g. MWh/h or kWh/(15min).
- Single-level optimization problem (i.e. the problem of the DSO and the problem of the TSO can be decoupled, as long as they know/forecast their state variables).
- The DLFM may clear before, simultaneous, or after the TSO's reserve or balancing market. The important feature is that DSOs and TSO must exchange information about their state variables, or at least about their active and reactive power exchange at the interface node (i.e. TSO-DSO coupling points).
- Convex reformulation of the Optimal Power Flow (OPF) algorithm; either a convex AC-OPF reformulation (SOCP) or a DC-OPF with an approximation of voltages at each bus.
- For the auction-based market clearing algorithm, Distribution Locational Marginal Prices (dLMPs) are computed at each distribution node or zone.

3.1.2 Algorithmic inputs and outputs and FMO/DSO frontend ideas

The following tables summarize the input parameters for the algorithm to run in the FMCT backend and output parameters for the results to be visualized in FMO's GUI (i.e. FMCT frontend) respectively.

Input parameters	FMO GUI in ATP	Central FG database
	Select FMO/DSO data per country (drop down menu with a few countries, e.g. Germany, Norway, Croatia)	The static Mongo-DB API will fetch the FMO user's inputs to the DB. The DB-FMCT API will fetch the selected time interval and selected markets from the central DB to the FMCT.
	Select time interval 'X' date to 'Y' date (cf. calendar)	
Energy balance forecast	Forecast load and generation at each node (assumed as known in WP5)	
Distribution network data	Lines with impedances, line current limits, bus voltage limits, bus voltage phase angle limits.	
Day-ahead energy market price quantity bids from all ESP users sorted by node and price (€/MWh)[9] [10]	Select for day-ahead energy market price data (cf. checkbox)	
Reserve market price quantity bids (up/down) from all ESP	Select for reserve market price data (cf. checkbox)	

[9] For reserve, DLFM and balancing markets, up and down regulation prices will be used.
[10] in the case of R-DLFM

users sorted by node and price (€/MW) [9] [10]		
DLFM market FlexOffers (price quantity bids up/down) from all ESP users sorted by node and prices[11] (€/MWh + €/MVar)	Select for DLFM price data (cf. checkbox)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST.
DLFM market FlexRequests (price quantity bids up/down) from DSO sorted by node and prices[11] (€/MWh + €/MVar)	Select DSO location areas (insert number)	The ATP-DB API will store the algorithmic results in the central DB.
Balancing market price quantity bids (up/down) from all ESP users sorted by node and price (€/MWh)	Select for balancing market price data (cf. checkbox)	The DB-ATP API will retrieve the data from the central DB.
ESP user unit specifications (fill in parameters in the GUI)	Fill in (for every ESP user unit – the user can add several units): - power capacity (kW) - energy capacity (kWh) - location of unit (node)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST.
Type of market clearing algorithm	Select from drop-down menu: - pay-as-bid - pay-as-clear	Type of market clearing algorithm
Type of optimal power flow	Select from dropdown menu: - Second order cone relaxation of AC-OPF - DC-OPF with approximations of losses and voltages	Type of optimal power flow
(*Optional:* Active power exchange from TSO)	Default value is 0	(*Optional:* Active power exchange from TSO) [3]
(*Optional:* Reactive power exchange from TSO)	Default value is 0	
(*Optional:* Excess active power FlexOffers not cleared in the FM, available for DLFM)	Default value is 0	

Output parameters	**FMO GUI in ATP**	**Central FG database**
dLMP for all distribution nodes	A graph that depicts: - Aggregate Quantity offered vs. price - Price accepted vs. node The ESP and DSO users should also be able to visualize these curves on their own GUIs.	
Quantity of active power, i.e., energy per time period for all distribution nodes	A graph that depicts the accepted price vs. node	The FMCT-ATP API will fetch the results from the FMCT to ATP.

[11] We assume a few location areas (cf. "polygons" concept from NODES platform) with different LMPs.

		The FMCT-DB API will store the same results to the central DB.
Voltages at all distribution nodes	A graph that depicts the voltages vs. node	
Power flows over all distribution lines	A graph that depicts the power flows vs. lines	
Voltage angles at all distribution nodes	A graph that depicts the voltage angles vs. node	
(*Optional:* Active power exchange with TSO)[12]		
(*Optional:* Reactive power exchange with TSO) [12]		
(*Optional:* Excess energy FlexOffers not cleared in the DLFM, available for TSO's balancing market) [12]	A graph that depicts: - Aggregate unaccepted quantity vs. price - Unaccepted price vs. node	

3.1.3 UML diagrams

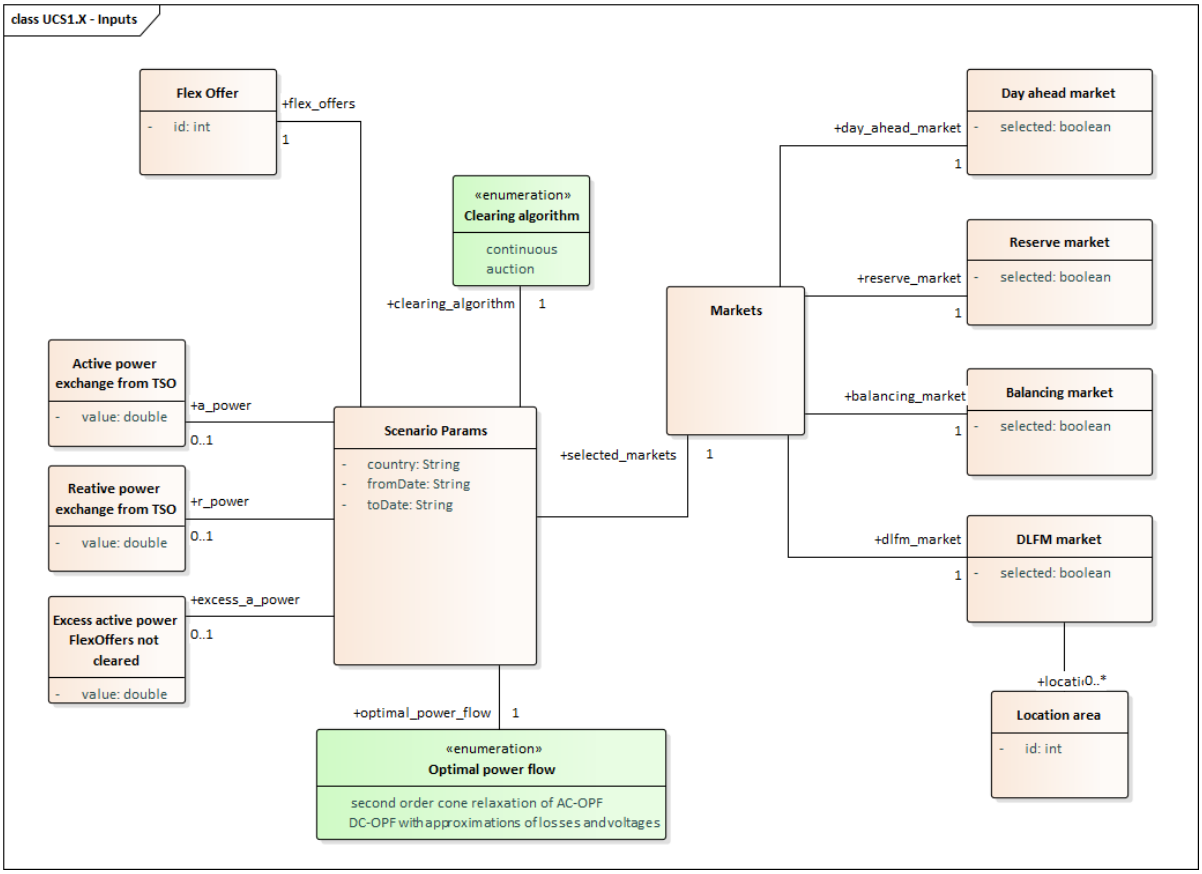

Figure 8: UML diagram for the ATP-FMCT API of UCS 1.1, UCS 1.2 and UCS 1.3 (i.e. FMO user's inputs filled in FMCT frontend and posted to FMCT backend)

[12] in the case of P-DLFM.

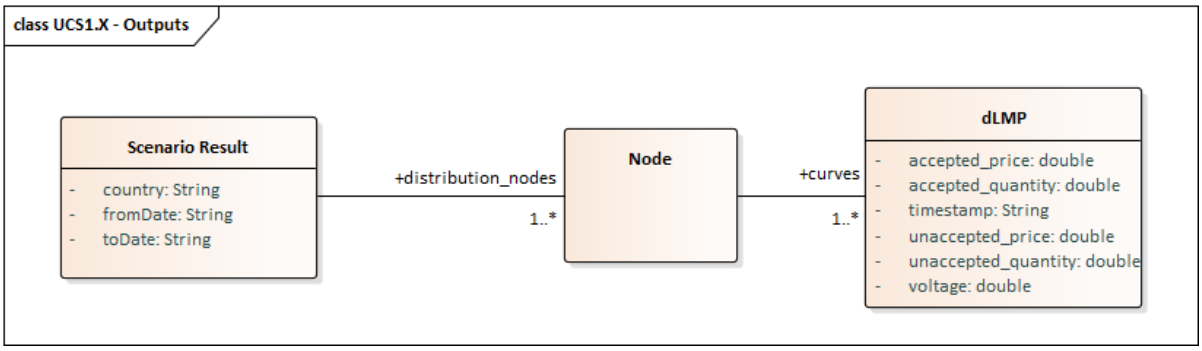

Figure 9: UML diagram for the ATP-FMCT API of UCS 1.1, UCS 1.2 and UCS 1.3 (i.e. algorithmic results produced by FMCT backend and visualized in FMCT frontend)

The two figures above depict the UML diagrams for UCS 1.1, UCS 1.2 and UCS 1.3 (the 3 of them share the same structure of inputs and outputs). The first one is the data model representation of the input parameters described in the table above regarding the ATP-FMCT API. The second one is the data model representation of the output parameters described in the table above regarding the FMCT-ATP API.

3.2 UCS 1.2 – DLFM clearing for the active power reserve (up/down) product

In this UCS, we consider a Flexibility Market Operator (FMO), who clears a **local active power reserve** market after (R-DLFM) the transmission level commitments have been cleared. This means that some of the local generators and loads may already have committed parts of their energy and/or reserve to the wholesale transmission level. The FMO runs a continuous pay-as-bid market where FlexRequest from the DSO and FlexOffers from FSPs are continuously accepted and added to the orderbook. When the prices match, a network check is performed in order to ensure that no network constraint is violated. Without loss of generality and within FLEXGRID's context, we assume that the full network model of the DSO is known to the FMO, as well as the active and reactive power setpoints committed in the wholesale transmission level market. The aim of the FMO is to maximize social welfare by matching all bids that result in feasible power flows. An auction-based market clearing algorithm (i.e. pay-as-clear) will also be available. In this algorithm, the FMO will gather all FlexRequests and FlexOffers for a given timeframe. When the gate closes, no other bids will be accepted and the network-aware auction-based market clearing algorithm will run.

The novelty of the FLEXGRID's algorithmic approach is that the FMO clears the market continuously (or on an auction basis) and under full consideration of network constraints, i.e., including line and transformer ratings, reactive power limits, and voltage bands. A second contribution is that this algorithm ensures that any combination of reserve activation is feasible for the network, opposed to current approaches, where one feasible reserve activation suffices.

3.2.1 Proposed algorithm to integrate in FMCT

Technical details about the mathematical model, algorithmic solution and performance evaluation results are provided in chapter 3 of D5.2 (i.e. TRL 3). Within WP6 context (i.e. TRL 4-5), our main goal is to demonstrate that the FMO user visualizes in ATP the FlexRequest

and FlexOffers that were accepted, and those that are rejected together with other important information that is presented below.

This algorithm could also be applicable for all other x-DLFM architecture variants; for P-DLFM if all setpoints from the wholesale market are set to zero, and for I-DLFM if the setpoint of active and reactive power exchange at the PCC are iteratively exchanged between wholesale and local market levels.

We distinguish two main operation modes for the FMO's GUI, namely:

- **Online operation:** The FMO user has the initiative. It accepts FlexOffers and FlexRequests and matches them in a continuous fashion whenever a new bid arrives. These cleared bids should also be made visible for the FSP user (i.e. FlexSeller) and DSO user (i.e. FlexBuyer).
- **Offline operation:** The FMO user runs various "what-if" simulation scenarios to identify how it can achieve maximum expected social welfare. Only the FMO user will be able to visualize the results.

The market clearing has the purpose to procure a vital DSO reserve service:

- **Congestion Management Reserve:** This reserve product consists of active power reserves (up- and downward) that are paid for their reserve power, but may not necessarily be activated in real-time.

In order to integrate the proposed mathematical model and algorithm at TRL 5, we have made the following assumptions:

- We clear only *active power reserves*, not energy.
- Single-level optimization problem (i.e. the problem of the DSO and the problem of the TSO can be decoupled, as long as they know/forecast their state variables).
- The DLFM may clear before, simultaneous, or after the TSO's FM. The important feature is that DSOs and TSO must exchange information about their state variables, or at least about their active and reactive power exchange at the interface node.
- Convex reformulation of the Optimal Power Flow (OPF) algorithm; either a convex AC-OPF reformulation (SOCP) or a DC-OPF with an approximation of voltages at each bus.
- For the auction-based market clearing algorithm, Distribution Locational Marginal Prices (dLMPs) must be computed at each distribution node.

3.2.2 Algorithmic inputs and outputs and FMO/DSO frontend ideas

The following tables summarize the input parameters for the algorithm to run in the FMCT backend and output parameters for the results to be visualized in FMO's GUI (i.e. FMCT frontend) respectively.

Input parameters	FMO GUI in ATP	Central FG database
	Select FMO/DSO data per country (drop down menu with a few countries, e.g. Germany, Norway, Croatia)	The ATP-DB API will fetch the FMO user's inputs to the DB. The DB-FMCT API will fetch the selected time

	Select time interval 'X' date to 'Y' date (cf. calendar)	interval and selected markets from the central DB to the FMCT
Energy balance forecast	Forecast load and generation at each node	
Distribution network data	In the form of a distribution network ID: Lines with impedances, line current limits, bus voltage limits, bus voltage phase angle limits.	
Day-ahead energy market price quantity bids from all ESP users sorted by node and price (€/MWh)	Select for day-ahead energy market price data (cf. checkbox)	
Reserve market price quantity bids (up/down) from all ESP users sorted by node and price (€/MW)	Select for reserve market price data (cf. checkbox)	
DLFM market price quantity bids (up/down) from all ESP users sorted by node and prices (€/MWh + €/MVar)	Select for DLFM price data (cf. checkbox)	
DLFM market FlexRequests (price quantity bids up/down) from DSO sorted by node and prices (€/MWh + €/MVar)	Select DSO location areas (insert number)	
Balancing market price quantity bids (up/down) from all ESP users sorted by node and price (€/MWh)	Select for balancing market price data (cf. checkbox)	
ESP user unit specifications (fill in parameters in the GUI)	Fill in (for every ESP user unit – the user can add several units): - power capacity (kW) - energy capacity (kWh) - location of unit (node)	The ATP-FMCT API will fetch the required data based on ESP user's inputs to FMCT.
Type of market clearing algorithm	Select from dropdown menu: - pay-as-bid - pay-as-clear	
Type of optimal power flow	Select from dropdown menu: - Second order cone relaxation of AC-OPF - DC-OPF with approximations of losses and voltages	
(*Optional:* Active power exchange from TSO)	Default value is 0	The ATP-FMCT API will fetch the required data from the TSO's inputs to FMCT
(*Optional:* Reactive power exchange from TSO)	Default value is 0	
(*Optional:* Excess active power FlexOffers not cleared in the TSO's reserve market, available for DLFM)	Default value is 0	

Output parameters	FMO GUI in ATP	Central FG database
dLMP for all distribution nodes	A graph that depicts: - Aggregate Quantity offered vs. price - Price accepted vs. node The ESP and DSO users should also be able to visualize these curves on their own GUIs.	
Quantity of active power reserve for all distribution nodes	A graph that depicts the accepted price vs. node	The FMCT-ATP API will fetch the results from the FMCT to ATP. The FMCT-DB API will store the same results to the central DB.
Votages at all distribution nodes	A graph that depicts the voltages vs. node	
Power flows over all distribution lines	A graph that depicts the power flows vs. lines	
Voltage angles at all distribution nodes	A graph that depicts the voltage angles vs. node	
(*Optional:* Active power exchange with TSO)[13]	Double type value	
(*Optional:* Reactive power exchange with TSO) [13]	Double type value	
(*Optional:* Excess active power FlexOffers not cleared in the DLFM, available for TSO's reserve market) [13]	A graph that depicts: - Aggregate unaccepted quantity vs. price - Unaccepted price vs. node	

3.2.3 UML diagrams

Taking into account the inputs and outputs of this UCS are the same also for UCS 1.1 and UCS1.3, the same UML diagrams depicted in Section 3.1.3 also apply to this UCS.

3.3 UCS 1.3 – DLFM clearing for the reactive power reserve (up/down) product

In this UCS, we consider a Flexibility Market Operator (FMO), who clears a **local reactive power reserve** market after (R-DLFM) the transmission level commitments have been cleared. This means that some of the local generators and loads may already have committed parts of their energy and/or reserve to the wholesale transmission level. The FMO runs a continuous pay-as-bid market where FlexRequest from the DSO and FlexOffers from ESPs are continuously accepted and added to the orderbook. When the prices match, a network check is performed in order to ensure that no network constraint is violated. Without loss of

[13] in the case of P-DLFM

generality and within FLEXGRID's context, we assume that the full network model of the DSO is known to the FMO, as well as the active and reactive power setpoints committed in the wholesale transmission level market. The aim of the FMO is to maximize social welfare by matching all bids that result in feasible power flows. An auction-based market clearing algorithm (i.e. pay-as-clear) will also be available. In this algorithm, the FMO will gather all FlexRequests and FlexOffers for a given timeframe. When the DLFM gate closes, no other bids will be accepted and the network-aware auction-based market clearing algorithm will run to clear the market.

The novelty of the FLEXGRID's algorithmic approach is that the FMO clears the market continuously (or on an auction basis) and under full consideration of network constraints, i.e., including line and transformer ratings, reactive power limits, and voltage bands. A second contribution is that this algorithm ensures that any combination of reserve activation is feasible for the network, opposed to current approaches, where one feasible reserve activation suffices.

3.3.1 Proposed algorithm to integrate in FMCT

Technical details about the mathematical model, algorithmic solution and performance evaluation results are provided in chapter 3 of D5.2 (i.e. TRL 3). Within WP6 context (i.e. TRL 4-5), our main goal is to demonstrate that the FMO user visualizes in ATP the FlexRequest and FlexOffers that were accepted, and those that are rejected together with other important information that is presented below.

This algorithm could also be applicable for all other x-DLFM architecture variants; for P-DLFM if all setpoints from the wholesale market are set to zero, and for I-DLFM if the setpoint of active and reactive power exchange at the PCC are iteratively exchanged between wholesale level and local levels.

We distinguish two main operation modes for the FMO's GUI, namely:

- **Online operation:** The FMO user has the initiative. It accepts FlexOffers and FlexRequests and matches them in a continuous fashion whenever a new bid arrives. These cleared bids should also be made visible for the FSP user (i.e. FlexSeller) and DSO user (i.e. FlexBuyer).
- **Offline operation:** The FMO user runs various "what-if" simulation scenarios to identify how it can achieve maximum expected social welfare. Only the FMO user will be able to visualize the results.

The market clearing has the purpose to procure a vital DSO reserve service:

- **Voltage Management Reserve:** This reserve product consists of reactive power reserves (up- and downward) that are paid for their reserve power, but may not necessarily be activated in real-time.

In order to integrate the proposed mathematical model and algorithm at TRL 5, we have made the following assumptions:

- We clear only *reactive power reserves*, not energy.

- Single-level optimization problem (i.e. the problem of the DSO and the problem of the TSO can be decoupled, as long as they know/forecast their state variables).
- The DLFM may clear before, simultaneous, or after the TSO's reserve market. The important feature is that DSOs and TSO must exchange information about their state variables, or at least about their active and reactive power exchange at the interface node.
- Convex reformulation of the Optimal Power Flow (OPF) algorithm; either a convex AC OPF reformulation (SOCP) or a DC OPF with an approximation of voltages at each bus.
- For the auction-based market clearing, Reactive Distribution Locational Marginal Prices (qLMPs) must be computed at each distribution node.

3.3.2 Algorithmic inputs and outputs and FMO/DSO frontend ideas

The following tables summarize the input parameters for the algorithm to run in the FMCT backend and output parameters for the results to be visualized in FMO's GUI (i.e. FMCT frontend) respectively.

Input parameters	FMO GUI in ATP	Central FG database
	Select FMO/DSO data per country (drop down menu with a few countries, e.g. Germany, Norway, Croatia)	The ATP-DB API will fetch the FMO user's inputs to the DB. The DB-FMCT API will fetch the selected time interval and selected markets from the central DB to the FMCT
	Select time interval 'X' date to 'Y' date (cf. calendar)	
Energy balance forecast	Forecast load and generation at each node	
Distribution network data	In the form of a distribution network ID: Lines with impedances, line current limits, bus voltage limits, bus voltage phase angle limits.	
Day-ahead energy market price quantity bids from all ESP users sorted by node and price (€/MWh)	Select for day-ahead energy market price data (cf. checkbox)	
Reserve market price quantity bids (up/down) from all ESP users sorted by node and price (€/MW)	Select for reserve market price data (cf. checkbox)	
DLFM market price quantity bids (up/down) from all ESP users sorted by node and prices (€/MWh + €/MVar)	Select for DLFM price data (cf. checkbox)	
DLFM market FlexRequests (price quantity bids up/down) from DSO sorted by node and prices (€/MWh + €/MVar)	Select DSO location areas (insert number)	

Balancing market price quantity bids (up/down) from all ESP users sorted by node and price (€/MWh)	Select for balancing market price data (cf. checkbox)	
ESP user unit specifications (fill in parameters in the GUI)	Fill in (for every ESP user unit – the user can add several units): • power capacity (kW) • energy capacity (kWh) • location of unit (node)	The ATP-FMCT API will fetch the required data based on ESP user's inputs to FMCT
Type of market clearing algorithm	Select from dropdown menu: - pay-as-bid - pay-as-clear	
Type of optimal power flow	Select from dropdown menu: - LinDistFlow	
(*Optional:* Active power exchange from TSO) → cf. R-DLFM case	Default value is 0	The ATP-FMCT API will fetch the required data from the TSO's inputs to FMCT.
(*Optional:* Reactive power exchange from TSO) → cf. R-DLFM case	Default value is 0	
(*Optional:* Excess reactive power FlexOffers not cleared in the FM, available for DLFM) → cf. R-DLFM case	Default value is 0	

Output parameters	**FMO GUI in ATP**	**Central FG database**
dLMP for all distribution nodes	A graph that depicts: • Aggregate Quantity offered vs. price • Price accepted vs. node The ESP and DSO users should also be able to visualize these curves on their own GUIs.	
qLMP for all distribution nodes	A graph that depicts: • Aggregate Quantity offered vs. price • Price accepted vs. node The ESP and DSO users should also be able to visualize these curves on their own GUIs.	The FMCT-ATP API will fetch the results from the FMCT to ATP. The FMCT-DB API will store the same results to the central DB.
Quantity of active power reserve for all distribution nodes	A graph that depicts the accepted price vs. node	The FMCT-ATP API will fetch the results from the FMCT to ATP.

Quantity of reactive power reserve for all distribution nodes	A graph that depicts the accepted price vs. node	The FMCT-DB API will store the same results to the central DB.
Voltages at all distribution nodes	A graph that depicts the voltages vs. node	
Power flows over all distribution lines	A graph that depicts the power flows vs. lines	
Voltage angles at all distribution nodes	A graph that depicts the voltage angles vs. node	
(*Optional:* Active power exchange with TSO) → cf. P-DLFM case	Double type value	
(*Optional:* Reactive power exchange with TSO) → cf. P-DLFM case	Double type value	
(*Optional:* Excess active power FlexOffers not cleared in the DLFM, available for TSO's reserve market) → cf. P-DLFM case	A graph that depicts: • Aggregate unaccepted quantity vs. price • Unaccepted price vs. node	

3.3.3 UML diagrams

Taking into account the inputs and outputs of this UCS are the same also for UCS 1.1 and UCS 1.2, the same stricture of UML diagrams depicted in Section 3.1.3 above also apply to this UCS.

3.4 Sequence diagram for communication between ATP frontend and FMCT backend

As already discussed, we will integrate three UCS (and respective network-aware market clearing algorithms) in the Flexibility Market Clearing Toolkit (FMCT). The S/W architecture follows a modular-by-design approach, which allows the different S/W modules to be developed on a standalone basis by the various partners and communicate with each other via well-defined and fine-grained APIs. The following figure describes in five main steps the process that will be followed. There are three main S/W components that will be developed, namely:

- **ATP GUI or else FMCT frontend:** this will be developed by ETRA.
- **FMCT backend:** this will be developed by DTU (cf. UCS 1.1 – 1.3).
- **FLEXGRID Central Database:** this will be developed by ETRA, while real-life input data will be provided by DTU (in collaboration with other consortium partners, too).

First of all, the FMO user will login the FLEXGRID ATP, will be authenticated through a single sign-on process and then will be redirected to the main FMO's GUI. Via the FMCT application, the FMO user will be able to visualize, configure and manage the DLFM under consideration. Three main products are considered: i) active power (energy) product (cf. UCS 1.1), ii) active power reserve product (cf. UCS 1.2), and iii) reactive power reserve product (cf. UCS 1.3).

Moreover, the FMO user will be able to run the DLFM clearing algorithms in order to decide on optimal dispatch decisions that minimize the flexibility procurement cost and communicate them to the involved stakeholders (i.e. DSO and ESPs).

For each one of the three algorithms, there will be a tab in the FMCT frontend. Once the FMO user clicks on one tab, s/he will be able to configure/customize/fill in the input parameters that are needed for each algorithm to be able to run. Once the FMO user clicks on the "Run algorithm" button, step 1 process will be followed[14]. More specifically, the API client that resides at the FMCT frontend will automatically gather all input parameters and will send them to the API server that resides at the FMCT backend.

After the FMCT backend receives the input parameters, the next step is to request for the required input data from the FLEXGRID central database (DB). More specifically, an API client that resides at FMCT backend may request for input data from an API server residing at the central DB. In step 3, the input data is retrieved, and now the algorithm can be executed.

Once the algorithm produces the results, these output parameters will be automatically gathered by the FMCT-ATP API and will be sent to the FMCT frontend so that the FMO user can visualize the results in a comprehensive and user-friendly manner. The final step (i.e. step 5) is for the FMO user to understand the results and if s/he is interested in further elaborating on them, then s/he can optionally select to store them in the central DB in order to be able to retrieve, visualize and possibly compare them with other market clearing results in the future.

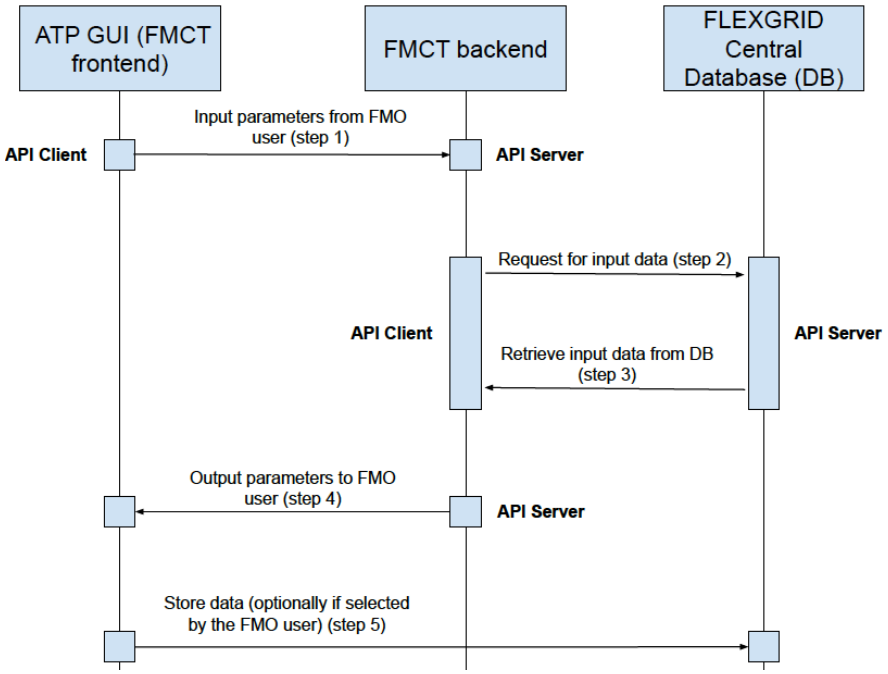

Figure 10: Sequence diagram for communication among ATP frontend, FMCT backend and central FLEXGRID database

[14] More technical details about the steps depicted in Figure 10 are provided in FLEXGRID D6.2.

4 Data model for the ESP user's frontend and FST backend

4.1 UCS 2.1 – Minimize ESP's Operational Expenditures (OPEX)

In this UCS, we consider a profit-oriented Energy Service Provider (ESP), who owns a set of Battery Storage Units (BSUs). In the case that the model is not network-aware, the location of the BSUs is not relevant, but only their characteristics (e.g. capacity, efficiency, etc.). On the TRL 5 level, we consider: 1) Day-ahead energy market (DA-EM), 2) Reactive Distribution Level Flexibility Market (R-DLFM) and 3) Balancing Market (BM). The ESP user participates in the DA-EM and has the opportunity to maximize its profits by taking part in DLFM, too. However, such market behaviour may lead to a change in the agreed DA-EM schedule and this will mean that the ESP will have to pay/or get paid for the imbalances that it created in the BM.

In order for the ESP to meet the FlexRequest requirements and maximize profit, an optimal scheduling algorithm is utilized. More specifically, the FLEXGRID optimal scheduling algorithm aims to minimize ESP's operational expenses as sub-optimal strategies may significantly reduce profits and even potentially endanger sustainability of the company's business model.

4.1.1 Proposed algorithm to integrate in FST

Technical details about the mathematical model, algorithmic solution and initial performance evaluation results are provided in chapter 3 of D4.1 and chapter 3 of D4.2 (i.e. TRL 3). Within WP6 context (i.e. TRL 5), our main goal is to demonstrate that the ESP user is able to visualize its operational expenses (OPEX), categorized per different criteria (e.g. type of asset, point in time). From the numerical data and visualizations, the ESP user should gain insight what could possibly lower/increase its OPEX if changes are made in the DA-EM schedule. More precisely, the ESP user should be able to compare the new schedule (i.e. after responding to a FlexRequest in intra-day timeframe) versus the old one (i.e. the DA-EM schedule that has been agreed in the day-ahead timeframe).

We distinguish two main operation modes for the ESP's GUI, namely:

- **Online operation:** We assume that the day-ahead energy market (DA-EM) dispatch is given and should be respected by the ESP. Then, a FlexRequest issued by DSO/TSO needs to be met by the ESP. Both DA-EM dispatch and the FlexRequest should be made visible for the ESP user. The updated/new schedule (i.e. result of the proposed optimal scheduling algorithm) should also be made visible to the DSO and FMO user.
- **Offline operation:** The ESP user runs various "what-if" simulation scenarios assuming various FlexRequests and FlexAsset portfolios. The goal is to assess the ESP's hypothetical participation in the proposed DLFM in the future in a way that is profitable for its business operation. Only the ESP user will be able to visualize these results.

In order to integrate the proposed mathematical model and algorithm at TRL 5 (i.e. in the FLEXGRID ATP), we have made the following assumptions:

- We solve a single-level optimization problem (i.e. ESP is price-taker).
- We assume that the ESP user is not network-aware (specific case would be the one where ESP is also a DSO or in close relationship with the DSO).
- We assume that all FlexOffers in the proposed DLFM will be accepted.
- The predicted market prices are taken as a parameter, or in the case of the offline operation – historical market prices are taken as parameter.
- We assume that the network data is provided, should (in a specific case) the model be network-aware.
- We assume that the R-DLFM architecture is adopted, because this is the most compatible with the existing EU regulatory framework.

4.1.2 Algorithmic inputs and outputs and ESP frontend ideas

The following tables summarize the input parameters for the algorithm to run in the FST backend and output parameters for the results to be visualized in ESP's GUI (i.e. FST frontend) respectively.

Input parameters	**ESP GUI in ATP**	**Central FG database**
	Select MO/TSO data per country (drop down menu with a few countries, e.g. Germany, Norway, Croatia)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST. The DB-FST API will fetch the required market price data, schedules and FlexRequests from the selected country, selected time interval and day-ahead market from the central DB to the FST.
	Select time interval 'X' date to 'Y' date (cf. calendar)	
Day-ahead energy market price data: a vector of 24-hourly price values per selected day and country (€/MWh)	Select for day-ahead energy market price data (cf. checkbox)	
Day-ahead energy schedule	Fill in or choose from an existing one	
FlexRequest	Fill in or choose from an existing one	
Balancing market price data: vector of 24-hourly price values per selected day and country (€/MWh)	Fill in or choose from an existing one	
Storage unit specifications (fill in parameters in the GUI)	Fill in (for every storage unit – the user can add several units): • power capacity (KW) • energy capacity (KWh) • inefficiency rate (%) • initial/final SoE (%) • location of storage unit (location id)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST.
RES and consumption data	Fill in or choose from existing data that is available in the DB	The ATP-FST API will fetch the required data

		based on ESP user's inputs to FST.
Option to store data in the central FLEXGRID database	Checkbox that can be either checked or not.	The ATP-DB API will store the algorithmic results in the central DB.
Past OPEX minimization scenarios	The ESP can select a past scenario to view it in the FST GUI.	The DB-ATP API will retrieve the data from the central DB.

Output parameters	ESP GUI in ATP	Central FG database
One optimized FlexOffer curve per selected market → 24-hourly vector of (quantity) for the given timeframe	A graph per selected market that depicts: • Quantity offered vs. time The FMO and DSO users should also be able to visualize these curves on their own GUIs.	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB.
Revenues (€) per selected market and scenario	A graph per selected market that depicts: • Excessive revenues (€) from participating in the DLFM Only the ESP user visualizes these results.	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB.
Scheduling	A graph that depicts: • Old vs. new schedule per controllable FlexASset Only the ESP user visualizes these results	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB

According to the information above, a draft version of the ESP GUI sketches has been developed by ETRA as well as the first version of the FST-ATP-DB APIs. Indicative screenshots and more technical details can be found in D6.2.

4.1.3 UML diagrams for UCS 2.1

The following two figures depict the UML diagram for UCS 2.1. The first one is the data model representation of the input parameters described in the table above regarding the ATP-FST API. The second one is the data model representation of the output parameters described in the table above regarding the FST-ATP API.

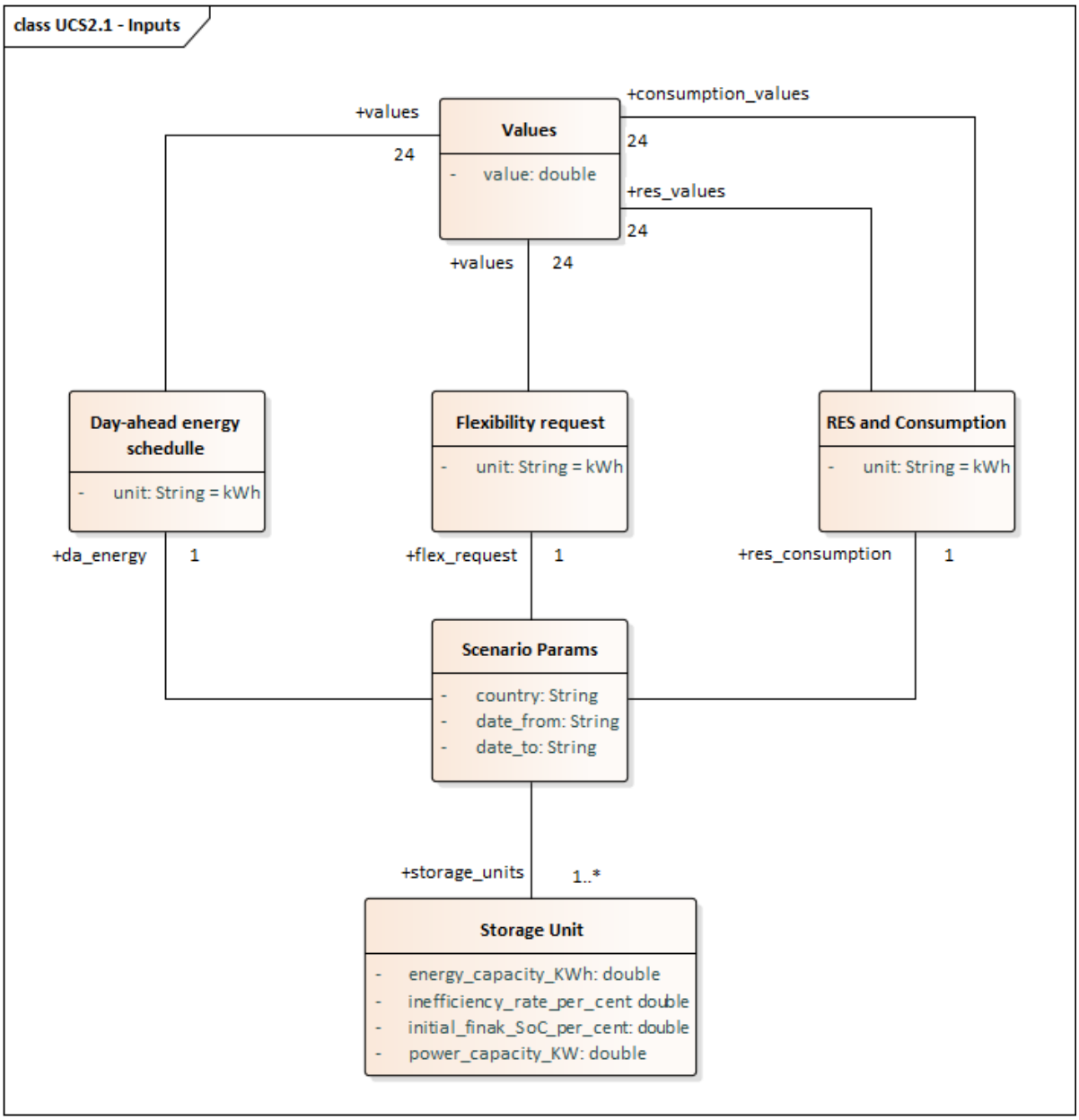

Figure 11: UML diagram for the ATP-FST API of UCS 2.1 (i.e. ESP user's inputs filled in FST frontend and posted to FST backend)

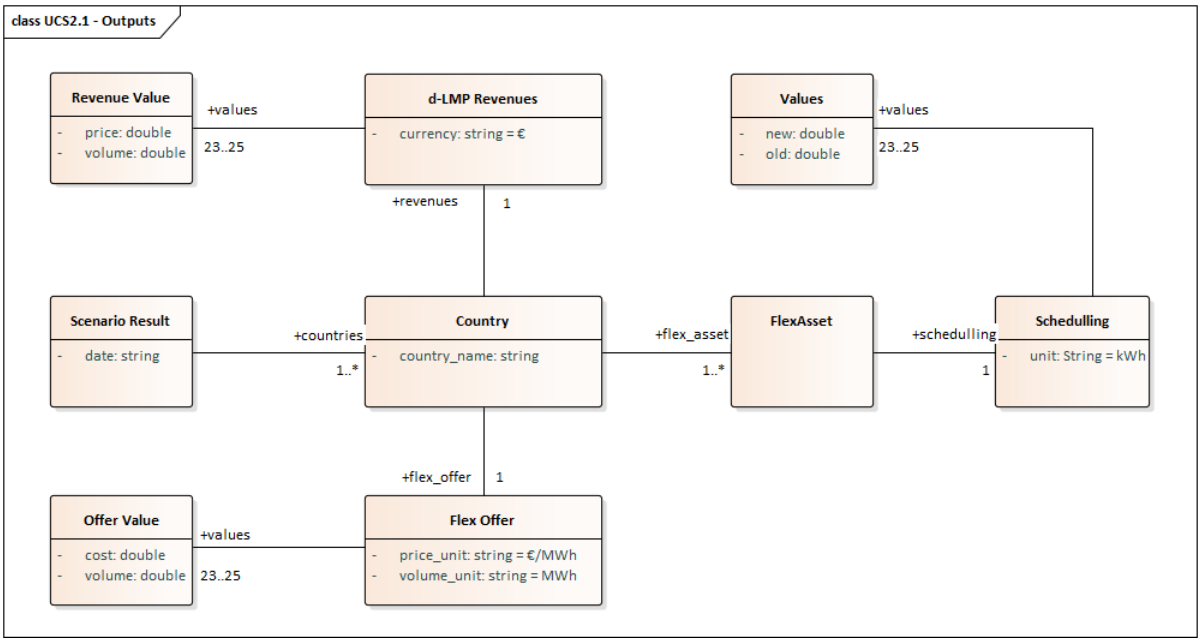

Figure 12: UML diagram for the ATP- FST API of UCS 2.1 (i.e. algorithmic results produced by FST backend and visualized in FST frontend)

4.2 UCS 2.2 – Minimize ESP's Capital Expenditures (CAPEX)

In this UCS, we consider a profit-oriented Energy Service Provider (ESP), who may already own various FlexAssets and wants to possibly invest in new ones in the future. Those FlexAssets (emphasizing mainly on the Battery Storage Units for the TRL 3) may be located on various nodes of a radial distribution network. Depending on the ESP's insight into the network topology given by a respective DSO, the respective level of granularity may vary from the node resolution to the division of the observed radial network into several zones[15]. For the WP6 purposes, the ESP is assumed to participate in the following markets: 1) Day-ahead energy market (DA-EM), 2) Reserve market (RM), 3) Reactive Distribution Level Flexibility Market (R-DLFM).

The ESP's CAPEX minimization problem focuses on the use of a FlexAsset siting and sizing algorithm, when deciding on ESP's future investments on new FlexAssets. It is formulated as a single-level problem with a network-aware approach. Although the respective ESP might not have a full insight of the network topology, the most important information about the zones, determined by the respective DSO upon some criteria (similarly to the NODES approach), are provided. Moreover, **CAPEX minimization problem is dependent on various given conditions (e.g. ESP wants to reduce OPEX by 5% by investing in new FlexAssets) and circumstances on the observed markets. Hence, OPEX should also be considered, while tackling the CAPEX minimization problem. Such an approach should enable efficient exploitation of available instruments to ensure reliable energy supply with the minimum CAPEX. The main novelty lies on the inclusion of R-DLFM and DN-aware approach.**

[15] This approach is adopted by NODES flexibility market. NODES follows a hierarchical structure for dividing the DN topology into several location areas/zones.

4.2.1 Proposed algorithm to integrate in FST

Technical details about the mathematical model, algorithmic solution and performance evaluation results are provided in chapter 5 of D4.1 and chapter 4 of D4.2 (i.e. TRL 3). Within WP6 context (i.e. TRL 5), our main goal is to demonstrate ESP user's ability to visualize its total investment costs with respect to the given objective (e.g. 5% or 10% OPEX reduction). Furthermore, it should be easy to categorize the costs per different criteria (type of asset and its main characteristics, asset location, network status). In such manner, the ESP user should have a better insight why is a specific investment (on RES and/or FlexAssets) needed and what benefits (in addition to costs) it brings. In other words, both the minimum size of the new FlexAsset to install and its location should be visualized.

The ESP user runs various "what-if" simulation scenarios (only offline operation) :

- **Offline operation:** The ESP user runs various "what-if" simulation scenarios assuming various mixes of FlexRequests and FlexAsset portfolios. ESP assumes a given OPEX reduction target (e.g. 5%) and tries to find the minimum CAPEX to meet this target. Moreover, we assume a few DSO areas. Each DSO area may have different DLFM price series, thus implying the potential DN-level congestion/voltage problems. Only the ESP user will be able to visualize the results.

In order to integrate the proposed mathematical model and algorithm at TRL 5, we have made the following assumptions:

- We formulated the model as a single-level optimization problem (i.e. ESP is price-taker).
- The model is network-aware (at least following NODES platform's design approach).
- We assume that the ESP has the possibility to participate in various electricity markets.
- We assume that ESP and DSO may be one entity or at least the DSO provides some high-level topology data (cf. NODES platform's design approach).
- We assume data transparency.
- We take the predicted (or historical) market prices as an input parameter.

4.2.2 Algorithmic inputs and outputs and ESP frontend ideas

The following tables summarize the input parameters for the algorithm to run in the FST backend and output parameters for the results to be visualized in ESP's GUI (i.e. FST frontend) respectively.

Input parameters	ESP GUI in ATP	Central FG database
	Select MO/TSO data per country (drop down menu with a few countries, e.g. Germany, Norway, Croatia)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST. The DB-FST API will fetch the required market price data from the selected country, selected time interval and selected markets from the central DB to the FST.
	Select time interval 'X' date to 'Y' date (cf. calendar)	
	Select users from the previously selected MO/TSO	

Day-ahead energy market price data: a vector of 24-hourly price values per selected day and country (€/MWh)	Select for day-ahead energy market price data (cf. checkbox)	
Reserve market (e.g. secondary or tertiary) price data (up/down): a vector of 24-hourly price values per selected day and country (€/MW)	Select for reserve market price data (cf. checkbox)	
DLFM market price data: a vector of 24-hourly price values per selected day, location area and type of service (€/MWh + €/MVar)	Select for DLFM price data (cf. checkbox) Select DSO location areas (insert location id, e.g. 1, 2, … for a given TSO area)	
Balancing market price data: a vector of 24-hourly price values per selected day and country (€/MWh)	Select for balancing market price data (cf. checkbox)	
Network topology data (fill in the parameters in the GUI or fetch from the DB)	Select from the few pre-saved network topologies or fill in the required parameters. Among other, it includes data about existing FlexAssets	The ATP-FST API will fetch the required data based on ESP user's inputs to FST or from the pre-saved network topologies in the central DB
Specifications both of current storage units in ESP's portfolio (if any) and potential ones (fill in parameters in the GUI) Some predefined should already be stored in the central DB	Fill in (for every storage unit – the user can add several units): • power capacity (KW) • energy capacity (KWh) • inefficiency rate (%) • initial/final SoE (%) • location of storage unit (location id)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST or from the pre-saved ones in the central DB.
OPEX reduction target (%)	Fill in the value	The ATP-FST API will fetch the required data based on ESP user's inputs to FST
Specifications of potential FlexAssets (fill in parameters in the GUI) Some predefined should already be stored in the central DB	Fill in the values or choose from the existing ones	The ATP-FST API will fetch the required data based on ESP user's inputs to FST or from the pre-saved ones in the central DB.
Constraints regarding viable locations for potential new FlexAssets and budget constraints (if any)	Fill in the forbidden locations and maximum budget.	The ATP-FST API will fetch the required data based on ESP user's inputs to FST or from the pre-saved ones in the central DB.

Option to store data in the central FLEXGRID database	Checkbox that can be either checked or not.	The ATP-DB API will store the algorithmic results in the central DB.
Various CAPEX minimization scenarios	The ESP can select a past scenario to view it in the FST GUI.	The DB-ATP API will retrieve the data from the central DB.

Output parameters	ESP GUI in ATP	Central FG database
Revenues (€) per selected market and scenario	A graph per selected market that depicts: • Revenues (€) vs. market Only the ESP user visualizes these results.	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB.
Siting (location in the grid) and Sizing (kW/kWh)	Show the minimum size of new FlexAssets to install and in which DSO area. Only the ESP user visualizes these results.	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB
CAPEX (€)	Show the required indicative investment budget (euros) for the assumed timeframe (e.g. 1 year) and RoI (in years). Only the ESP user visualizes these results.	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB

According to the information above, a draft version of the ESP GUI sketches has been developed by ETRA as well as the first version of the FST-ATP-DB APIs. Indicative screenshots and more technical details can be found in D6.2.

4.2.3 UML diagrams for UCS 2.2

The following two figures depict the UML diagram for UCS 2.2. The first one is the data model representation of the input parameters described in the table above regarding the ATP-FST API. The second one is the data model representation of the output parameters described in the table above regarding the FST-ATP API.

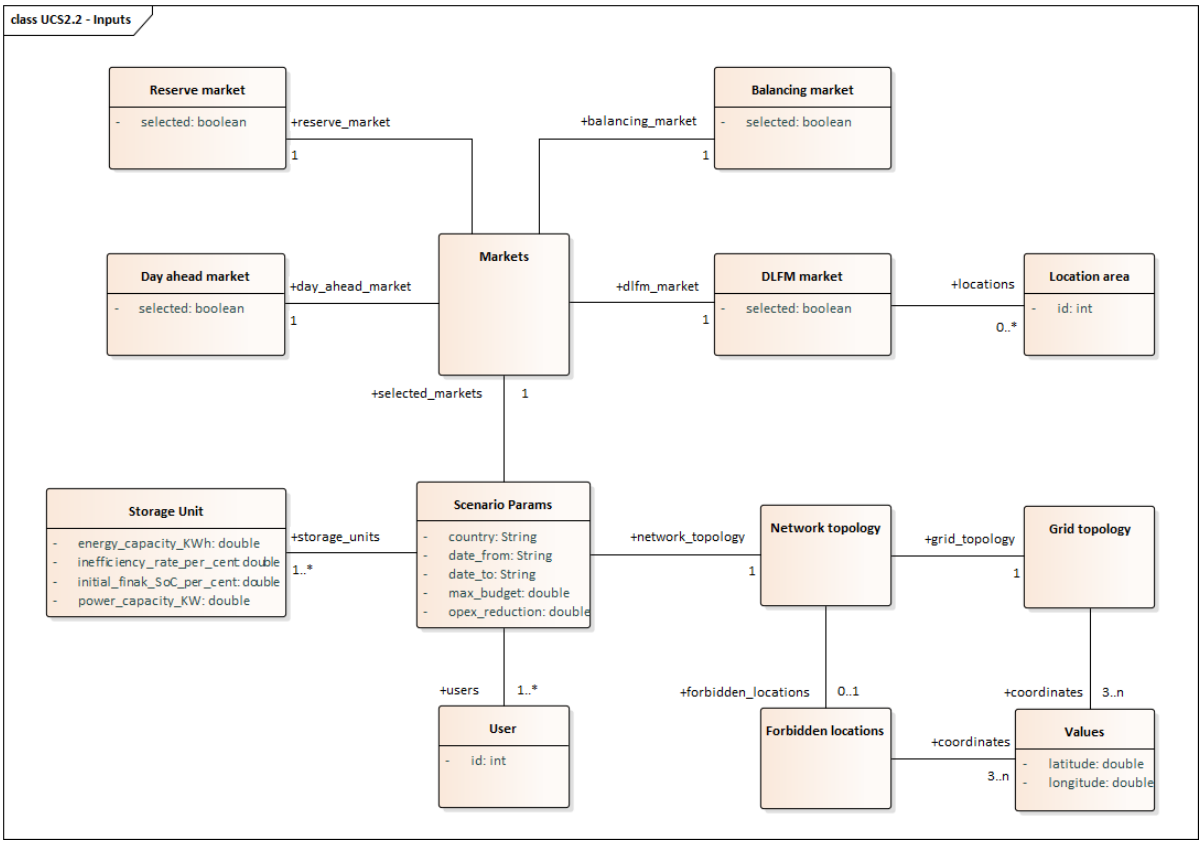

Figure 13: UML diagram for the ATP-FST API of UCS 2.2 (i.e. ESP user's inputs filled in FST frontend and posted to FST backend)

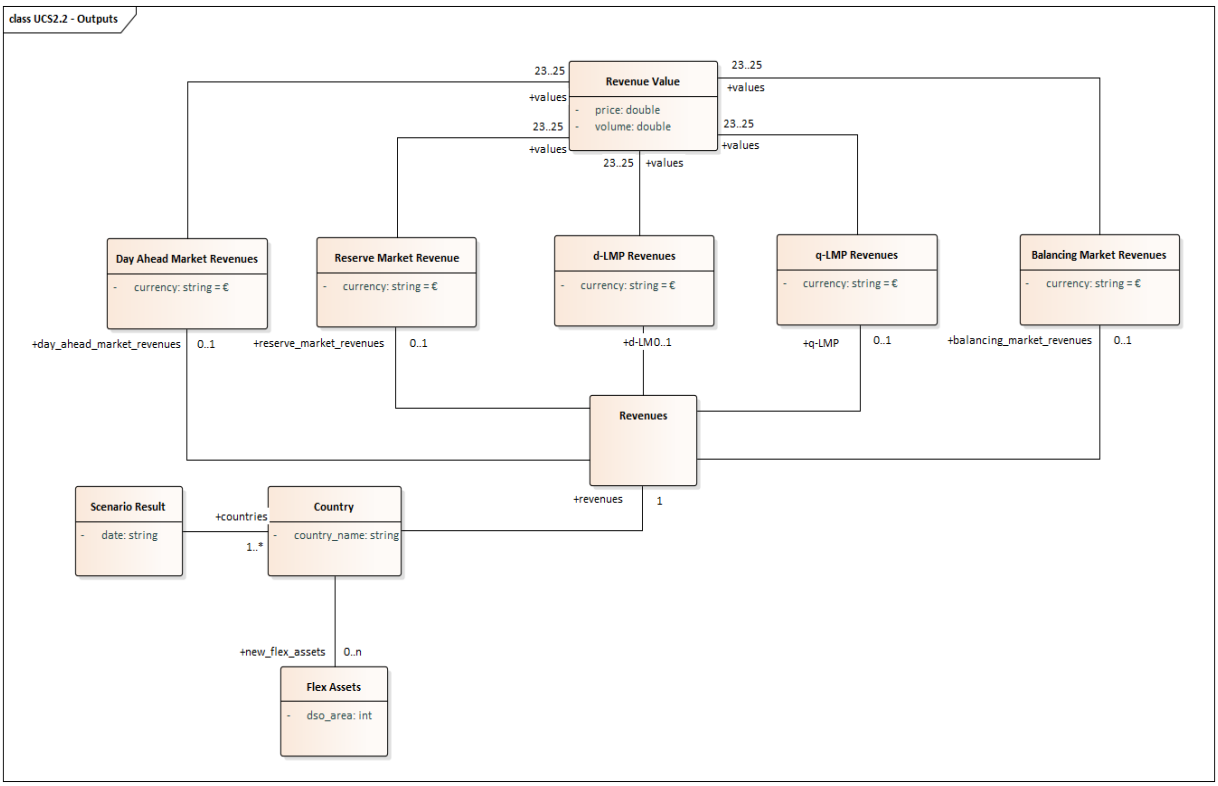

Figure 14: UML diagram for the ATP- FST API of UCS 2.2 (i.e. algorithmic results produced by FST backend and visualized in FST frontend)

4.3 UCS 2.3 – Maximize ESP's stacked revenues

In this UCS, we consider a profit-seeker Energy Service Provider (ESP), who owns a set of Battery Storage Units (BSUs) located at various nodes of a radial distribution network. In order to maximize its profits, the ESP may participate in several energy/reserve markets and dynamically optimize its bidding strategy. Without loss of generality and within FLEXGRID's context, we assume the ESP's participation in four markets: 1) Day-Ahead Energy Market (DA-EM) operated by the MO, 2) Day-Ahead Reserve Market (DA-RM) operated by the TSO, 3) Day-Ahead Distribution-Level Flexibility Market (DA-DLFM) operated by a novel market entity called Flexibility Market Operator (FMO), and 4) Balancing Market (BM) operated by the TSO.

The objective function of the ESP's problem is to maximize its aggregated profits from the four aforementioned markets. **The novelty of the FLEXGRID's mathematical model and algorithmic approach is that the ESP co-optimizes its participation in various markets instead of simply participating in each one of them individually in a sequential manner.** As far as the day-ahead wholesale energy market is concerned, the ESP decides the BSUs' operation schedule by taking as input the nodal price, which corresponds to the node of the transmission grid at which the distribution network is connected. Secondly, the ESP makes profit by providing upward and downward reserves in the DA-RM. The upward/downward reserve prices are obtained from the reserve market clearing process and are the same throughout the transmission grid. Thirdly, the ESP participates in the DA-DLFM by providing flexibility services to the DSO (i.e. active and reactive power (P-flexibility and Q-flexibility) based on nodal prices within the distribution network). Finally, the ESP participates in the near-real-time BM to balance its portfolio.

4.3.1 Proposed algorithm to integrate in FST

Technical details about the mathematical model, algorithmic solution and performance evaluation results are provided in chapter 5 of D4.1 and chapter 5 of D4.2 (i.e. TRL 3). Within WP6 context (i.e. TRL 5), our main goal is to demonstrate that the ESP user visualizes in ATP its business profits by simultaneously participating in a different combination of the afore-mentioned markets. Of course, profits will be more if the ESP participates simultaneously in more markets.

We distinguish two main operation modes for the ESP's GUI, namely:

- **Online operation:** The ESP user has the initiative. It takes market price forecasting data for the four markets (i.e. DA-EM, DA-RM, DA-DLFM and BM) and calculates four optimal energy/Flex offers to submit in ATP. These offers should also be made visible for the FMO user (i.e. DLFM operator) and DSO user (i.e. FlexBuyer).
- **Offline operation:** The ESP user runs various "what-if" simulation scenarios via running a stacked revenue maximization algorithm to identify how it can achieve maximum expected profits. Only the ESP user will be able to visualize the results.

In order to integrate the proposed mathematical model and algorithm at TRL 5, we have made the following assumptions:

- We assume a single-level optimization problem (i.e. ESP is price-taker, not price-maker), instead of the more complex bi-level optimization model that has been developed within WP4 context.

- We do not take into account the underlying network model. In other words, we assume that all FlexOffers do not cause any network/grid problems.
- We assume that all FlexOffers will be accepted, so the FlexAssets (batteries) will be accordingly scheduled.
- We may assume that historical market prices are the real ones (especially for the offline operation). If we use the "market price forecasting" module developed by UCY (see section 4.4 below), then we should first run the forecasting algorithm, which takes as input a selected market price data series and produces as output the forecast market price data series. Then, we assume that the latter data series will be the real market prices at the time of delivery.
- We assume that the R-DLFM architecture is adopted, because this is the most compatible with the existing EU regulatory framework.

4.3.2 Algorithmic inputs and outputs and ESP frontend ideas

The following tables summarize the input parameters for the algorithm to run in the FST backend and output parameters for the results to be visualized in ESP's GUI (i.e. FST frontend) respectively.

Input parameters	ESP GUI in ATP	Central FG database
	Select MO/TSO data per country (drop down menu with a few countries, e.g. Germany, Norway, Croatia)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST. The DB-FST API will fetch the required market price data from the selected country, selected time interval and selected markets from the central DB to the FST.
	Select time interval 'X' date to 'Y' date (cf. calendar)	
Day-ahead energy market price data: a vector of 24-hourly price values per selected day and country (€/MWh)	Select for day-ahead energy market price data (cf. checkbox[16])	
Reserve market (e.g. secondary or tertiary) price data (up/down): a vector of 24-hourly price values per selected day and country (€/MW)[17]	Select for reserve market price data (cf. checkbox)	
DLFM market price data: a vector of 24-hourly price values per selected day, location area and type of service[18] (€/MWh + €/MVar)	Select for DLFM price data (cf. checkbox) Select DSO location areas (insert location id, e.g. 1, 2, … for a given TSO area)	

[16] If the ESP user checks the checkbox, then the data from the respected market will be retrieved from the database. If not, then we assume a simulation scenario in which the ESP does not participate in a given market.
[17] For reserve market, up and down regulation prices will be used.
[18] We assume a few location areas (cf. "polygons" from NODES platform) with different LMPs. We also assume 2 types of services for the DN: i) d-LMPs for local congestion management problem and ii) q-LMPs for voltage control problem. We assume that we use realistic DLFM prices based on NODES experience and international literature. We also assume that the price of up/down regulation is the same.

Balancing market price data[19]: a vector of 24-hourly price values per selected day and country (€/MWh)	Select for balancing market price data (cf. checkbox)	
Storage unit specifications (fill in parameters in the GUI)	Fill in (for every storage unit – the user can add several units): - power capacity (KW) - energy capacity (KWh) - inefficiency rate (%) - initial/final SoC (%) - location of storage unit (location id)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST.
Option to store data in the central FLEXGRID database	Checkbox that can be either checked or not!	The ATP-DB API will store the algorithmic results in the central DB.
Stacked revenue scenarios to view in FST GUI (ATP)	The ESP can select a past scenario to view it in the FST GUI.	The DB-ATP API will retrieve the data from the central DB.

Output parameters	ESP GUI in ATP	Central FG database
One optimized energy/Flex offer curve per selected market --> 24-hourly vector of (quantity, time) for the given market price (i.e. ESP is a pricer-taker)[20]	A graph per selected market that depicts: • Quantity offered vs. time The FMO and DSO users should also be able to visualize these curves on their own GUIs.	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB (if option has been selected by ESP user → see above).
Revenues (€) per selected market and scenario[21]	A graph per selected market that depicts: • Revenues (€) vs. market Only the ESP user visualizes these results.	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB. (if option has been selected by ESP user → see above).

The data model of UCS 2.3 is currently the most mature one and can be found in FLEXGRID GitHub area[22]. For more convenience and easier use of FLEXGRID services after the end of

[19] We may also assume that the price of up/down regulation is the same for the balancing market.

[20] We will have 6 curves in the same graph, namely: i) day-ahead energy (positive & negative quantity values), ii) reserve (one curve for up-regulation quantity and one curve for down/regulation quantity), iii) DLFM (one curve for active (P) power quantity and one curve for reactive power (Q) quantity), iv) Balancing energy (positive & negative power quantity values).

[21] In the x axis, we will have 4 markets (one bar per market showing the revenues in the y axis).

[22] https://github.com/FlexGrid/atp_service

the project's lifetime, we provide the FLEXGRID API data models in the form of swagger (and .yaml) files. For example, the next figure provides an indicative screenshot of this service.

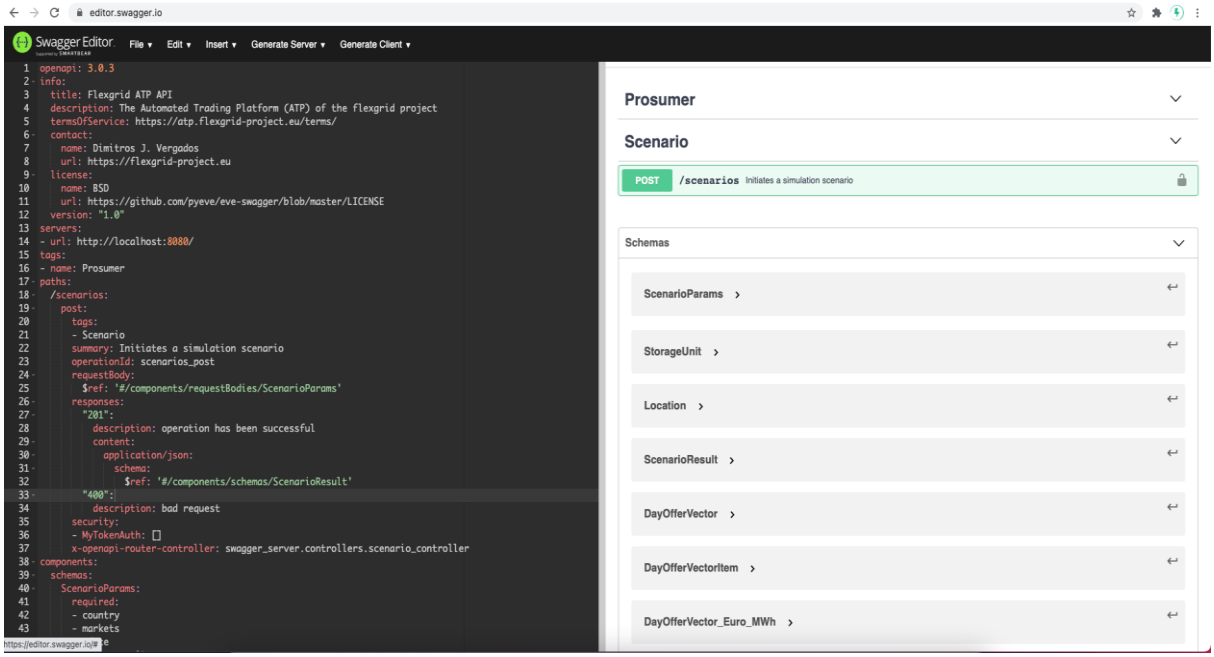

Figure 15: Example of swagger file for UCS 2.3 API data model

According to the information above, a draft version of the ESP GUI sketches has been developed by ETRA as well as the first version of the FST-ATP-DB APIs. Indicative screenshots and more technical details can be found in D6.2. It should also be noted that UCS 2.3 algorithm will be fully integrated in FLEXGRID ATP by M21 and a respective live demonstration will take place during the Period 1 review meeting on 22/06/2021.

4.3.3 UML diagrams for UCS 2.3

The following two figures depict the UML diagram for UCS 2.3. The first one is the data model representation of the input parameters described in the table above regarding the ATP-FST API. The second one is the data model representation of the output parameters described in the table above regarding the FST-ATP API.

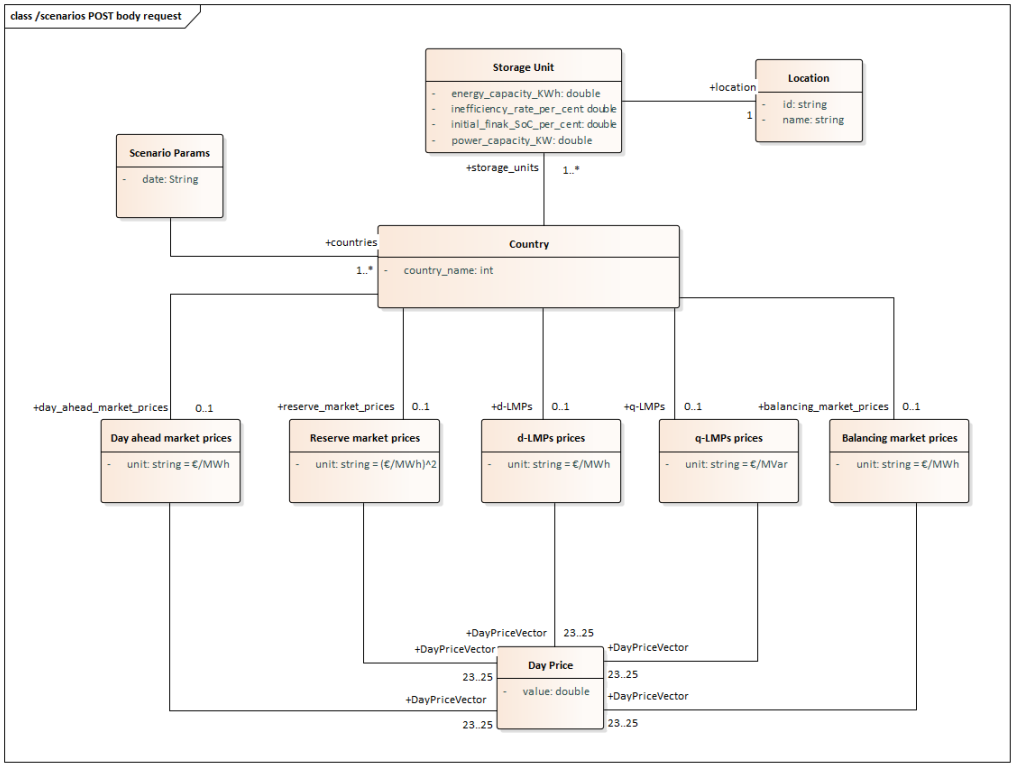

Figure 16: UML diagram for the ATP-FST API of UCS 2.3 (i.e. ESP user's inputs filled in FST frontend and posted to FST backend)

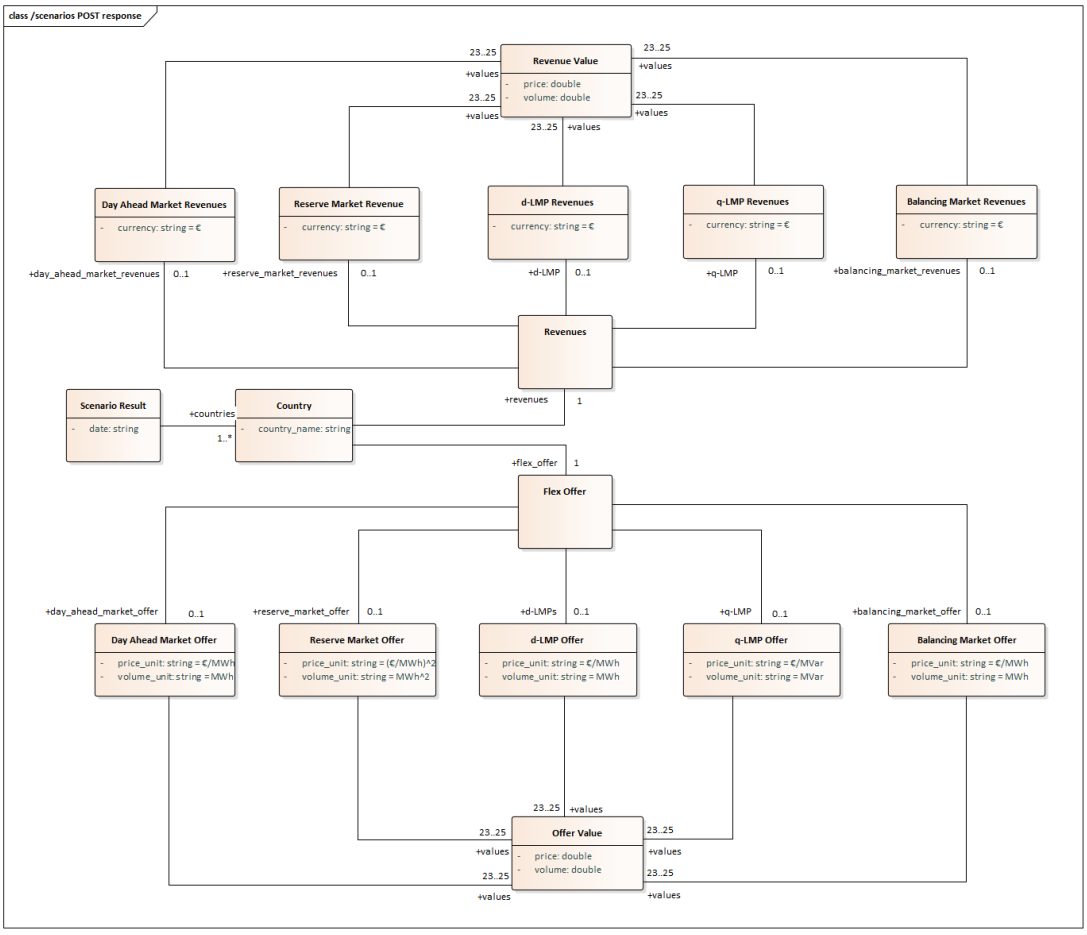

Figure 17: UML diagram for the FST-ATP API of UCS 2.3 (i.e. algorithmic results produced by FST backend and visualized in FST frontend)

4.4 UCS 4.4 – PV and Market price forecasting

In this UCS, we deal with two forecasting problems, namely: i) PV generation forecasting, and ii) market price forecasting. In the following two subsections, we provide relevant information for these two problems.

4.4.1 PV generation forecasting

4.4.1.1 Proposed algorithm to integrate in FST

Advanced generation forecasting tool should be developed that will provide dynamic estimation of RES production curves (RPCs) based on historical and other data for specific geographical locations. These forecasts will help in hedging the ESP's risks and allow the sustainable commercial exploitation of the energy produced by RES.

A set of cutting-edge machine learning models (Bayesian Regularised Neural Network (BRNN)) and methods are proposed for the PV generation forecasting problem. BRNN utilises the Bayes-Newton regularisation to minimise the weights of the neural network and consequently minimise the error of the model. **BRNN are very efficient in terms of computational time and processing, while the number of input parameters (or else features) that are required to adapt to the behaviour of a specific PV plant or for a larger aggregation area, are minimum**.

- **Online operation:** The PV generation forecasting tool requires as input parameters numerical weather prediction (NWP) and sun angles (elevation and azimuth) data to calculate the day-ahead PV power generation forecast. The models will be trained based on historical NWP data and power measurements.
- **Offline operation:** The forecasting models will be trained with historical parameters only (no live training will be performed), while the PV generation forecasting output will be based only on historical parameters.

Assumptions:

- Historical data (NWP and power measurements) will be provided for at least 6-8 months (fewer months could also be utilised but with decreased accuracy).
- Aggregated data will be provided for specific areas to provide aggregated DA PV production forecasting. If aggregated data is not available, the UCY will aggregate the data from specific points.
- The NWP data will be provided to the UCY in order to provide day-ahead PV generation forecast.

4.4.1.2 Algorithmic inputs and outputs and ESP frontend ideas

The following tables summarize the input parameters for the algorithm to run in the FST backend and output parameters for the results to be visualized in ESP's GUI (i.e. FST frontend) respectively.

Input parameters	ESP GUI in ATP	Central FG database
	Select data per country (drop down menu with a few countries,	The ATP-FST API will fetch the required data based on

	e.g. Cyprus, Germany, Norway, Croatia).	ESP user's inputs to FST. The DB-FST API will fetch the required data from the selected country and selected time interval from the central DB to the FST.
	Select time interval 'X' date to 'Y' date (cf. calendar) – [YYYY-MM-DD hh:mm:ss].	
Nominal installed capacity of the PV system [W].	Fill in the nominal installed capacity of the PV system in W.	
PV system coordinates (latitude, longitude).	Fill in the PV system coordinates in decimal degrees format: ● Latitude (e.g. 35.21474) ● Longitude (e.g. 33.25541).	
Historical measured P_{ac} power data [W].	Download button that includes the template ("download.csv" file) of the csv file that needs to be uploaded. Upload button to upload the historical data in CSV file: Timestamp (YYYY-MM-DD hh:mm:ss), Historical NWP - GHI (W/m^2), Historical NWP - T_{amb} (˚C), Historical measured - P_{ac} (W).	
Historical NWP data: ● Historical forecasted GHI or G_{POA} $[W/m^2]$ ● Historical forecasted T_{amb} [˚C].		
Day-ahead (48 points – half-hour resolution) NWP data per PV system location area: ● GHI or G_{POA} $[W/m^2]$ ● T_{amb} [˚C].	Download button that includes the template ("download.csv" file) of the csv file that needs to be uploaded. Upload button to upload the NWP data per PV system location are in CSV file: Timestamp (YYYY-MM-DD hh:mm:ss), NWP - GHI (W/m^2), NWP - T_{amb} (˚C).	

Output parameters	ESP GUI in ATP	Central FG database
Day-ahead PV power generation forecast [W] - 48 points (half-hour resolution)	A Line graph that depicts: ● PV power generation forecast [W] vs Timestamp [YYYY-MM-DD hh:mm:ss]	● The FST-ATP API will fetch the results from the FST to ATP. ● The ATP-DB API will store the results to the central DB.

4.4.2 Market price forecasting

4.4.2.1 Proposed algorithm to integrate in FST

Technical details concerning the mathematical model, algorithmic solution and performance evaluation results are provided in Chapter 2 of D4.1 and Chapter 2 of D4.2. The goal of the algorithm is to create a reliable forecasting tool that will be used to forecast the Day-Ahead 24 electricity values and their corresponding confidence intervals using historical data from

specific areas and forecasts of other variables such as weather conditions and power demand. This tool will help to address the risks and thereby will provide insights to ESP's bidding, scheduling and planning processes (cf. UCS 2.1-2.3) in view of increasing its profits.

The algorithm will be based on the Extreme Learning Machine (ELM) method. This algorithm is a single layer feed-forward network (SLFN) having the important feature of the short learning period. The algorithm would be able to run both online and offline as described below:

- **Online operation:** The algorithm requires historical data, through which it will be trained. Historical data from the day-ahead energy market or any other auction-based energy market (e.g. balancing market) as well as forecasts of other related variables pertaining to weather conditions and energy demand will be used as inputs to the algorithm. As output, the day-ahead forecasted 24 values of electricity along with the corresponding confidence intervals will be provided.
- **Offline operation:** The ESP can use the available data as described above to carry out simulations of different scenarios in order to assess their validity with respect to several forecast accuracy KPIs.

Assumptions:

- The data of 72 hourly values are entered as input to the algorithm.
- Available data will be used, depending on the market in which the ESP would like to get the forecast prices.

4.4.2.2 Algorithmic inputs and outputs and ESP/aggregator frontend ideas

The following tables summarize the input parameters for the market price forecasting algorithm to run in the FST backend and output parameters for the results to be visualized in ESP's GUI (i.e. FST frontend) respectively.

Input parameters	ESP GUI in ATP	Central FG database
	Select MO/TSO data per country (drop down menu with a few countries, e.g. Germany, Norway, Croatia)	The ATP-FST API will fetch the required data based on ESP user's inputs to FST. The DB-FST API will fetch the required market price data from the selected country, selected time interval and selected markets from the central DB to the FST.
	Select 'X' date for forecast (cf. calendar) – [dd/MM/yyyy]	
Day-ahead energy market price data: a vector of 72-hourly price values per selected day and country (€/MWh)	Select for day-ahead energy market price data	
Selection of data from any auction-based market with uniform pricing: a vector of 72-hourly price values per selected day and country (€/MWh)	Select for corresponding energy market price data	
Day ahead weather forecast		

Output parameters	ESP GUI in ATP	Central FG database
One curve with day-ahead 24-hourly vector for the given timeframe	A graph that depicts: - 24 day-ahead hourly prices vs. time (dd/MM/yyyy, HH:mm:ss)	The FST-ATP API will fetch the results from the FST to ATP. The ATP-DB API will store the same results to the central DB.
Confidence Intervals and Market Forecast Accuracy Level		

4.4.3 UML diagrams for UCS 4.4

The following four figures depict the UML diagrams for UCS 4.4:

- The first one is the data model representation of the input parameters related with the PV generation forecasting described in the table above regarding the ATP-FST API.
- The second one is the data model representation of the output parameters related with the PV generation forecasting described in the table above regarding the FST-ATP API.
- The third one is the data model representation of the input parameters related with the market price forecasting described in the table above regarding the ATP-FST API.
- The fourth one is the data model representation of the output parameters related with the market price forecasting described in the table above regarding the FST-ATP API.

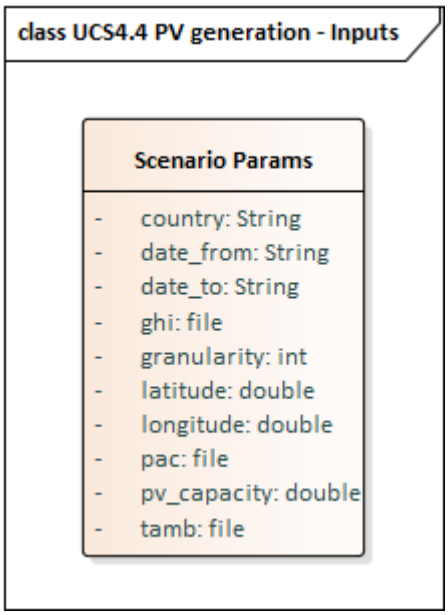

Figure 18: UML diagram for the ATP-FST API of UCS 4.4 for PV generation forecast (i.e. ESP user's inputs filled in FST frontend and posted to FST backend)

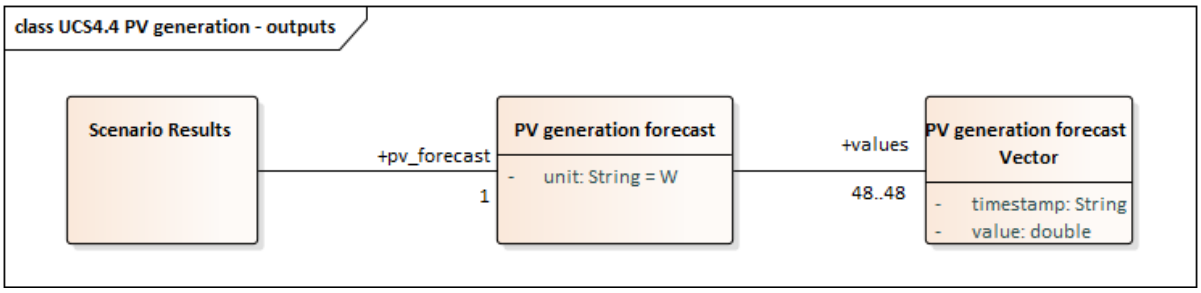

Figure 19: UML diagram for the FST-ATP API of UCS 4.4 for PV generation forecast (i.e. algorithmic results produced by FST backend and visualized in FST frontend)

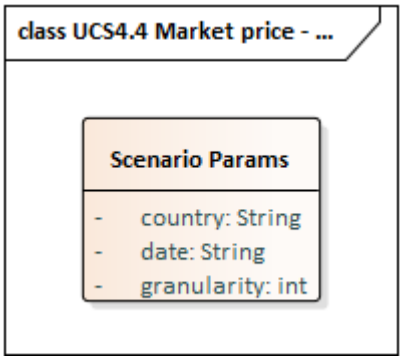

Figure 20: UML diagram for the ATP-FST API of UCS 4.4 for market price forecasting (i.e. ESP user's inputs filled in FST frontend and posted to FST backend)

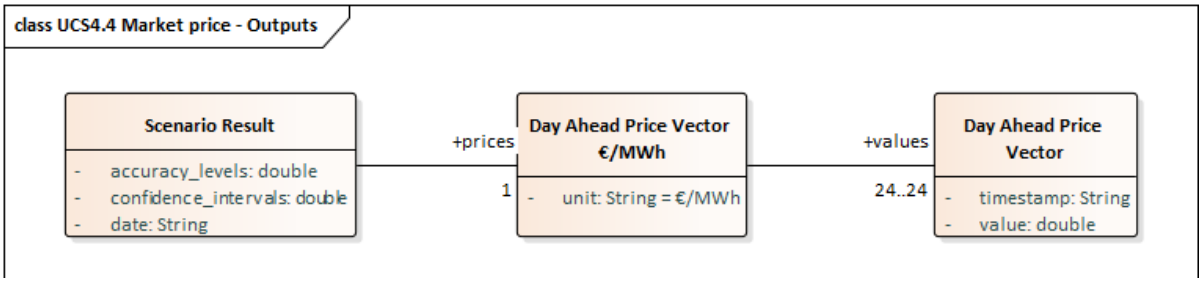

Figure 21: UML diagram for the FST-ATP API of UCS 4.4 for market price forecasting (i.e. algorithmic results produced by FST backend and visualized in FST frontend)

4.5 Sequence diagram for communication between ATP frontend and FST backend

As already discussed, we will integrate four UCS in the FlexSuppliers' Toolkit (FST). The S/W architecture follows a modular-by-design approach, which allows the different S/W modules to be developed on a standalone basis by the various partners and communicate with each other via well-defined and fine-grained APIs. The following figure describes in five main steps the process that will be followed[23]. There are three main S/W components that will be developed, namely:

- **ATP GUI or else FST frontend:** this will be developed by ETRA.
- **FST backend:** this will be developed collaboratively by UNIZG (cf. UCS 2.1 & 2.2), ICCS (UCS 2.3) and UCY (UCS 4.4).

[23] More technical details about the steps depicted in Figure 22 are provided in FLEXGRID D6.2.

- **FLEXGRID Central Database:** this will be developed by ETRA, while real-life input data will be provided by each one of the three research partners.

First of all, the ESP user will login the FLEXGRID ATP, will be authenticated through a single sign-on process and then will be redirected to the main ESP's GUI. Via the FST application, the ESP user will be able to visualize, configure and manage its FlexAssets. Moreover, the ESP user will be able to run four main algorithms in order to be able to make optimal scheduling (cf. UCS 2.1), planning (cf. UCS 2.2), bidding (cf. UCS 2.3) and forecasting decisions (cf. UCS 4.4).

For each one of the four algorithms, there will be a tab in the FST frontend. Once the ESP user clicks on one tab, s/he will be able to configure/customize/fill in the input parameters that are needed for each algorithm to be able to run. Once the ESP user clicks on the "Run algorithm" button, step 1 process will be followed. More specifically, the API client that resides at the FST frontend will automatically gather all input parameters and will send them to the API server that resides at the FST backend.

After the FST backend receives the input parameters, the next step is to request for the required input data from the FLEXGRID central database (DB). More specifically, an API client that resides at FST backend request for input data from an API server residing at the central DB. In step 3, the input data is retrieved, and now the algorithm can be executed.

Once the algorithm produces the results, these output parameters will be automatically gathered by the FST-ATP API and will be sent to the FST frontend so that the ESP user can visualize the results in a comprehensive and user-friendly manner. The final step (i.e. step 5) is for the ESP user to understand the results and if s/he is interested in further elaborating them, then s/he can optionally select to store them in the central DB in order to be able to retrieve, visualize and possibly compare them with other results in the future.

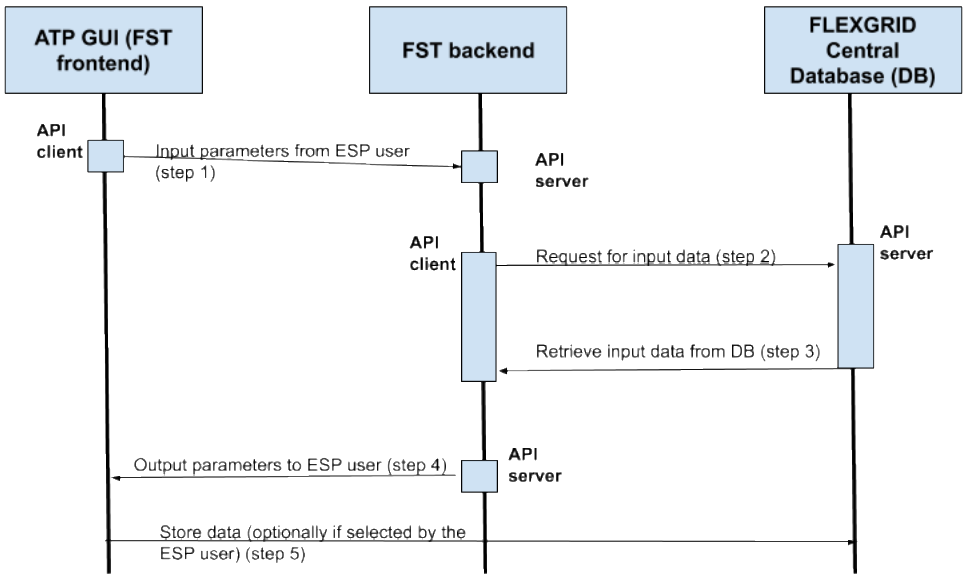

Figure 22: Sequence diagram for communication among ATP frontend, FST backend and central database

5 Data model for the aggregator user's frontend and AFAT backend

5.1 UCS 4.1 – Manage a FlexRequest

In this UCS, we consider a commercial and independent aggregator entity, whose objective is to increase its profits from selling various flexibility services by optimally representing and managing his aggregated flexibility portfolio consisting of DERs of end users. End users are motivated to participate in the aggregator's portfolio with monetary incentives, which are established through appropriate FlexContracts.

The aggregator's objective is to maximize its profits from participation in the flexibility market. This translates to maximization of the revenues and minimization of the associated costs. The revenues of the aggregator increase with positive responses to FlexRequests. The associated costs can be divided into two categories. The first are end-user compensations for provision of flexibility, defined in FlexContracts. The second involves potential imbalance costs, meaning the financial effect of activating flexibility and deviating from the baseline (scheduled energy profile of the flexibility assets). Presence of imbalance costs depends on the interaction of the flexibility market with other existing energy markets.

5.1.1 Proposed algorithm to integrate in AFAT

The more complex problem formulation and technical details of mathematical/system model, algorithmic solution, simulation setup performance evaluation results for reaching the objective of the aggregator, when managing a FlexRequest-Activation are analyzed in chapter 3 of D3.1 and chapter 2 of D3.2 (TRL 3). The goal of the implementation of UCS 4.1 in WP6 (TRL 5) is the demonstration and visualization of the portfolio of the aggregator user and the profits (revenues-costs) of effectively responding to FlexRequests.

The independent aggregator user visualizes in AFAT the profit of accepting a FlexRequest and the "consequences"/remaining flexibility of its portfolio after the positive response. The goal is to deviate from the baseline only by the amount of energy of the FlexRequest, which was accepted by the aggregator. This requires appropriate selection of FlexAssets to activate/dispatch based on cost and effects on future time slots.

Two modes of operation are considered:

- **Online operation:** A new FlexRequest-Dispatch is published in real-time and the aggregator has to dynamically decide the updated dispatch per flexibility asset / end user. For a sequence of FlexRequests, there are iterative runs of the algorithm.
- **Offline operation:** The aggregator performs "what-if" simulation scenarios (different configurations of FlexContracts (e.g. cost, availability), expansion/modification of portfolio, different sequence of FlexRequests) to determine strategies for optimal response to future FlexRequests, creating FlexContracts and expanding its portfolio.

In order to integrate the proposed mathematical model and algorithm at TRL 5, the following assumptions are made:

- The flexibility request is a FlexRequest-Dispatch/Activation, where a positive response from the aggregator requires the dispatch/activation of flexibility assets.
- The underlying network model is not taken into account. The dispatch/activation of flexibility assets does not cause network/grid problems.
- Potential imbalance costs, if needed, will be based on available historical prices of markets (intraday market or balancing market).
- The minimum Market Time Unit (MTU) is 15 minutes. This stands for energy requested by FlexRequests and flexibility information of the FlexAssets.
- We assume simple types of FlexContracts, which allow the majority of information concerning the cost of activation of each FlexAsset for each MTU to be known a priori.

This algorithm, with some small modifications, can be used for all instances of x-DLFM architectures, as the focus is the interaction between the aggregator and its customers/end-users (i.e. B2C) and not on the interaction of the aggregator with the energy or flexibility market (i.e. B2B).

5.1.2 Algorithmic inputs and outputs and aggregator frontend ideas

The required inputs for the UCS 4.1 algorithm to run are:

- Information on the aggregator's portfolio of FlexAssets
- Reserved flexibility obligations of the aggregator
- FlexRequest-Dispatch information
- Location and Date
- Market data

The outputs of the algorithm are the following:

- Response to FlexRequest (Accept/Reject)
- Dispatched/Activated FlexAssets for fulfilling the FlexRequest
- Updated Table of aggregator's portfolio
- Aggregator and end-user profits (revenue, costs)

In the following tables, the input and output parameters of the algorithm running in the AFAT backend and the visualization of the information and results in the aggregator's GUI (AFAT frontend) are summarized.

Input parameters	Aggregator GUI in ATP	Central FG database
	Select location and date (drop-down menu for bidding zones & calendar)	
Table of FlexAssets with information for each MTU: - Baseline Consumption - Actual Consumption - Availability - Amount of Flexibility - Cost	Select Aggregator's portfolio (drop-down menu with default scenarios and customized scenarios saved by the user) Portfolio – Visualization of Table of FlexAssets	The ATP-DB API will fetch the required data (depending on the user's input) from the central DB

- Grid Location - End-user		The ATP-AFAT API will fetch the required data based on user's inputs to AFAT
	Options to modify Aggregator's portfolio - Remove FlexAsset - Add FlexAsset - Modify parameters of FlexAsset Store scenario of Aggregator's Portfolio in the DB (separate option)	The ATP-DB API will store the aggregator's portfolio to the central DB The ATP-AFAT API will fetch the required data based on user's inputs to AFAT
	Select scenario of obligations due to FlexRequests-Reserve (drop-down menu with default scenarios and scenarios saved by the user)	
	Option to modify reservation of flexibility - Remove reservation - Add reservation - Modify reservation Store scenario of reserve obligations in the DB (separate option)	The ATP-DB API will store the scenario of flexibility reservation to the central DB The ATP-AFAT API will fetch the required data based on user's inputs to AFAT
Market data for the appropriate MTUs and location (€/MWh)	Include market data for imbalance cost (checkbox) [24]	The ATP-DB API should fetch the appropriate market data (if possible) depending on the user's input
FlexRequest-Dispatch - Type of Energy - Amount of Energy - MTU - Price - Location etc.	Manage FlexRequest click button to run the algorithm with the current settings	The ATP-AFAT API will fetch the required data based on user's inputs to AFAT
Option to save current state to the DB	Save option	The ATP-DB API will store the appropriate data in the central DB.

Output parameters	Aggregator GUI in ATP	Central FG database
Response to FlexRequest (Accept/Reject)	The response to the FlexRequest is shown. The rest of the outputs are shown only if the response to the FlexRequest is "Accept"	The AFAT-ATP API will fetch the results from the AFAT to ATP.

[24] If the user selects the checkbox, the appropriate data will be retrieved from the DB. The imbalance costs will be estimated on intraday market data (adjust scheduled curve) or balancing market (penalty)

Vector of dispatch decisions per FlexAsset for relevant MTUs	A graph depicting the use/activation of each FlexAsset/end-user	The ATP-DB API will store the same results to the central DB.
Updated Table of FlexAssets	Updated Table of FlexAssets	
Deviation of energy curve of FlexAssets cf. Baseline / Actual consumption	A graph depicting the deviation of the real energy curve and the baseline one. This should also be visualized per FlexAsset/end-user.	
Aggregator's profit (revenue/cost) per MTU	A graph depicting the profit, revenue and cost of aggregator for each MTU.	

According to the information above, a draft version of the Aggregator GUI sketches has been developed by ETRA as well as the first version of the AFAT-ATP-DB APIs. Indicative screenshots and more technical details can be found in D6.2.

5.1.3 UML diagrams

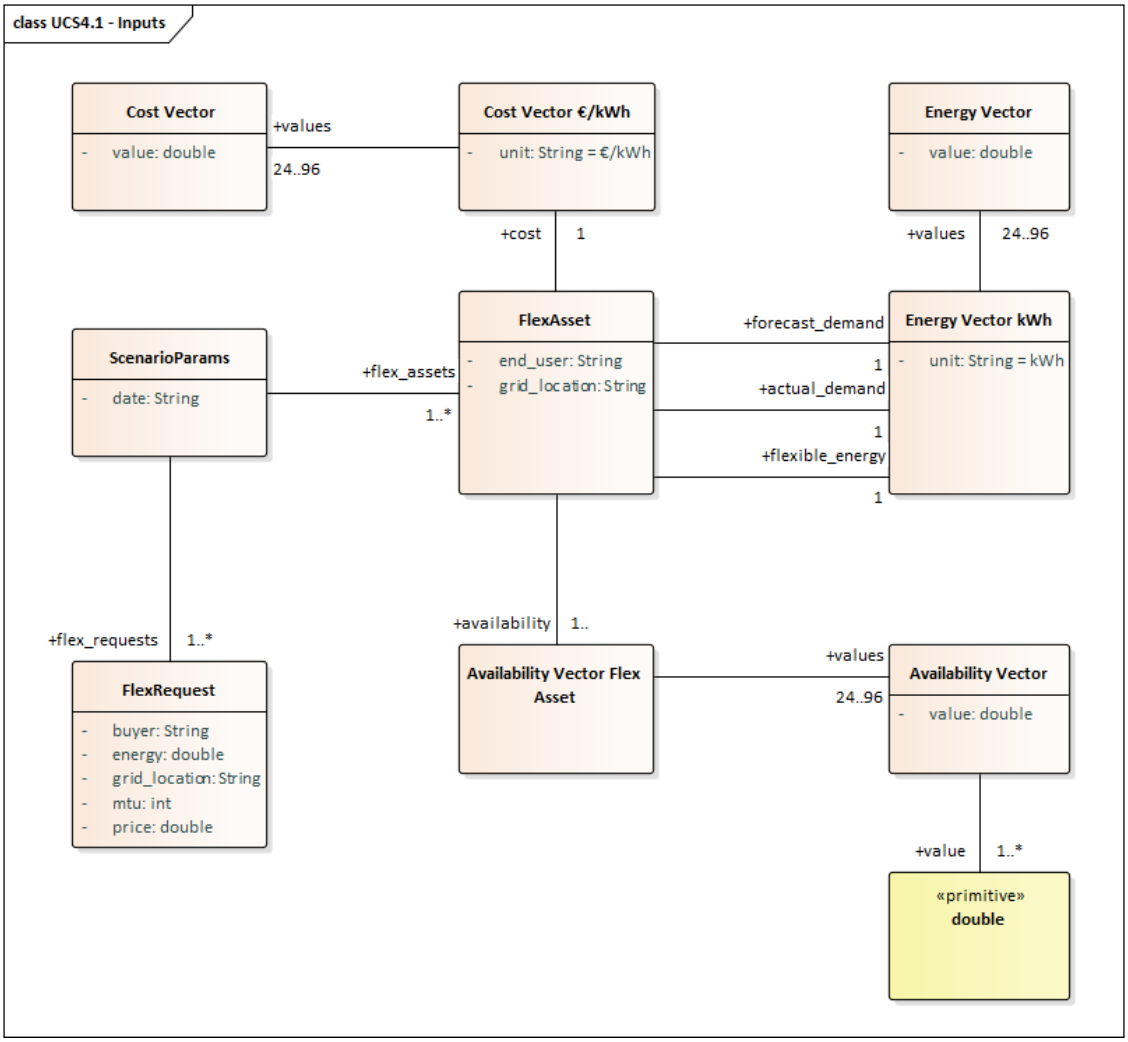

Figure 23: UML diagram for the ATP-AFAT API of UCS 4.1 (i.e. Aggregator user's inputs filled in AFAT frontend and posted to AFAT backend)

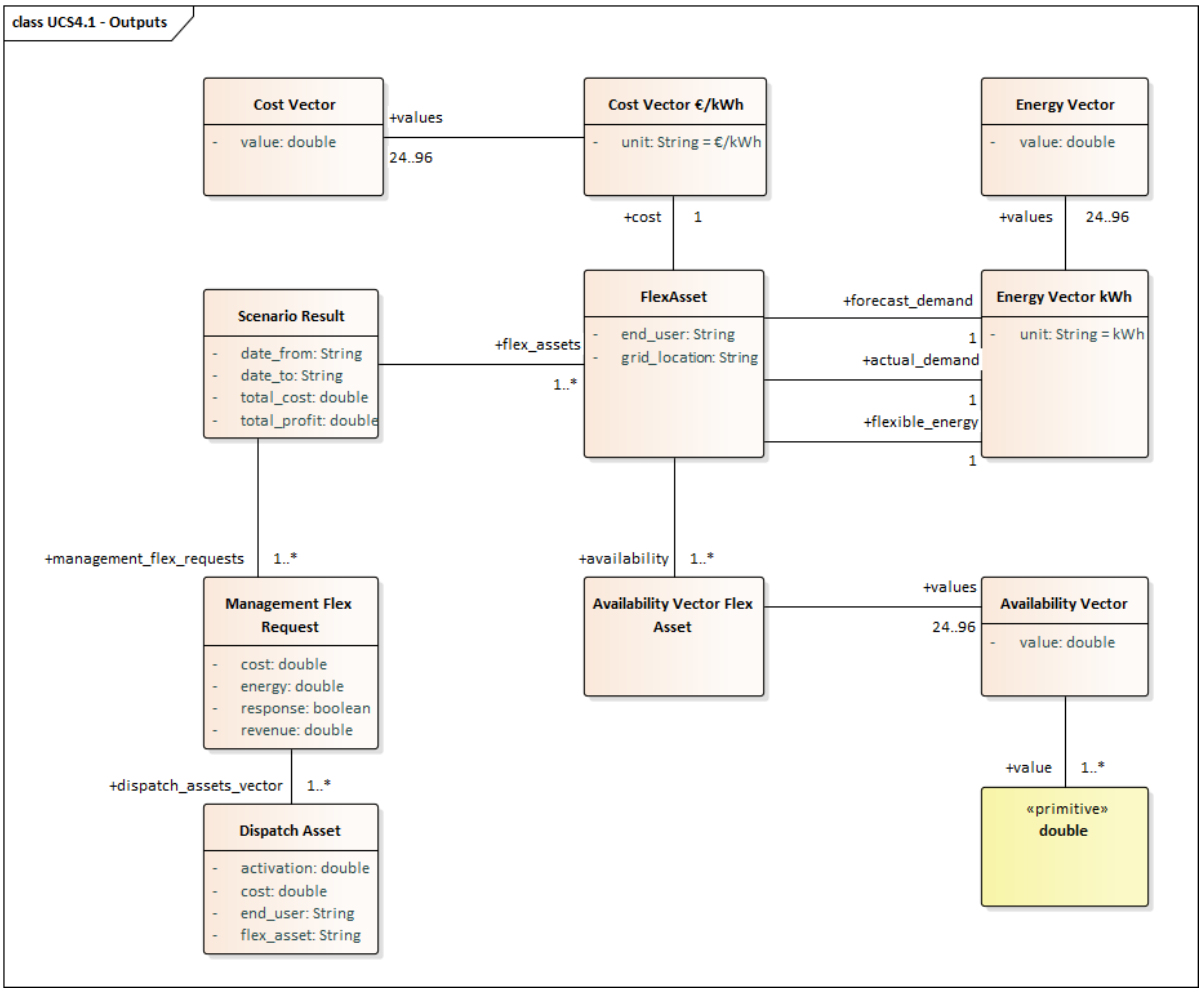

Figure 24: UML diagram for the AFAT-ATP API of UCS 4.1 (i.e. algorithmic results produced by AFAT backend and visualized in AFAT frontend)

The two figures above depict the UML diagrams for UCS 4.1. The first one is the data model representation of the input parameters described in the table above regarding the ATP-AFAT API. The second one is the data model representation of the output parameters described in the table above regarding the AFAT-ATP API.

5.2 UCS 4.3 – Create a FlexOffer

For a given timeframe (e.g. one or more timeslots ahead), the aggregator runs an automated flexibility aggregation algorithm to determine/create a FlexOffer that best represents the current status of its portfolio and submits it to the FLEXGRID ATP. This FlexOffer may be used either in the: i) TSO's balancing market (cf. "no-DLFM" architecture), or ii) proposed DLFM market operated by the FMO to solve DN-level problems.

Within WP6, the goal is to demonstrate that the aggregator user can visualize a FlexOffer and then submit (post) it in FLEXGRID ATP at a specific time instance regarding its participation in the DLFM market. Then, the FMO user will also be able to visualize this FlexOffer as well as

the DSO (i.e. FlexBuyer). If this FlexOffer is not accepted in DLFM, it may be forwarded to the TSO's balancing market[25].

5.2.1 Proposed algorithm to integrate in AFAT

Technical details about the mathematical model, algorithmic solution and performance evaluation results are provided in chapter 4 of D3.1 and chapter 3 of D3.2 (i.e. TRL 3).

In this UCS, two modes of operation for the aggregator's GUI are considered, namely:

- **Online operation** is when the aggregator wants to create a FlexOffer in real-time (in order to submit it in the ATP) based on the current availability of FlexAssets (cf. FlexContract per FlexAsset that denotes the available reserve capacity).
- **Offline operation** is when the aggregator wants to run "what-if" scenarios to see whether it is more beneficial to participate in the existing TN-level balancing market or DN-level balancing market (i.e. DLFM)[26].

In order to integrate the proposed mathematical model and algorithm at TRL 5, we have made the following assumptions:

- Single-level optimization problem (i.e. aggregator is price-taker, not price-maker).
- We do not take into account the underlying network model. We assume that all FlexOffers do not cause network/grid problems.
- We do not take into consideration time coupling constraints, complex block offers, and other types of block offer constraints.
- We assume that the aggregator gets paid for: i) the availability it offers to the TSO/DSO (€/MW), and ii) the energy that is activated at the delivery time (€/MWh).

Regarding the input to the FlexOffer creation algorithm, every FlexAsset sends to the aggregator for each timeslot and each DN location id: i) one individual FlexOffer curve for up-reserve (quantity vs. price), and ii) one individual FlexOffer for down-reserve (quantity vs. price). As of the output of the algorithm, the aggregator sends to FLEXGRID ATP for each timeslot and each DN location id: i) one aggregated FlexOffer curve for up-reserve (quantity vs. price), and ii) one aggregated FlexOffer for down-reserve (quantity vs. price).

5.2.2 Algorithmic inputs and outputs and aggregator frontend ideas

The following tables summarize the input parameters for the algorithm to run in the AFAT backend and output parameters for the results to be visualized in aggregator's GUI (i.e. AFAT frontend) respectively.

Input parameters	Aggregator GUI in ATP	Central FG database
Portfolio id (e.g. number 1, 2, 3)	Select aggregator's portfolio (drop down menu with a few portfolios, e.g. BADENOVA	The ATP-AFAT API will fetch the required data

[25] ETRA will develop this API to emulate this operation. ETRA will also create a tab to demonstrate a "TSO user view" in which the TSO user can visualize this FlexOffer that has been forwarded (cf. UCS 1.4).

[26] UCS 2.3 deals with this problem in a similar way for the ESP user, so we can use the UCS 2.3 algorithm for the offline mode of operation, too.

	data, Cyprus retailer's data, Greek retailer's data[27])	based on aggregator user's inputs to AFAT. Then, the DB-AFAT API will fetch the required historical data from the central DB to the AFAT.
Set of FlexAsset ids	Select a set of FlexAssets from a list (i.e. scroll down and manually select the FlexAssets). Default option is to have a checkbox to select all FlexAssets for a given portfolio selected above.	
Start date time + end date time	Select time interval 'X' date to 'Y' date (cf. calendar)	
Time granularity id	Select from a drop-down menu (15-minute, 1-hour, 1-day)	
Location id (DSO area)	List of DSO areas for the user to select (default option is that the root/whole DSO area is selected, where the entire FlexAsset portfolio can be used)[28]	
Individual FlexOffer from each individual FlexAsset[29]	- one FlexOffer curve for up-reserve (quantity vs. time)[30] - one FlexOffer for down-reserve (quantity vs. time) - location id No input is required by the aggregator user. We will have one static FlexOffer per FlexAsset stored in the central DB.	
Option to store data in the central FLEXGRID database	Checkbox that can be either checked or not.	
Past scenario results to view in AFAT GUI (ATP)	The aggregator user can select a past scenario to view it in the AFAT GUI.	

Output parameters	Aggregator GUI in ATP	Central FG database
One FlexOffer curve per product (i.e. up-reserve and down-reserve)	A graph per product that depicts: - Quantity offered vs. time (at a given price) - Quantity offered vs. price (at a given timeslot) The FMO and DSO users should also be able to visualize these FlexOffer curves on their own GUIs.	The AFAT-ATP API will fetch the results from the AFAT to ATP. The ATP-DB API may store the same results to the central DB.

[27] We assume that each aggregator has already registered its own FlexAssets (i.e. portfolio) in the FLEXGRID ATP. ETRA will elaborate on the existing NODES API (https://nodes-demo.azurewebsites.net/swagger/index.html#/).

[28] Following up the NODES paradigm, we can have an hierarchical structure. For example: i) id 1 for the entire DSO area (default area), ii) id 1.1 or 1.2 or id 1.3 for a DSO sub-area, iii) id 1.1.1 for a specific LV-feeder experiencing network problems at a distribution network edge.

[29] If we want to make it more complex (assume more complex FlexContract), we may assume that each individual FlexAsset sends for each timeslot several quantity vs. price (tuples).

[30] For example, if the aggregator user selects a time interval of one day (cf. day-ahead) and 1-hour time granularity, the individual FlexOffer will be a vector of 24-hourly quantities offered vs. time (for a given price).

Potential revenues (€) for the selected scenario (if FlexOffer is accepted)	A graph per selected market (TN- or DN-level reserve market) that depicts: - Expected revenues (€) vs. time Only the aggregator user can visualize these results.	The AFAT-ATP API will fetch the results from the AFAT to ATP. The ATP-DB API may store the same results to the central DB.

5.2.3 UML diagrams

The following two figures depict the UML diagram for UCS 4.3. The first one is the data model representation of the input parameters described in the table above regarding the ATP-AFAT API. The second one is the data model representation of the output parameters described in the table above regarding the AFAT-ATP API.

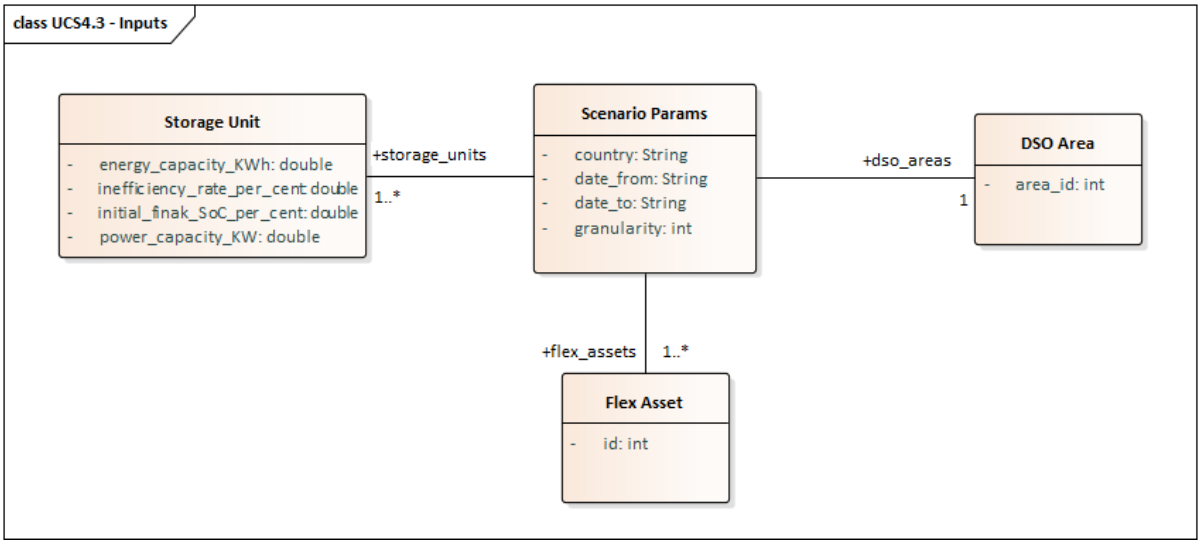

Figure 25: UML diagram for the ATP-AFAT API of UCS 4.3 (i.e. Aggregator user's inputs filled in AFAT frontend and posted to AFAT backend)

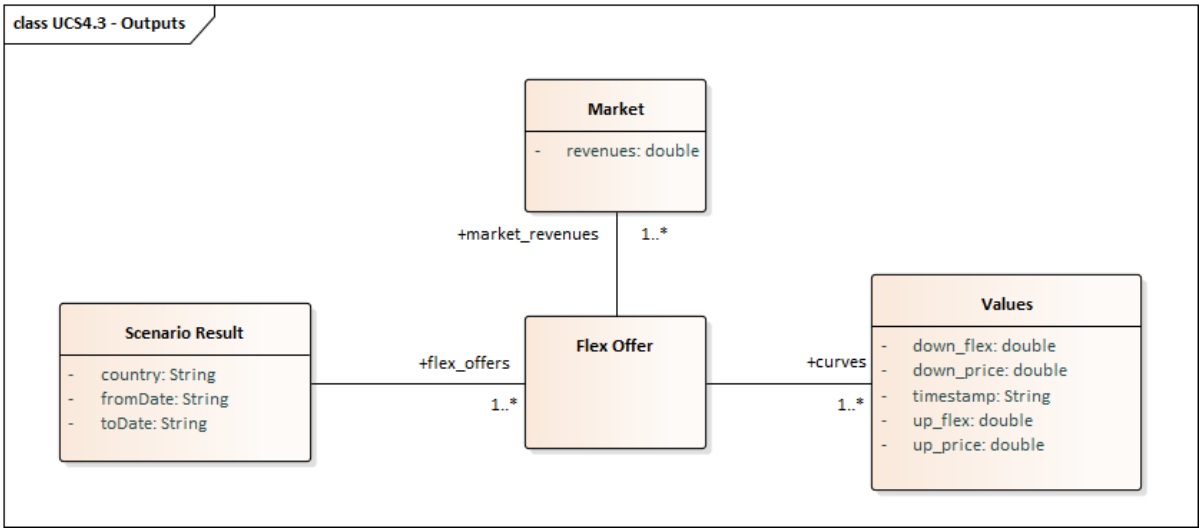

Figure 26: UML diagram for the AFAT-ATP API of UCS 4.3 (i.e. algorithmic results produced by AFAT backend and visualized in AFAT frontend)

5.3 UCS 4.2 – Manage a novel B2C flexibility market

In this UCS, we assume that a DSO has made a FlexRequest (i.e. quantity vs. price curve for each given timeslot). The aggregator user wants to run various "what-if" simulation scenarios (i.e. offline operation) to determine better ways (via retail pricing schemes) to operate a novel B2C flexibility market, in which the end energy prosumers compete with each other. In other words, the aggregator runs a retail pricing algorithm to test and evaluate the impact that new FlexContracts (with its end users) would have on several KPIs such as:

- aggregator's revenues,
- aggregated end users' welfare,
- quantity of flexibility offered to the system,
- individual end user's welfare.

5.3.1 Proposed algorithm to integrate in AFAT

Technical details about the mathematical model, algorithmic solution and performance evaluation results are provided in chapter 5 of D3.1 and chapter 4 of D3.2 (i.e. TRL 3).

In this UCS, only one operation mode for the aggregator's GUI is considered, namely:

- **Offline operation:** The aggregator user runs various "what-if" simulation scenarios via running an advanced retail pricing algorithm (Behavioral Real Time Pricing – B-RTP) to identify how it can recommend a new (more beneficial) FlexContract to a set of end energy prosumers. Only the aggregator user will be able to visualize the results.

In order to integrate the proposed mathematical model and algorithm at TRL 5, we have made the following assumptions:

- We assume that the end energy prosumers dispose the required ICT infrastructure and equipment to automatically create individual FlexOffers and respond to the iterative pricing signals sent by the aggregator.
- We assume that the aggregator has already registered all its individual FlexAssets in the ATP (like it is done in the NODES platform) with all the required FlexAsset specifications.
- We do not take into account the underlying network model. We assume that all FlexOffers do not cause any network/grid problems, which is a rational assumption given the fact that a DSO has already made a FlexRequest and this is an input to our algorithm.
- We follow a decentralized optimization approach (in contrary with the UCS 4.1, in which a centralized optimization is adopted).
- We assume historical energy consumption and production data. So, we run exhaustive offline simulation scenarios to identify potential energy prosumers, who are eager to become more flexible (i.e. select a more beneficial FlexContract) in order to earn more financial rewards in the future.
- We assume that the DSO has made the FlexRequest based on existing load/generation forecast data that it already acquires for its distribution network.

5.3.2 Algorithmic inputs and outputs and aggregator frontend ideas

The following tables summarize the input parameters for the algorithm to run in the AFAT backend and output parameters for the results to be visualized in aggregator's GUI (i.e. AFAT frontend) respectively.

Input parameters	Aggregator GUI in ATP	Central FG database
Portfolio id (e.g. number 1, 2, 3)	Select aggregator's portfolio (drop down menu with a few portfolios, e.g. BADENOVA data, Cyprus retailer's data, Greek retailer's data)[31]	The ATP-AFAT API will fetch the required data based on aggregator user's inputs to AFAT backend (i.e. algorithm). The DB-AFAT API will fetch the required historical data from the central DB to the AFAT backend (i.e. algorithm).
Set of prosumer ids	Select a set of end energy prosumers from a list (i.e. scroll down and manually select the end prosumers)[32]	
Start date time + end date time	Select time interval 'X' date to 'Y' date (cf. calendar)	
Time granularity id	Select from a drop-down menu: 15-minute, 1-hour, 1-day	
Timeframe id	Select from a drop-down menu: (default, Mon-Fri ONLY, Weekend ONLY, night hours ONLY, Peak hours ONLY)	
FlexRequest: a vector of price/quantity tuples for each 15-min timeslot	Select a FlexRequest from a drop-down menu [33] or retrieve the FlexRequest that is manually created by the DSO user in ATP[34]	
FlexOffer: a vector of price/quantity tuples for each 1-hour timeslot per FlexAsset	No input is required by the aggregator user. We will have one static Flexoffer per end user stored in the central DB[35]	
Storage unit specifications (fill in parameters in the GUI)	Fill in (for every storage unit – the user can add several units): - power capacity (KW) - energy capacity (KWh) - inefficiency rate (%) - initial/final SoC (%) - location id	
Curtailable load specifications (fill in parameters in the GUI)	Fill in (for every load unit – the user can add several units): - power consumption (KW)	

[31] 15-minute granularity of energy consumption and RES data is available

[32] All energy consumption and production datasets are stored in the central DB.

[33] There is a set of indicative FlexRequests, which are stored in the central FG database.

[34] Once the DSO user creates a FlexRequest in the respective DSO GUI (cf. UCS 1.1-1.3), then this FlexRequest is automatically posted in the aggregator/ESP user's GUI.

[35] We do not need to have a sketch for this in the GUI. Once the aggregator user selects the set of prosumer ids (see above), then the algorithm will retrieve the static FlexOffers (one per end user) from the central DB. This static FlexOffer is a mathematical interpretation of each end user's FlexContract.

FlexContract id	Select manually 1 or more retail pricing algorithms from a drop-down menu to be compared: - Fixed pricing, - Real-Time Pricing (γ=0), - Behavioral RTP (γ=1), - Behavioral RTP (γ=0.5), - Behavioral RTP (γ=1.5)	
Gamma 'γ' parameter?	Fill in a number in a required field. This 'γ' value can be 0<γ<2.[36]	
Profit margin parameter 'p' (optional)	Fill in a number in a required field[37]	
Option to store data in the central FLEXGRID database	Checkbox that can be either checked or not.	The ATP-DB will store the algorithmic results in the central DB.
Scenario results to view in AFAT GUI (ATP)	The aggregator user can select a past scenario to view it in the AFAT GUI.	The DB-ATP will retrieve the data from the central DB.

Output parameters	**Aggregator GUI in ATP**	**Central FG database**
Flexibility revenues	A graph per selected pricing scheme that depicts: - flexibility revenues vs. 'γ'?	The AFAT-ATP API will fetch the results from the AFAT to ATP. The ATP-DB API will store the same results to the central DB (if the related checkbox has been checked).
Aggregated end Users' Welfare (AUW)	A graph per selected pricing scheme that depicts: - aggregated users' welfare vs. 'γ'?	
Quantity of aggregated flexibility offered	A graph per selected pricing scheme that depicts: - quantity of flexibility offered vs. 'γ'?	
Welfare per end user *(optional)*	A histogram per selected pricing scheme that depicts: - 'X' axis: set of selected end users - 'Y' axis: UW with B-RTP (γ)/ UW with RTP (γ=0)	

[36] When γ=0, we have the RTP model in which all end users get the same reward in €/flexibility unit, even though some of the them did not contribute anything in the FlexRequest. When γ=1, we have a fully personalized RTP scheme, in which the flexible end users get rewarded according to each one's contribution, while inflexible end users do not get any reward. When γ>1, then the inflexible end users get penalized, because they did not contribute anything in a case of a critical FlexRequest.

[37] This parameter can be any number between 0 and 1 and represents the percentage of flexibility revenues that will get the aggregator as profit. The residual percentage of aggregator's revenues will be shared among end users. In the extreme case, in which p=0, all flexibility revenues are shared among the end users.

5.2.3 UML diagrams

The following two figures depict the UML diagram for UCS 4.2. The first one is the data model representation of the input parameters described in the table above regarding the ATP-AFAT API. The second one is the data model representation of the output parameters described in the table above regarding the AFAT-ATP API.

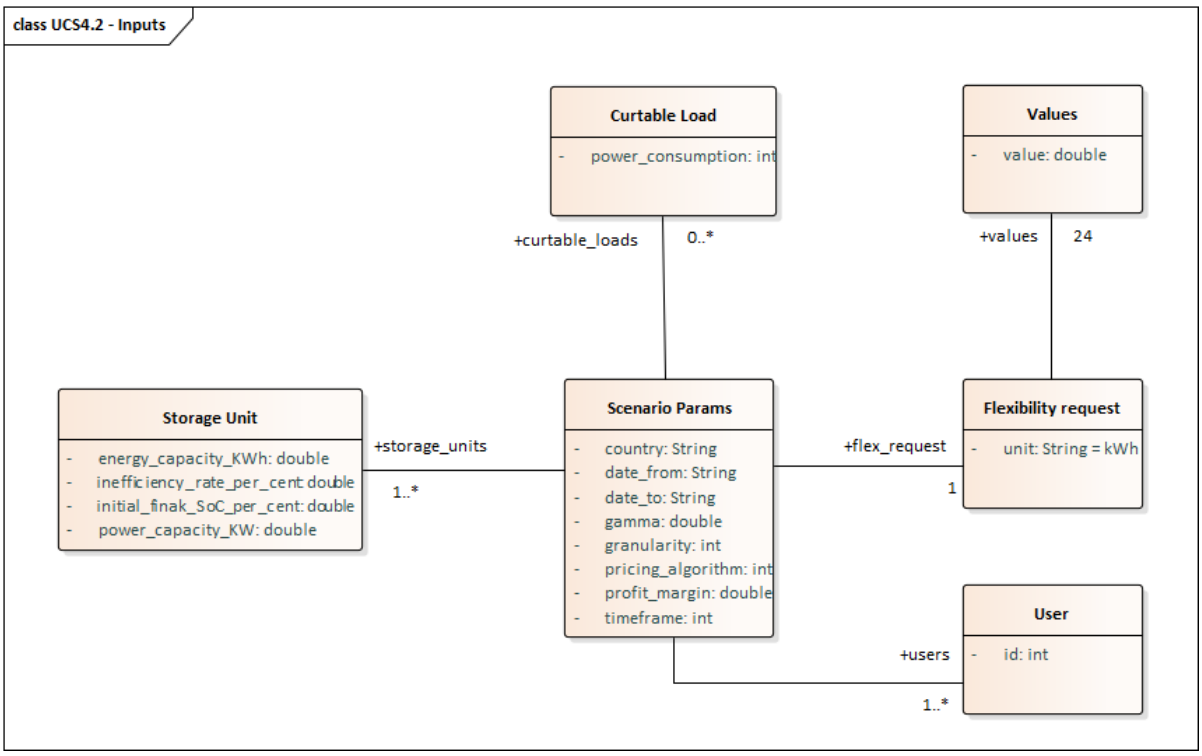

Figure 27: UML diagram for the ATP-AFAT API of UCS 4.2 (i.e. Aggregator user's inputs filled in AFAT frontend and posted to AFAT backend)

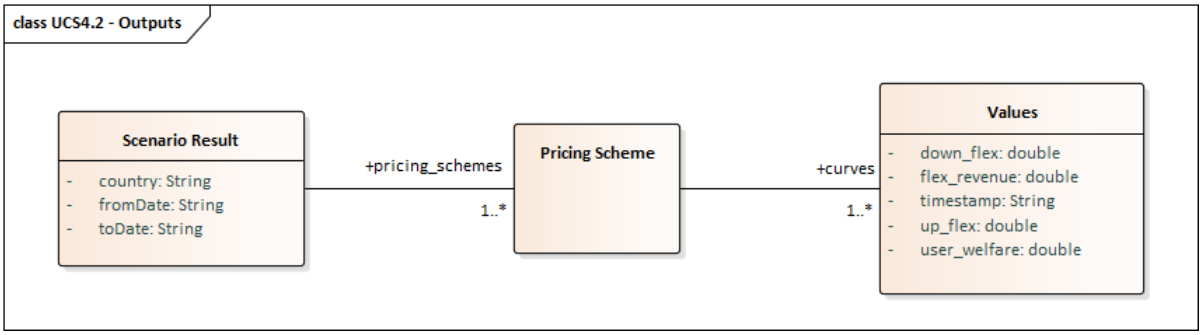

Figure 28: UML diagram for the AFAT-ATP API of UCS 4.2 (i.e. algorithmic results produced by AFAT backend and visualized in AFAT frontend)

5.4 Sequence diagram for communication among ATP frontend and AFAT backend

As already discussed, we will integrate three UCS in the Automated Flexibility Aggregation Toolkit (AFAT). The S/W architecture follows a modular-by-design approach, which allows the different S/W modules to be developed on a standalone basis by the two involved partners (i.e. UCY, ICCS) and communicate with each other via well-defined and fine-grained APIs. The following figure describes in five main steps the process that will be followed[38]. There are three main S/W components that will be developed, namely:

- **ATP GUI or else AFAT frontend:** this will be developed by ETRA.
- **AFAT backend:** this will be developed collaboratively by UCY (cf. UCS 4.1) and ICCS (cf. UCS 4.2 and 4.3).
- **FLEXGRID Central Database:** this will be developed by ETRA, while real-life input data will be provided by each one of the two research partners in collaboration with industrial partners, too.

First of all, the aggregator user will login the FLEXGRID ATP, will be authenticated through a single sign-on process and then will be redirected to the main aggregator's GUI. Via the AFAT application, the aggregator user will be able to visualize, configure and manage its end user portfolio. Moreover, the aggregator user will be able to run three main algorithms in order to be able to make optimally manage a FlexRequest (cf. UCS 4.1), create a FlexOffer (cf. UCS 4.2) and manage a novel B2C flexibility market (cf. UCS 4.3).

For each one of the three algorithms, there will be a tab in the AFAT frontend. Once the aggregator user clicks on one tab, s/he will be able to configure/customize/fill in the input parameters that are needed for each algorithm to be able to run. Once the aggregator user clicks on the "Run algorithm" button, step 1 process will be followed. More specifically, the API client that resides at the AFAT frontend will automatically gather all input parameters and will send them to the API server that resides at the AFAT backend.

After the AFAT backend receives the input parameters, the next step is to request for the required input data from the FLEXGRID central database (DB). More specifically, an API client that resides at AFAT backend request for input data from an API server residing at the central DB. In step 3, the input data is retrieved, and now the algorithm can be executed.

Once the algorithm produces the results, these output parameters will be automatically gathered by the AFAT-ATP API and will be sent to the AFAT frontend so that the aggregator user can visualize the results in a comprehensive and user-friendly manner. The final step (i.e. step 5) is for the aggregator user to understand the results and if s/he is interested in further elaborating them, then s/he can optionally select to store them in the central DB in order to be able to retrieve, visualize and possibly compare them with other results in the future.

[38] More technical details about the steps depicted in Figure 29 are provided in FLEXGRID D6.2.

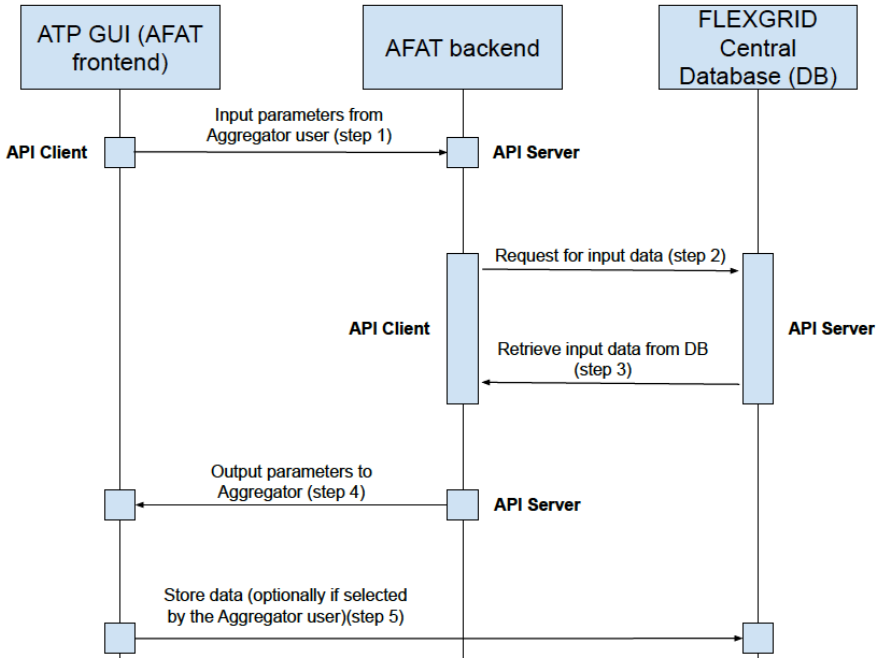

Figure 29: Sequence diagram for communication among ATP frontend, AFAT backend and central database

6 Conclusions and next steps

Conclusively, during the next months, FLEXGRID consortium will elaborate on the data modeling work presented in this deliverable towards delevering the 1st version of FLEXGRID ATP in Month 24, which corresponds to project's milestone #8 "Release of the first integrated FLEXGRID system prototype".

During the Period 1 review meeting, which will take place on 22nd June 2021, the FLEXGRID consortium will demonstrate the first algorithm that is integrated in FLEXGRID ATP. In other words, the ESP user will be able to fill in all simulation parameters, execute the algorithm and after the algorithm's run, the results will be automatically shown in the FLEXGRID ATP. For this purpose, the first REST API server and REST API client will be developed and the communication with the central FLEXGRID database will be demonstrated, too. After M21, a similar S/W integration process will be followed for all UCS and thus algorithms. Once a new algorithm has been exhaustively tested and validated at TRL 3 by a research partner in the context of WP3, WP4 and WP5, the next step will be the S/W integration phase, which is ETRA's main responsibility. Two rounds of S/W implementation and integration will take place in order to timely identify any possible technical problems and apply pre-agreed contingency measures.

As also depicted in the figure below, the delivery of "alpha" version of FLEXGRID ATP is scheduled for M24, while the "beta" version is expected to take place in M33.

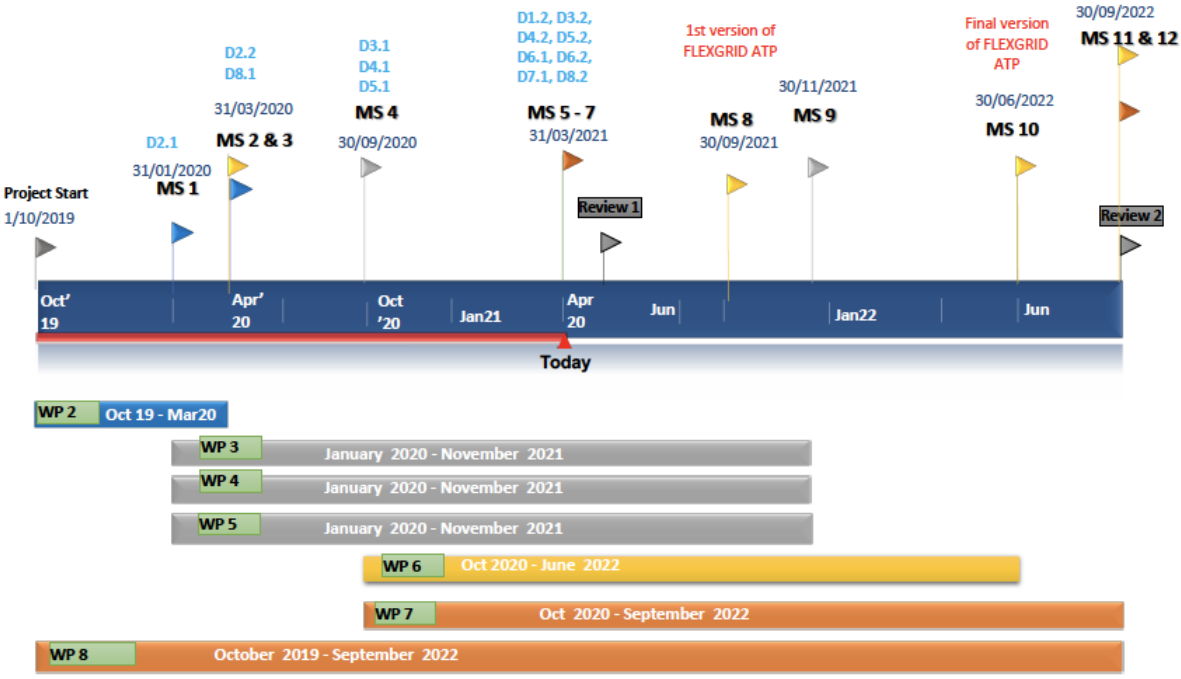

Figure 30: Current FLEXGRID project's timeline schedule (MS 6 has been accomplished)