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

The overall FLEXGRID architecture design, high-level model and system specifications

Deliverable D2.2

Document Information

Scheduled delivery	31.03.2020
Actual delivery	31.03.2020
Version	Final
Responsible Partner	ETRA

Dissemination Level

PU Public

Contributors

Victor Lacort (ETRA), Aurelio Lazaro Chueca (ETRA), Prodromos Makris (ICCS), Nikolaos Efthymiopoulos (ICCS), Georgios Tsaousoglou (ICCS), Konstantinos Steriotis (ICCS), Bryan Pellerin (SIN), Matin Bagherpour (NPC), Robert Gehrcke (NPC), Steinar Rune Eriksen (NODES), Enno Boettcher (NODES), Gesa Milzer (NODES), George Georghiou (UCY), Venizelos Efthymiou (UCY), Hrvoje Pandzic (UNIZG-FER), Domagoj Badanjak (UNIZG-FER), Tonci Tadin (HOPS), Heni Radanovic (HOPS), Malte Thoma (BDNV), Spyros Chatzivasileiadis (DTU), Elea Marie Prat (DTU), Mihai Calin (AIT), Helfried Brunner (AIT)

Internal Reviewers

Manos Varvarigos, Prodromos Makris (ICCS)
Matin Bagherpour, Robert Gehrcke (NPC)
Gesa Milzer (NODES)

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
CA	Consortium Agreement
D	Deliverable
DoA	Description of Action
EC	European Commission
GA	Grant Agreement
MS	Milestone
TMT	Technical Management Team
WP	Work Package
UC	Use Case
HLUC	High Level Use Case
UCS	Use Case Scenario

Technical terminology

Acronym	Definition
ATP	Automated Trading Platform
AFAT	Automated Flexibility Aggregation Toolkit
BB	Building Block
BMC	Business Model Canvas
CBAC	Content Based Access Control
DAD	Day-Ahead Dispatch
DB	Data Base
DER	Distributed Energy Resource
DLD	Distribution Level Dispatch
DLFM	Distribution Level Flexibility Market
DMP	Data Management Plan
DND	Distribution Network Dispatch
DNDA	Distribution Network Dispatch Algorithm
DNPA	Distribution Network Payment Algorithm
DR	Demand Response
DSM	Demand Side Management
DSO	Distribution System Operator
EC	Energy Community
ECC	Energy Consumption Curve
EE	Energy Efficiency
EP	Energy Program
ESCO	Energy Services Company
FST	FlexSupplier's Toolkit
FMCT	Flexibility Market Clearing Toolkit
HVAC	Heating, Ventilation and Air Conditioning

ICT	Information and Communications Technology
IPR	Intellectual Property Rights
JSON	Javascript Object Notation
KPI	Key Performance Indicator
MCA	Market Clearing Algorithm
MDG	Model Driven Generation
MG	Micro-Grid
OAS	Open API Specification
PBAC	Policy Based Access Control
P-DLFM	Proactive Distribution Level Flexibility Market
RA	Reference Architecture
RBAC	Role Based Access Control
R-DLFM	Reactive Distribution Level Flexibility Market
RES	Renewable Energy Sources
REST	Representational State Transfer
RTP	Real Time Pricing
SAP	System Analysis Phase
SGAM	Smart Grid Architecture Model
SWOT	Strengths Weaknesses Opportunities Threats
TLD	Transmission Level Dispatch
TLMP	Transmission Network Locational Marginal Price
TNDA	Transmission Network Dispatch Algorithm
ToU	Time of Use
URI	Universal Resource Identifier
UUID	Universally Unique Identifier
VPC	Value Proposition Canvas
WM	Wholesale Market

Table of Contents

List of Figures and Tables

List of Figures

List of Tables

Document History

This deliverable includes the research outputs of Task 2.4. It includes the design of the overall FLEXGRID system architecture and the high-level technical specifications of each subsystem.

Table 1: Document History Summary

Revision Date	File version	Summary of Changes
11/12/2019	v0.1	Draft ToC circulated within the entire consortium.
8/1/2020	v0.2	All partners commented on the draft ToC structure.
20/1/2020	v0.3	Final ToC has been created by ETRA and has been circulated to all partners together with all writing task delegations for 1st round of contributions.
2/3/2020	v0.4	All partners contributed 1st round inputs, ETRA integrated text and sent this version to be reviewed by ICCS, NODES & NPC.
10/3/2020	v0.5	Reviewed version was commented by each partner, integrating proposed changes.
24/3/2020	v0.6	ETRA integrated 2nd round contributions.
24/3/2020	v0.7	ETRA integrated changes from all partners providing this version to all and ICCS, NPC & NODES started 2nd revision.
27/3/2020	v0.8	ETRA addressed 2nd revision comments.
27/3/2020	v0.9	Pre-final version circulated to all partners for final minor changes.
31/3/2020	v1.0	ICCS (coordinator) makes final enhancements/changes based on all partners' final comments and submits in ECAS portal.

Executive Summary

This report is the second official deliverable of H2020-GA-863876 FLEXGRID project dealing with the initial description of the FLEXGRID's software architecture. It includes several novel smart grid architecture models that incorporate the operation of Distribution Level Flexibility Markets (DLFMs) as R&I motivation for FLEXGRID. It also addresses the SGAM reference architecture based on CEN-CENELEC-ETSI Smart Grid Coordination Group for contextualizing the S/W architecture. The technical specifications, design decisions and implementation directives for API development in FLEXGRID S/W platform for WP6 are stated as well. Moreover, a brief description of the internal architecture for the different modules conforming the platform (i.e. ATP, AFAT, FST, FMCT) are included. Finally, it presents the first three layers of SGAM diagrams for the relevant Use Case Scenarios to be considered on the FLEXGRID S/W platform and will be developed at TRL 5. In the Appendix, the expected input and output data that will be used by the different S/W modules, that serve as the first step for designing the Data Model that will be developed in WP6 can be found.

With this D2.2 deliverable, WP2 is now finished and Milestone#3 has been achieved.

1. Introduction

1.1 Purpose of the document

This document presents the **FLEXGRID Architecture** specified using the Smart Grid Reference Architecture Model (SGAM) from CEN-CENELEC-ETSI that considers wider European initiatives in this domain in terms of communication protocol and data modelling standards.

The purpose of this document is to provide the tools to model the architecture of an interoperable, secure and flexible architecture, which will consider smart grids addressing active distribution networks with high penetration of renewable energy sources (RES). FLEXGRID proposes several flexibility market architectures in order to allow DSOs (which we consider as the core of the future smart grid architecture) to host in a high degree of Distributed Energy Resources (DERs) and interact in an efficient way with energy sector stakeholders and existing energy markets in a beneficial way for all market participants. FLEXGRID also facilitates Energy Service Providers (ESPs) to optimally plan and operate their energy services. Finally, independent aggregators can efficiently aggregate flexibility units from numerous end users and retailers can automatically derive advanced flexibility contracts with their end clients. The final goal of this document is to give an overview of the major architectural concepts pursued in the FLEXGRID project and to show how they are aligned with the project objectives.

The FLEXGRID S/W architecture is typically specified across the five **SGAM interoperability layers**: Business, Function, Information, Communication, and Component layers. The approach taken in FLEXGRID focuses on the technical description of the S/W architecture and the specification of the system design—in terms of underlying infrastructure, components, communication protocols and data model standards— while the functional information is provided in a textual form. This way, the design is modelled upon the first three technical layers, namely “component”, “communication” and “information” layers.

In addition to the S/W architecture model, new data models have been introduced to address specific requirements of FLEXGRID services. A detailed description of these data models is initialized and aligned with the information layer of the architecture model, based on the detailed Use Cases description from D2.1.

1.2 Scope of the document

The document covers the modelling of the project’s S/W architecture using the European SGAM framework, in a systematic manner that is also closely related to the architectures already developed in other H2020 projects, such as INCREASE, STORY, WISEGRID and NOBEL GRID.

Standards and interoperable data models to be used in the context of the FLEXGRID tools are analysed. Furthermore, new data models have been considered in order to allow new reasoning mechanisms in the process of automated decisions and collective awareness.

1.3 Structure of the document

The document starts with the definition of a few main novel smart grid architecture models that incorporate the operation of Distribution Level Flexibility Markets (DLFMs), which is the core architectural proposal of FLEXGRID project (Section 2). Moreover, the document comprises an overview on the S/W architecture definition (Section 3), introducing the Smart Grid Architecture Model, which will be used as the framework for modelling in a systematic and unified way both the FLEXGRID tools and the Use Cases. It then continues with technical specifications about mechanisms for the data interoperability (Section 4). The subsequent section 5 describes in detail the architecture of the FLEXGRID tools following SGAM diagram modelling for the relevant Use Case Scenarios (UCS). As an appendix, the related Data Model that will be further elaborated in the context of WP6 work is initially described.

2. Proposed distribution network flexibility market architectures

In this section, the definition of a few novel smart grid architecture models that incorporate the operation of Distribution Level Flexibility Markets (DLFMs) (i.e. the core architectural proposal of FLEXGRID project) takes place.

2.1 FLEXGRID R&I motivation

The goal of FLEXGRID is to enable energy sector stakeholders (DSOs, TSOs, RES producers, retailers, energy/flexibility service providers) to: i) easily and effectively create advanced Energy Services (ESs), ii) interact in a dynamic and efficient way with their environment (electricity grid) and the rest of the stakeholders, and iii) automate and optimize the planning and operation of their ESs. In this way, FLEXGRID envisages secure, sustainable, competitive, and affordable ESs. In order to facilitate bottom-up investments, modern smart grids have to cope with the challenging distribution network management. Thus, FLEXGRID develops flexibility market architectures, which allow DSOs to: a) integrate, through an open market, Distributed Energy Resources (DERs) in a scalable way and, b) efficiently interact with all energy sector stakeholders. In this way, several market stakeholders from both FlexDemand (i.e. DSOs/TSOs) and FlexSupply sides will benefit from FLEXGRID services. More specifically, in this section, we present the predominant distribution network flexibility market architectures that FLEXGRID will develop. In addition, we discuss their advantages and disadvantages in terms of efficiency and compatibility with today's smart grid and energy markets operation.

The integration of large amounts of DERs such as PV/Wind generation, Electric Vehicles (EVs), Energy Storage Systems (ESS) and Demand Side Management (DSM) tools in distribution networks, poses new challenges and opportunities for the power sector according to the recently released Clean Energy Package[1]. Relying only on grid investments to cope up with this new situation in Power Systems would be very expensive and consequently inefficient. Moreover, distribution networks become the core of smart grids and their physical constraints constitute the barrier in terms of ES cost and stability. In more detail, the volatile and unpredictable distributed energy generation that high RES penetration introduces, creates additional challenges that relate to congestion, reactive power instability and voltage issues in the distribution grid. In this new landscape, FLEXGRID focuses in four major research threads.

The first research thread examines in depth the operation of the existing energy markets and the evolution of energy market architectures. It unfolds around the development of advanced market clearing algorithms able to adequately model the underlying grid and ensure market efficiency.

[1] 'Clean Energy package', https://ec.europa.eu/info/news/clean-energy-all-europeans-package-completed-good-consumers-good-growth-and-jobs-and-good-planet-2019-may-22_en, accessed 03 March 2020.

The second research thread of FLEXGRID spans around the efficient aggregation of end user flexibility assets (e.g. ESS, DSM, EVs, etc.) from Energy/Flexibility Service Providers and their optimal and parallel use in multiple energy markets.

The third research thread focuses on the monitoring of the transmission and the distribution networks in smart grids and in resolving the problems that high RES penetration introduces such as congestion issues and voltage control.

The fourth research thread relates to the optimal operation of assets that ESPs (i.e. FSPs, Producer and consumer aggregators) dispose and the advanced planning of their investments according to a careful examination of markets and competition.

In the rest of this section and deliverable, we focus on the first research thread and we present three different flexibility market architectures. We emphasize in the trade-off among: i) the level of compatibility of the proposed architecture with the existing smart grid and energy markets' architecture, ii) the efficiency or else potential gains for various energy sector stakeholders (e.g. ESPs/FSPs, producers, retailers), and iii) social welfare maximization.

Note: All proposed FLEXGRID architectural variants are based on the assumption of future high RES penetration scenarios. One architecture is close to the existing smart grid, markets' architecture and regulatory framework (cf. section 2.2.1), while the other two are mostly research proposals based on recently released Clean Energy Package guidelines (cf. sections 2.2.2 and 2.2.3).

2.2 Proposed FLEXGRID market architectures

FLEXGRID proposes three main market architecture variants. The first one acts *reactively* to the existing energy markets and in this way sacrifices efficiency, but on the other hand it is compatible with today's grid and markets' operation. The second one ensures an a-priori feasible dispatch of FlexAssets that reside at the distribution network by proposing a *proactive* distribution network aware market. The third architecture assumes the evolvement of the existing markets (day ahead and balancing), but offers the maximum possible smart grid efficiency by maximizing the system's social welfare and thus bringing benefits for all involved actors in the smart grid ecosystem.

2.2.1 A wholesale market compatible and reactive distribution network aware flexibility market architecture

The objective of this distribution network level flexibility market (DLFM) architecture is to be compatible with the existing regulatory framework. This is done by interacting with the existing Wholesale Market (WM) taking the Day-Ahead Dispatch (DAD) as given and then trying to deal with distribution level imbalances via the proposed DLFM. Moreover, it is capable of coping with forecast inaccuracies in energy production and consumption in assets connected to the distribution and transmission network.

The drawback of this approach is the possibility of infeasible distribution of DAD at the distribution network level due to the lack of (low cost) flexibility assets. This means that local RES assets cannot be fully utilized leading thus to undesired high RES spillage. This happens mainly due to the fact that distribution network constraints are not taken into consideration in today's market clearing models and distribution network is just seen as a "copper plate"[2]. All these lead to unsatisfied producers/consumers and sub-optimal social welfare. Furthermore, in cases in which DAD is modified, spot market price in the transmission level has to be paid leading thus to undesired high re-dispatch costs. Finally, the absence of joint optimization between transmission and distribution leads to lower levels of social welfare, which deteriorates the financial sustainability of all stakeholders.

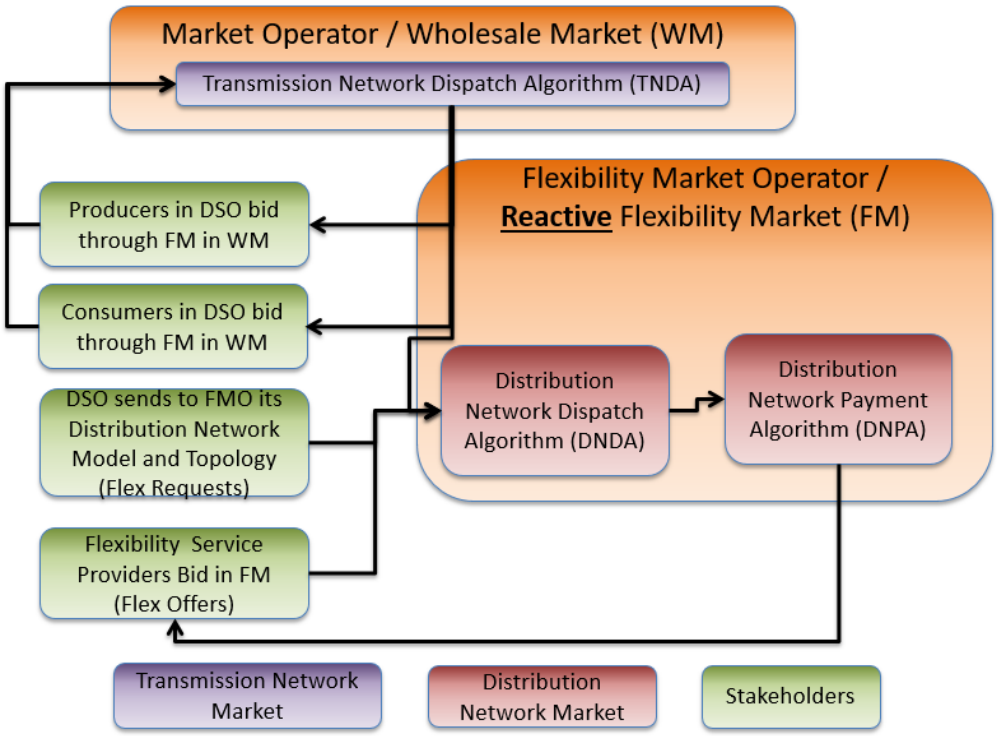

Figure 1: Reactive Distribution Level Flexibility Market (R-DLFM)

In this architecture, the steps of the process that the reactive flexibility market follows are:
Step 1: Flexibility Market Operator (FMO) takes as input the WM DAD that is composed from the power flows in the coupling point with TSO and the dispatch that concerns prosumers in its distribution network.
Step 2: DSO sends information that suffice to model its distribution network to FMO
Step 3: Flexibility providers (e.g. aggregators/ESPs) connected to the DSO send their flexibility asset bids to FMO.
Step 4: FMO generates Distribution Network Dispatch (DND) through the execution of an optimization algorithm noted as Distribution Network Dispatch Algorithm (DNDA), which:

- respects distribution network constraints (i.e. a) active/reactive power balance, b) mitigates network congestion, c) accommodates voltage control)

[2] FLEXGRID's proposal is to model a Distribution Level Flexibility Market (DLFM), in which distribution network constraints are taken into consideration.

- implements fundamental economic rules, which means that: distribution network constraints will be satisfied by activating flexibility bids in the least costly manner.

Step 5: Flexibility assets are compensated for their operation according to a Distribution Network Payment Algorithm (DNPA) that FMO executes, which can be: i) pay–as-bid , ii) a pricing according to the dual variables of DNDA, or iii) an auction that facilitates additional market requirements (e.g. truthful bidding, market power mitigation, etc.)

An identical process is used in (near) real time for the realization of a distribution level flexibility market that reacts to the existing balancing market. It will take as inputs: the imbalances from the WM (as derived from the traditional balancing market), the DNDA and the real time availability of flexibility assets.

2.2.2 Feasibility check of the distribution network and optimization of wholesale market biddings through a proactive distribution network aware flexibility market architecture

In order to mitigate the drawback of the aforementioned architecture (which is the difficulty to manage an infeasible or expensive DAD of the existing WM), FLEXGRID proposes an optimization of biddings within a distribution network in advance (i.e. proactively) by the FMO. In this way, an a-priori feasible dispatch of the assets that reside in the distribution network is ensured.

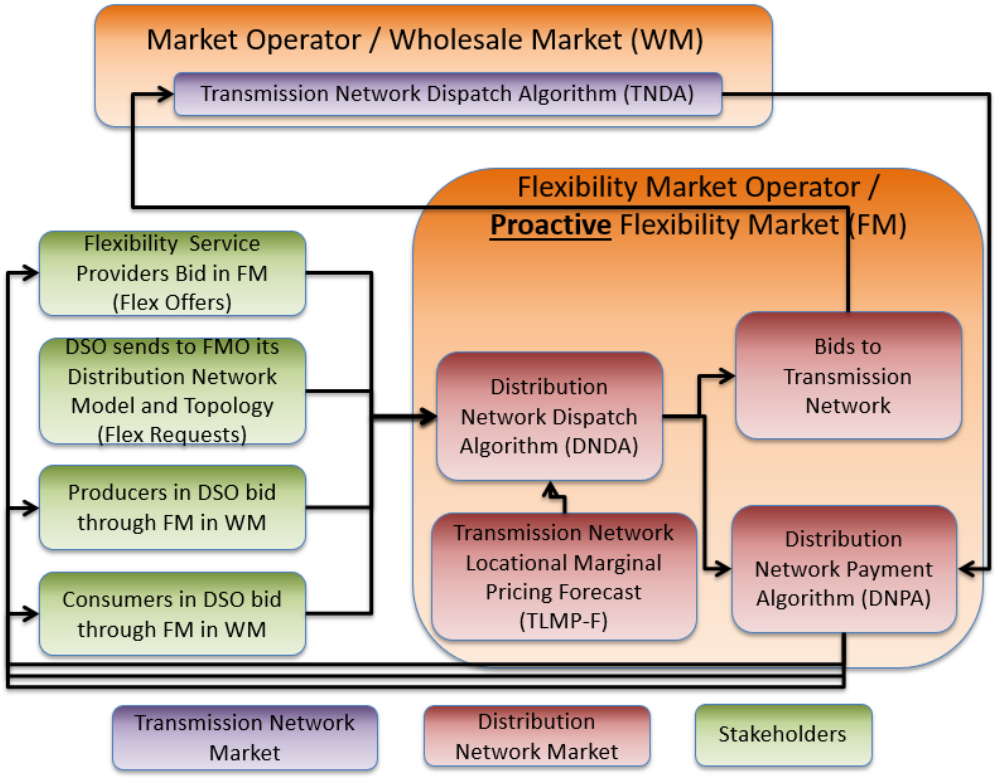

Figure 2: Proactive Distribution Level Flexibility Market (P-DLFM)

On the other hand, in order to allow the FMO to operate proactively, an accurate estimation of the Transmission Network Locational Marginal Prices (TLMPs) in the coupling point between the DSO and TSO is required. An underestimation in TLMPs will possibly result a demand that is lower than the one calculated by the Distribution Network Dispatch - DND (unless the FMO or the DSO pay an additional cost in order to keep the system budget

balanced). An overestimation in TLMPs will possibly lead to a respective over-calculation of generation dispatch (unless the FMO or the DSO pay an additional cost in order to keep the system budget balanced).

Therefore, the steps of the Proactive Distribution Level Flexibility Market (P-DLFM) illustrated in the figure above with respect to the constraints and costs that distribution network introduces are:
Step 1: DSO sends information that suffice to model its distribution network to FMO
Step 2: Flexibility providers (e.g. ESPs/aggregators) connected to the DSO send their flexibility asset bids (quantity, price) to FMO
Step 3: Producers connected to the DSO send their production bids to FMO
Step 4: Retailers connected to the DSO send their consumption bids to FMO
Step 5: FMO (or any other party) generates a forecast for the Transmission Location Marginal Prices - TLMPs for the coupling point, in which DSO is connected.
Step 6: FMO generates the Distribution Network Dispatch (DND) through the execution of an algorithm noted as Distribution Network Dispatch Algorithm (DNDA), which:

- respects distribution network constraints (i.e. a) active/reactive power balance, b) network congestion, c) voltage control issues)
- ensures fundamental economic rules, which means that: i) producers and consumers are paid if and only if a price higher or equal with their bids is feasible and financially sustainable for the grid, and ii) each flexibility asset will be used and get paid if its use is mandatory in the set of assets with the minimum flexibility cost that is needed in order to respect the distribution network constraints.

Step 7: Transmission Level Dispatch (TLD) uses as input bids according to DNDA. DNDA may reduce the quantity in the initial bids, but it does not have the rights to reduce the bidding prices.
Step 8: After TLD's execution, the Distribution Network Payment Algorithm (DNPA) uses as inputs: i) bidding prices of flexibility providers, producers and consumers, ii) Transmission Level Dispatch (TLD) and iii) Distribution Level Dispatch (DLD) in order to derive the compensations for all the stakeholders that bid in Steps 2-4.

2.2.3 A clean-slate approach towards a market based smart grid architecture with optimal social welfare

Novel smart grid architectures, which are able to maximize social welfare lead to: i) energy services with lower cost for consumers, ii) more revenue streams for energy producers and Energy/Flexibility Service Providers (ESPs/FSPs), and iii) lower operation costs for network/system operators (i.e. TSO and DSOs). In order to achieve this in a smart grid with very high, distributed RES and flexibility penetration, in which distribution network faces congestion and voltage issues, an evolved market architecture though an advanced interaction between TSO and DSO is needed. In this perspective, a new market architecture is needed, that evolves the existing architecture of the wholesale market (day ahead and balancing market) and is not compatible with their existing versions.

The figure below presents the market clearing process of a unified energy market, in which stakeholders in both the distribution and the transmission networks are able to trade energy

without causing market imbalances in subsequent markets and network instability problems in other parts of the network[3]. In a nutshell, the core of the proposed market architecture is a unified market clearing based on an iterative process (cf. yellow arrows in the figure below) between the Market Operator – MO (manages the Transmission Network through the operation of the Wholesale Market –WM) and the Flexibility Market Operator – FMO (manages the Distribution Network according to an innovative flexibility market proposed by FLEXGRID).

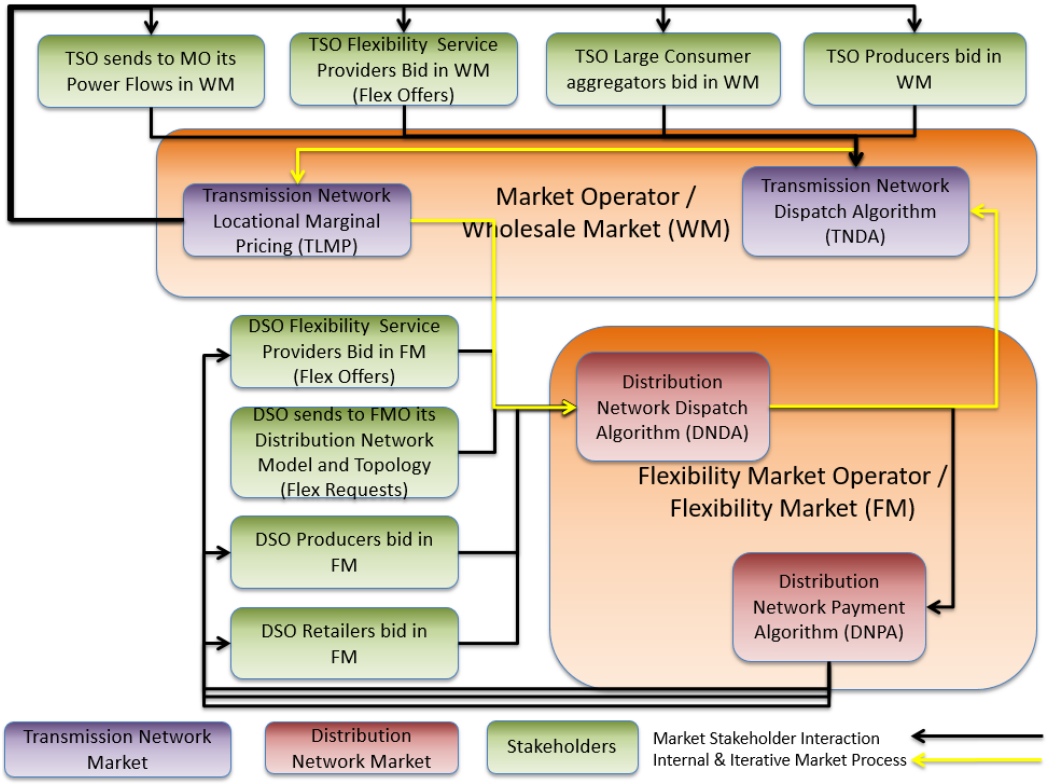

Figure 3: Market based smart grid architecture with optimal social welfare

At each iteration of this process and according to the bids of the transmission network market stakeholders, MO derives a time series (according to the scheduling horizon) of prices (noted as Transmission Network Locational Marginal Prices – TLMPs) for each node in the transmission network. These nodes include the coupling points through which each distribution network exchanges power with the transmission network. FMO of each DSO area takes as input: i) TLMPs that MO derived, and ii) the bids of the distribution level market stakeholders. In a second step, it derives a time series of power flows (Distribution Network Dispatch –DND) in each node of the distribution network and updates the coupling point power flow time series. The termination condition of this iterative process is an identical dispatch in the transmission and in the distribution networks in two consecutive iterations. According to the final dispatch, the pricing in the transmission network is done with the

[3] Please note that the activation of a local FlexAsset in the distribution network may cause market efficiencies and imbalancies in the balancing market operated by the TSO.

existing pricing policy in today's smart grids (TLMPs) and the pricing in the distribution network is done through a payment algorithm that the FMO executes[4].

The necessary steps for the operation of the proposed Market Clearing Algorithm (MCA) process are:

Step 1: DSO sends information that suffice to model its distribution network to FMO

Step 2: Flexibility Service Providers (e.g. ESPs/flexibility aggregators), which are connected to the DSO, send their flexibility asset (e.g. ESS, DSM) bids (FlexOffers) to FMO. Each FlexOffer includes the cost/utility function of the FSP and its device's operating constraints.

Step 3: Producers connected to distribution network (e.g. RES, prosumers) send their bids (production quantity forecast and price) to the FMO.

Step 4: Consumers connected to the distribution network (i.e. demand aggregators) send their bids (demand for different prices) to the FMO.

Step 5: MO generates a forecast of the TLMPs for the first iteration of MCA noted as TLMP.

Step 6: In each iteration k of MCA, the sub-steps below are followed:

Step 6a: Taking the nodal TLMP[k] in the coupling point between with the TSO as input, the FMO of each DSO generates a dispatch (noted as Distribution Network Dispatch – DND[k]) through the execution of Distribution Network Dispatch Algorithm (DNDA). DNDA should factorize the distribution network constraints. Thus, DNDA: i) respects distribution network constraints (i.e. a) active/reactive power balance, b) power lines capacities, c) voltage limits, ii) maximizes DND efficiency, which means that:

- each generation asset in the distribution network will sell its production if the sum of its bid and the possible unitary flexibility cost that is needed to accommodate the distribution network flows that it introduces, is less or equal than the TLMP[k] of the coupling point.
- each consumption asset in the distribution network will buy energy if its price bid is higher than the sum of the TLMP in the coupling point and the possible unitary flexibility cost that is needed for the distribution network flows that it introduces.
- each flexibility asset will be used and get paid if its use is mandatory in order to reach a dispatch able to respect the distribution network constraints with the minimum possible flexibility cost.
- The Power Traded (noted as PT[k]) with the transmission network through the coupling point is calculated.

Step 6b: DLFM stakeholders are compensated for their operation according to Distribution Network Payment Algorithm (DNPA). Various DNPAs with several requirements may take place (e.g. budget balanced, proof to market power abuse, privacy aware, etc.). Major DNPA categories that FLEXGRID will develop are: i) pay as bid, ii) pricing based on the dual variables of the DNDA, which are derived from the execution of an AC –OPF model and algorithm, and iii) pricing algorithms that exploit auction theory.

Step 6c: The transmission network stakeholders (i.e. generators, demand aggregators, etc.) decide their dispatch based on the corresponding nodal TLMPs[k]. The TSO calculates its power flows based on the nodal TLMPs[k], which along with the transmission network stakeholders and the FMOs' decisions formulate the Transmission Network Dispatch (TND).

[4] This means that there may be different prices depending on what grid level a FlexAsset is connected and on which timeslot the FlexService is provided.

Step 6d: If the DND and the TND remain the same between two consecutive iterations of MCA, then it terminates. Otherwise, TLMP[k+1] are calculated by a TLMP Update Algorithm (TLMP-UA) (which uses as input all the aforementioned dispatches and TLMP[k]). FLEXGRID develops various TLMP-UAs mainly based on the Duality Theory [5] and Decomposition Techniques[6].

Step 7: The last calculation of TLMPs and the last calculation of DNPA determine the payments of participants in the transmission and in the distribution network respectively. The last TND and DND determine the dispatch in the two aforementioned networks.

Conclusively, the proposed MCA facilitates a day-ahead and energy balancing smart grid management. In the latter case, MCA will operate in a differential fashion in which (instead of production, consumption and flexibility that constitutes transport and distribution networks' operation feasible), there will be imbalances (Flex Requests) and Flex Offers that have to match the near-real-time imbalances.

Note: This section has set the research scope based on which the S/W architecture design work will be presented in the following sections. FLEXGRID's plan is to evolve and evaluate with real data all the aforementioned architectures and algorithms, such as: TLMP Update Algorithm, Distribution Network Dispatch Algorithm and Distribution Network Payment Algorithm. In this way, FLEXGRID will not only offer to future smart grids the best version of each of the aforementioned architectures, but it will also quantify the strengths and the weaknesses of each one of them. Moreover, FLEXGRID plans to compare these architectures in terms of their potential with respect to the features and the services that they are able offer to the ESPs/FSPs in order for the latter to optimize and constitute their investments financially sustainable.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, USA, 2004.

[6] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, "Decomposition Techniques in Mathematical Programming. Engineering and Science Applications". Heidelberg, Germany: Springer-Verlag, 2006.

3. FLEXGRID S/W architecture definition

A **Reference Architecture (RA)** can be defined as a work product used to describe a concrete (standard) architecture in a more abstract concept. As far as **Smart Grids (SG)** are concerned, the design of an RA needs to follow some particular standards (requirements) in order to be able to produce flexible, globally accepted models. While capable of describing the current state of a SG, a RA needs to take into account its upgrading prospects, in order to be functional in the years to come. Due to the different nature of the involved stakeholders, a coherent and flexible framework must be provided, where a proper categorization of all interested parts is feasible.

Continuous technological advancement in the micro-grid (MG) and distribution networks structure demands a more sophisticated architectural approach. Following the universal dominance of smart-grids (SG), the integration of ICT and market domains requires a multi-dimensional architectural structure so as to allow the participation of all stakeholders involved.

3.1 SGAM concept and objectives

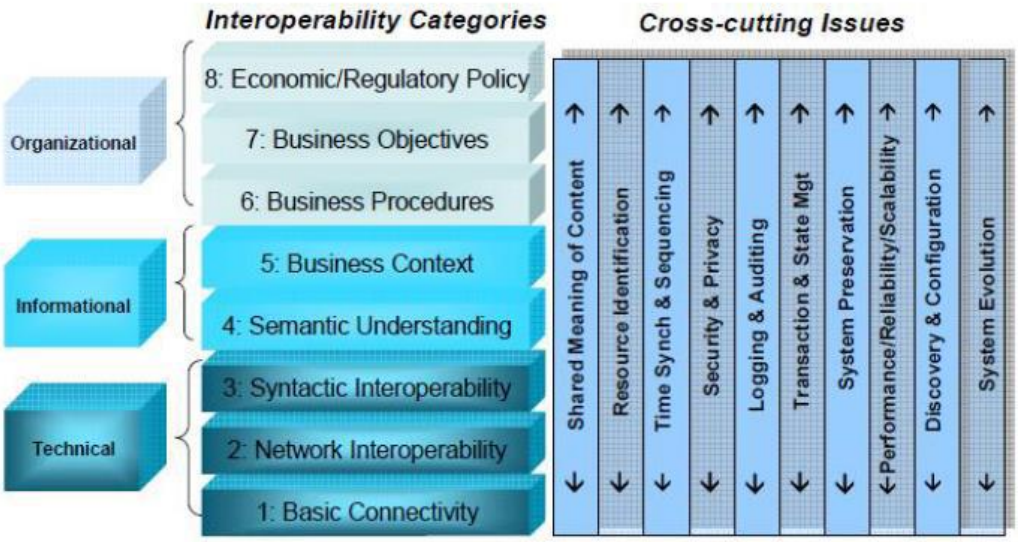

Figure 4 - Interoperability Categories and Cross Cutting Issues [7]

Based on CEN-CENELEC-ETSI Smart Grid Coordination Group report [8], the **Smart Grids Architecture Model (SGAM) framework** can be described as the architectural structure of a practical methodology, where each particular **Use Case Scenario** (**UCS**) can be modelled and analysed from different aspects. While modelling a UCS, the most important factor is the coherency of the whole process, as well as the production of an analytic and detailed object, where the role of each stakeholder is clearly defined. As far as the general presentation of a

[7] https://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf
[8] ftp://ftp.cencenelec.eu/EN/EuropeanStandardization/Fields/EnergySustainability/SmartGrid/CGSEG_Sec_0042.pdf

UCS is concerned, three main categories interoperate between each other, while various Cross-cutting issues (referring to relationships between the categories) need to be taken into account.

In the SGAM framework, these interoperability categories are aggregated into five different levels, the so called **SGAM layers,** as shown in the next figure (Interoperability Categories and layers). As can be seen, each layer refers to a different aspect of every UC, starting from the **Business layer** (referring to the business usage of the smart grid information exchanged, the involved market actors, business objectives, constraints, etc.), moving step by step to the **Component layer** (i.e. physical layer, including all entities of smart grid, such as the system equipment, the network infrastructure and the protection devices). Between these two, lie the **Function**, the **Information** and the **Communication layer**. These layers refer to the: i) functions implemented (functionality of UC), ii) the information object and data models exchanged between functions or actors (devices, applications, persons, organizations) and iii) the protocols/mechanisms used for the exchange of information, respectively.

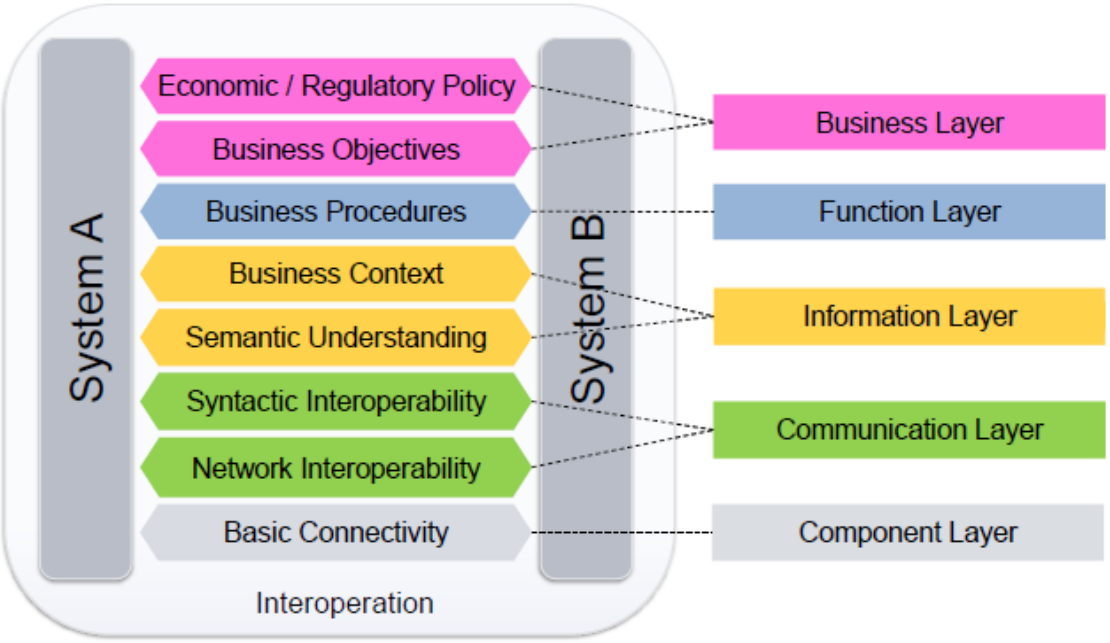

Figure 5 - Interoperability Categories and layers [9]

Subsequently, the interoperability layers need to be merged with another concept, the smart grid plane, to compose the 3D SGAM framework. In the smart grid plane, an important discrimination is made between the electrical processes (domains) and the information management viewpoints (zones) involved in every UCS. The five SGAM domains contain the **Bulk Generation domain** (massive generation of electricity), the **Transmission domain** (infrastructure and organization for the transportation of energy), the **Distribution domain**, the **Distributed Electrical Resources domain** (DER connected to the public distribution grid

[9] ftp://ftp.cencenelec.eu/EN/EuropeanStandardization/Fields/EnergySustainability/SmartGrid/CGSEG_Sec_0042.pdf

ranging from 3 kW-10.000 kW) and the **Customer Premises domain** (prosumers of electricity).

Moving on to the six SGAM zones, these are distinguished as follows. The **Process zone** (refers to the transformation of energy and the equipment involved) is followed by the **Field zone** (protection, control and monitor equipment) which, in turn, is succeeded by the **Station zone** (areal aggregation of previous level). Next comes the **Operation zone** (control operation systems such as DMS/EMS), followed by the **Enterprise zone** (commercial aspect/e.g. logistics, staff training, etc.) and finally the **Market zone** (commercialization of the produced energy).

3.2 SGAM layers' architecture

The basis for SGAM is a three-dimensional framework consisting of domains, zones and layers at its three axes.
The domains represent the traditional layout of the electrical energy infrastructure:

- *Generation* (generation of electrical energy in bulk quantities).
- *Transmission* (infrastructure and organization that transports electricity over long distances).
- *Distribution* (infrastructure and organization that distributes electricity to customers).
- *DER* (small-scale distributed energy resources directly connected to the public distribution grid).
- *Customer Premises* (end-users and producers of electricity).

On the other hand, the zones depict a typical hierarchical power system management:

- *Process* (physical energy conversion and primary equipment of the power system).
- *Field* (protection, control and monitor equipment).
- *Station* (aggregation level for fields, e.g. for substation automation).
- *Operation* (power system control operation in the respective domain, e.g. DMS/EMS).
- *Enterprise* (commercial and organizational processes, services and infrastructures for enterprises).
- *Market* (market operations possible along the energy conversion chain).

These two axes combine to form the *Component* layer, which represents the physical layer, including all system equipment, network infrastructure and protection devices. On top of the Component layer, four interoperability layers are placed.
With a completed UC analysis and the developed component layer, mapping UCs in SGAM and development of SGAM layers generally goes in the following order:

- *Business* layer (business view on the information exchange related to smart grids).
- *Function* layer (functions and services, including their relationships from an architectural viewpoint).
- *Information* layer (information that is being used and exchanged between functions, services and components).

- *Communication* layer (protocols and mechanisms for the interoperable exchange of information between components in the context of the underlying use case, function or service).

In this deliverable, FLEXGRID will define its S/W architecture according to SGAM model in the context of WP2, choosing a relevant subset of the UCS provided on D2.1. Additionally, FLEXGRID will follow SGAM model in its software integration process (WP6) and in the execution of its pilots and lab experimentations (WP7).

A graphical representation of the SGAM framework can be seen in the figure below:

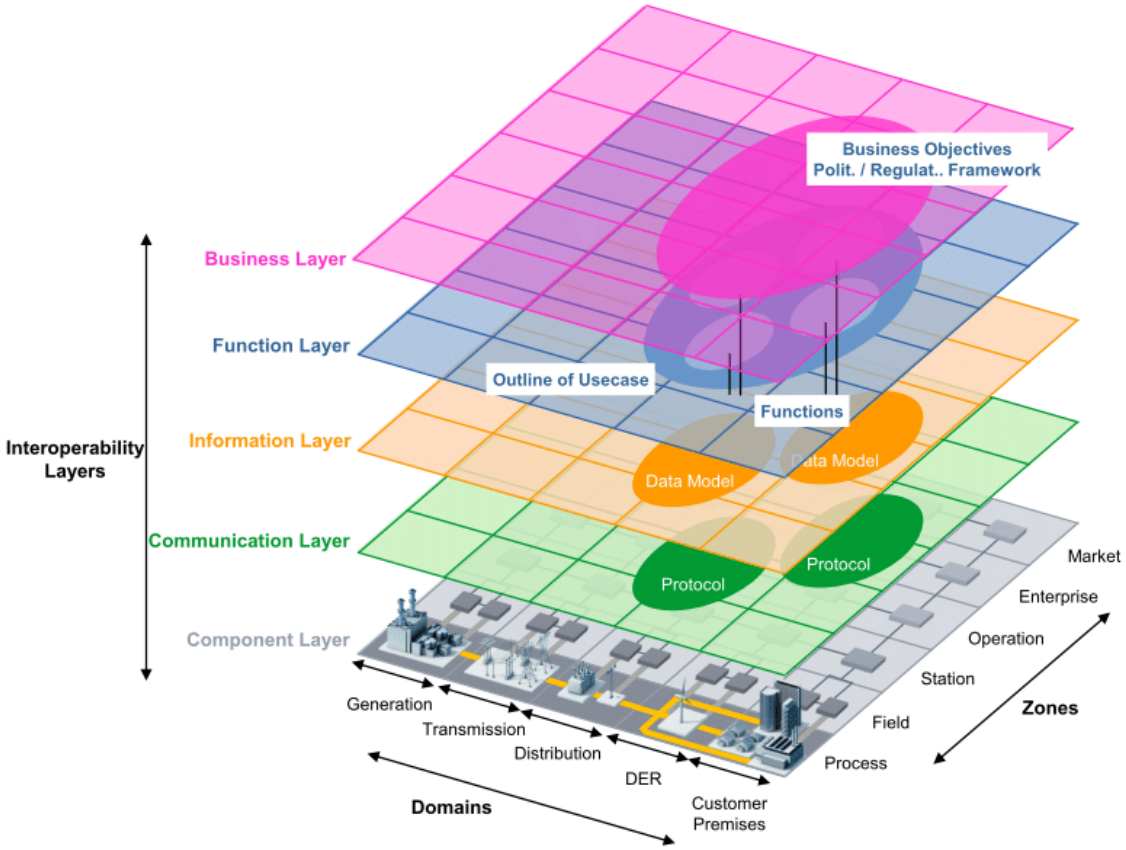

Figure 6 - The SGAM Framework [10]

3.3 SGAM Toolbox for Model-Driven Architecture Specification

In order to proceed with the modelling process of use cases, their incorporation into the SGAM Framework is necessary. This can be actualized by the use of the **SGAM Toolbox** [11], a caption of its basic architecture (V 0.2.0) being presented below (Figure 7 - The SGAM Toolbox Architecture).

[10] ftp://ftp.cencenelec.eu/EN/EuropeanStandardization/Fields/EnergySustainability/SmartGrid/CGSEG_Sec_0042.pdf

[11] https://sgam-toolbox.org/downloads/Introduction-to-SGAM-Toolbox.pdf

As can be seen above, the most important element of the architecture constitutes the SGAM Metamodel, since it provides the necessary outputs such as the layer metamodels, Diagram types, Toolboxes and Design Patterns. The SGAM model templates (used to simplify the use of SGAM Toolbox) are also based on the SGAM Metamodel.

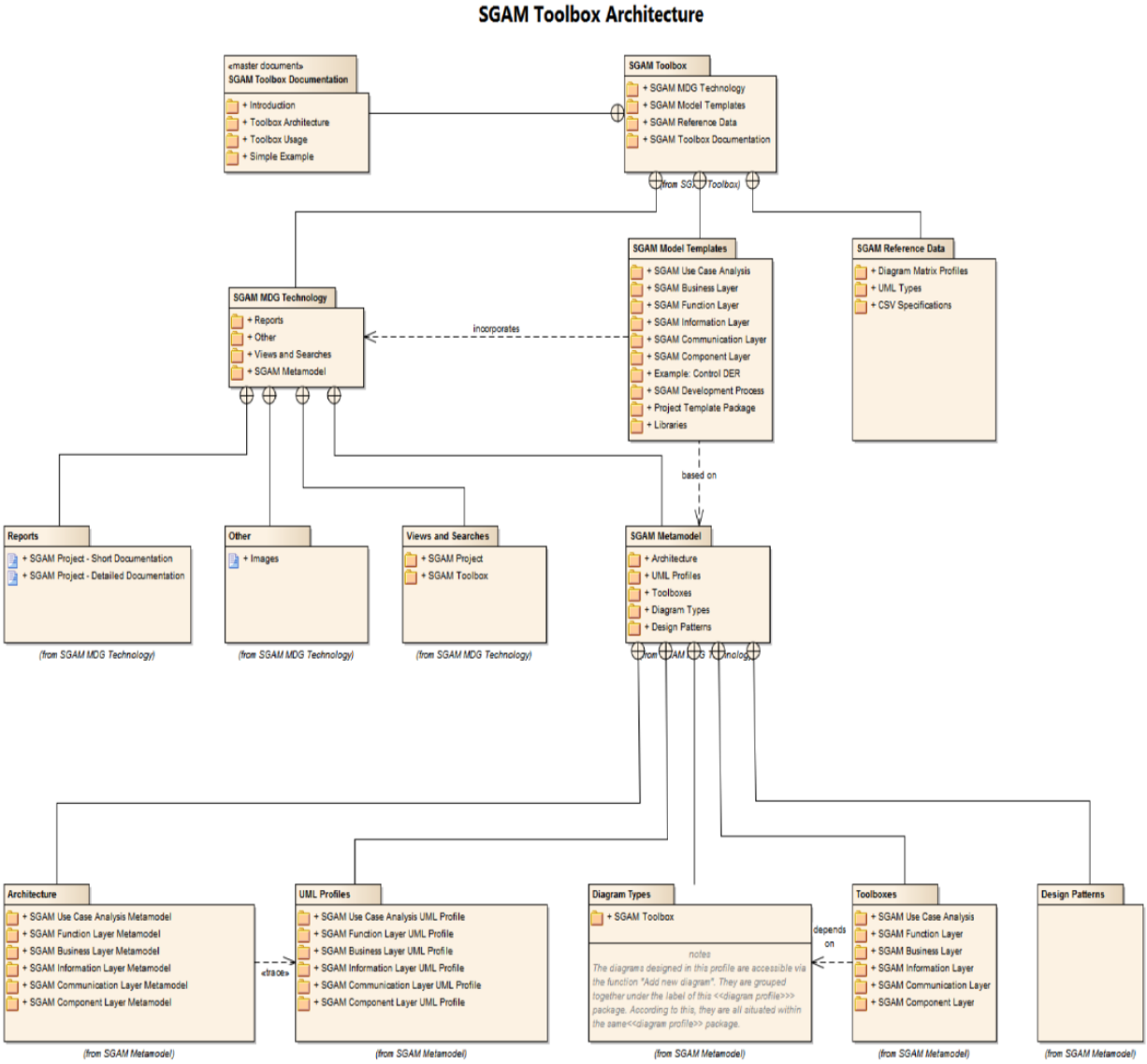

Figure 7 - The SGAM Toolbox Architecture [12]

Regarding the description of the **SGAM Metamodel**, this is extracted from the SGAM MDG (Model Driven Generation) Technologies (i.e. files that allow users to extend Enterprise Architect's modelling capabilities to specific domains and notations). MDG Technologies seamlessly plug into Enterprise Architect (software used to define SGAM architectures) to provide additional toolboxes, UML profiles, patterns, templates and other modelling resources. The last important component of the Toolbox architecture is the Reference Data,

[12] https://sgam-toolbox.org/downloads/Introduction-to-SGAM-Toolbox.pdf

which provides information regarding the Model Import/Export, as well as other important elements. In the figure below, the structure of the SGAM metamodel can be observed.

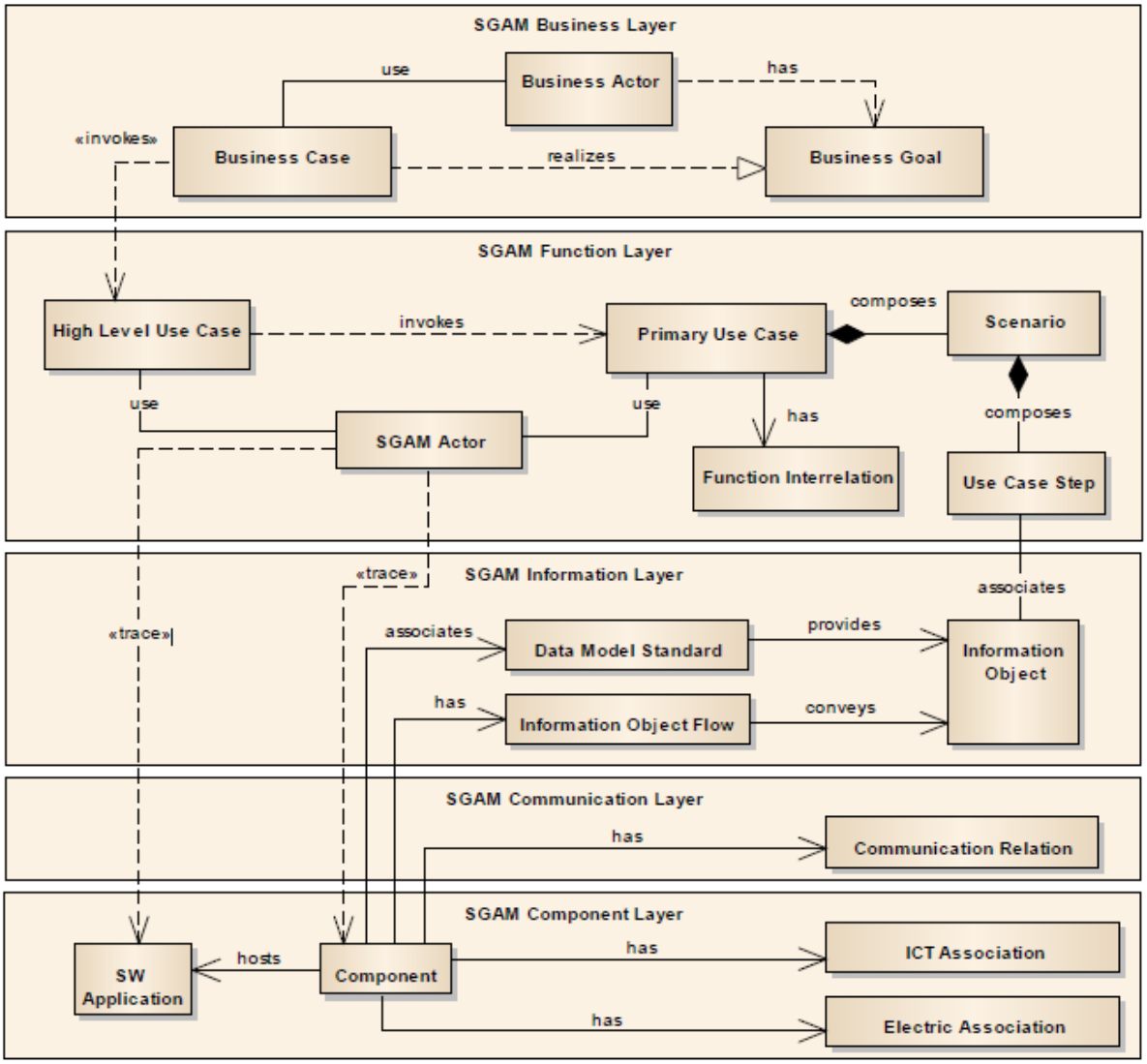

Figure 8 - The SGAM Metamodel [13]

Starting from top, a business actor needs to achieve a goal, while this is only possible through a business case (BC). Moving to the function layer, the High-Level Use Case (HLUC), which provides a general description of an idea/requirement, uses the SGAM actors (devices, applications, persons or organizations), while invoking a Primary Use Case (PUC). PUCs are described by specific scenarios and address the functionality aspect of a business process. While not depicted in the diagram above, a PUC is composed by one or more Secondary Use Cases (SUC). In the information layer, the objects, demanded to compose the scenarios invoked for the description of PUCs (information objects) are defined through the Data Model Standards. These objects contain some type of information exchanged (e.g. a fault report) between actors. Finally, in the SGAM Component layer, the components (electric

[13] https://sgam-toolbox.org/downloads/Introduction-to-SGAM-Toolbox.pdf

device, software, cables, etc.) related to the ICT and electric domains are included. The communication between two or more components is based on various protocols, defined in the Communication Relation.

One last obstacle that needs to be overcome in order to be able to produce UC models, through the SGAM Toolbox, in accordance with the SGAM Framework is the accurate and detailed transformation of UCs into objects that possess distinct SGAM-based characteristics. In other words, the development/UC mapping process needs to be clearly defined.

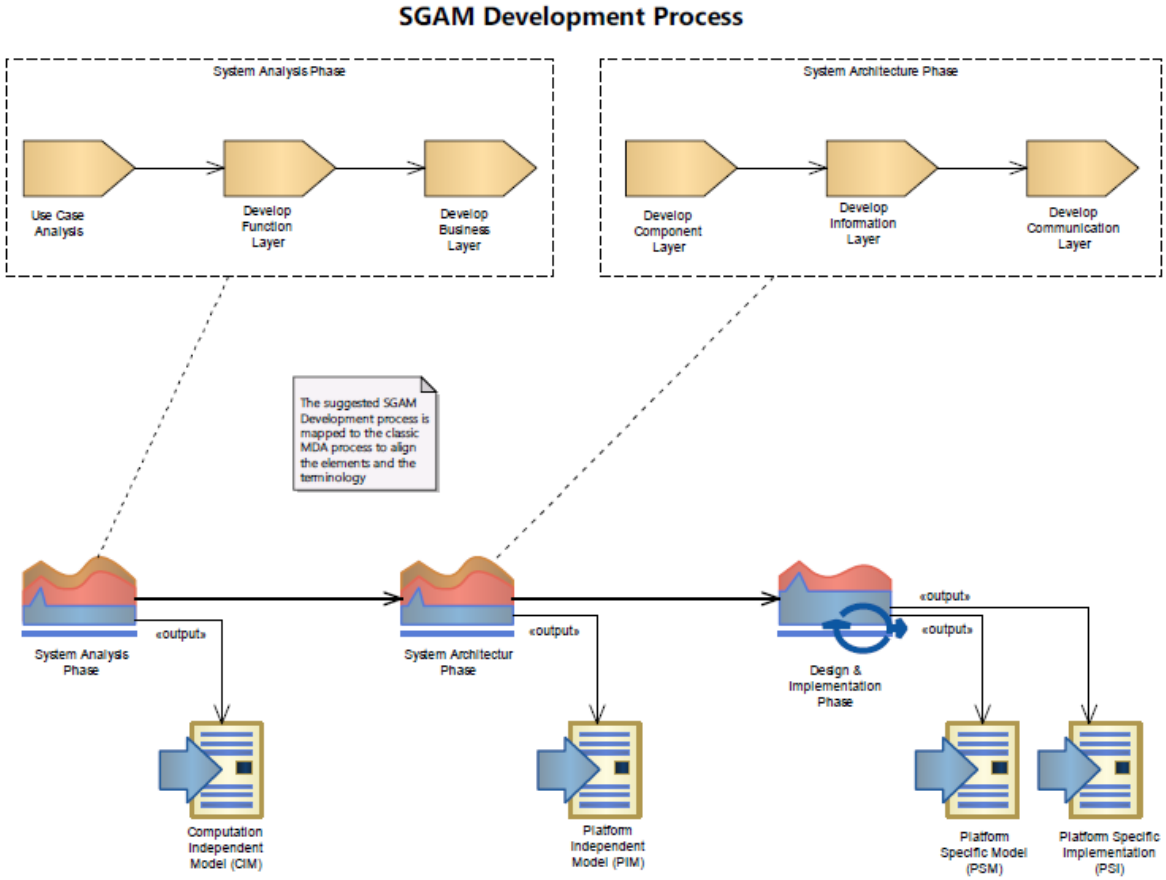

Figure 9 - SGAM Development Process [14]

As shown in the figure above, the process begins with the System Analysis Phase (SAP). During SAP, it needs to be confirmed that the UC description provides the necessary information (objective, UC diagram, actor name and type, precondition and assumptions, steps, information exchanged and requirements), so one can proceed with the development of each SGAM layer. In this phase, the development of the SGAM Function and Business layer should be implemented so as the business actors/goals/cases can be defined. Next comes the System Architecture Phase where the development of the Component, Information and Communication layer is done. Finally, the Design and Development Phase referring to the realization of individual systems can be actualized by any classic system engineering method,

[14] https://sgam-toolbox.org/downloads/Introduction-to-SGAM-Toolbox.pdf

as it does not need to be in compliance with the SGAM Framework. Nonetheless, this methodology is only a recommendation and slight deviations might be observed.

3.4 SGAM concepts in FLEXGRID

Architecture can be described as the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution [ISO/IEC42010[15]].

For simplicity sake, we are going to treat only HLUCs and one layer of their derived UCs, calling them Use Case Scenarios (UCSs).

A more thorough analysis of the description of the related connectivity issues (communication layer), an architecture field associated with the information exchanged between parties/actors (information layer) and the mapping of physical components to logical actors (component layer) are deemed necessary.

As in most modelling tools/frameworks, a compromise must be reached between the detailed description of an entity (each SG sector) and the reduction of complexity of the overall modelling process. In this case, a simple way to achieve this enterprise is the merging of some of the aforementioned architecture concepts into a more general context. Regarding to FLEXGRID, we estimate that its S/W architecture can be modelled just using the three first layers of the SGAM Framework:

- **Component Layer:** Refers to logical actors mapped into physical components.
- **Communication Layer:** This architecture concept is responsible for the elimination of any communication standard gaps.
- **Information Layer**: Refers to the data modelling and interfaces applicable in SGAM model.

In the SGAM Component Layer, the model transformation from the Computational Independent Model (reflecting system and software knowledge from a business perspective) to the Platform Independent Model (model of a system independent of the specific technological platform used to implement it) is represented. Firstly, logical actors have to be mapped into physical components (applications, power system equipment, protection devices, network infrastructures, computers) which must, in turn, be distributed properly among the different SGAM domains and zones. This way, the functional information of the system is turned into an architectural model.

Explaining further the actor mapping process, while inside the SGAM Component Layer, a physical component must be created for each actor involved in the UC. Of course, additional components need to be included, such as ICT networks or individual devices (e.g. a transformer). Subsequently, the relations between the components selected above have to be described and represented into the SGAM component layer.

[15] https://www.iso.org/standard/50508.html

Regarding the Communication Layer, the communication field is closely associated with the operational procedures of a SG. While each actor/component has to implement its own separate functions, the communication between them is essential, so that the individual processes can be coordinated. The information collected must also be sent to a central server so that the optimal operation of the SG may be achieved. The networks included in the overall communication architecture should be identified.

The communication layer aims at the description of the protocols and technology involved, so that the information exchange between different UC actors or components can be achieved. The protocols and mechanisms included in the overall process must be mapped to the appropriate zones and domains. As far as the development of the SGAM Communication Layer is concerned, this is organized in three simple steps. Firstly, the components are mapped in the Communication Layer diagram. Subsequently, communication paths relations are used to connect the different components. Last but not least, the appropriate protocols, as well as the involved technology, are defined, in respect to every communication path.

The representation of important information related to the SG elements is the main reason of existence of the SGAM Information Layer. The functionality of a SG is based upon the exchange of data between the actors/components involved. As seems logical, its architectural structure must support the inclusion of all interrelated entities, as well as the relationships between them and the different ways of interaction. This can be achieved by addressing three basic concepts:

- **Integration technology:** The smooth operation of a SG demands the individual contribution of many different systems and applications. These systems can be described as sources of information production and must be connected, for the optimization of the SG functionality. Consequently, a coupling of these separate systems has to be performed, while the preservation of their individuality (in terms of functionality and performance) is desired. In order for these requirements to be met, the development of new interfaces, which can guarantee the semantic and syntactic interoperability between the different systems and applications involved, is considered. In addition to this, the presence of an integration platform is essential, so that the new interfaces can be implemented upon and are able to communicate between each other. The final scope of the information layer is the provision of a link between different SGAM layers, or fields. This integration shall be performed via APIs.
- **Data Models:** Data models can be characterized as the core of the information layer architecture. In data models, business data are contained and organized in respect to the information included, through which the communication between different SG entities can be performed. As far as the data description language is concerned, a top-down approach is recommended, since it provides a plurality of advantages (such as the avoidance of useless translations or misunderstandings between different stakeholders and the increase of system flexibility). The most prominent standardized data models are the CIM (IEC 61968, 61970 62325) and IEC 61850 (an international standard defining communication protocols for intelligent electronic devices at electrical substations) data model.
- **Interfaces:** Technology independent interfaces (e.g. CIM profiles) are necessary, so that the communication and data model exchange between different SGAM layers,

domains and zones can be possible. While many different standards have been developed, only some of them are recommended, taking into consideration that the semantics and syntax should be stable as long as the system is considered functional. Since the definition of new interfaces between different SGAM layers can prove quite a challenge, the concept of **logical interfaces** was developed, aiming to the simplification of the whole process by providing a systematic way of developing the interfaces' specifications in regard to the logical relations (excluding any physical or technical characteristics). The development of logical interfaces can be summarized into three basic steps. At first, every UCS must be properly analyzed. The mapping of UCS actors to the appropriate domains and zones is performed in this step. Subsequently, the identification of exchangeable information is performed, a process involving the assignment of each piece of information to the associated logical interface (indicated by the dots in the logical interface circle, Figure 10). Finally, all different specifications are merged.

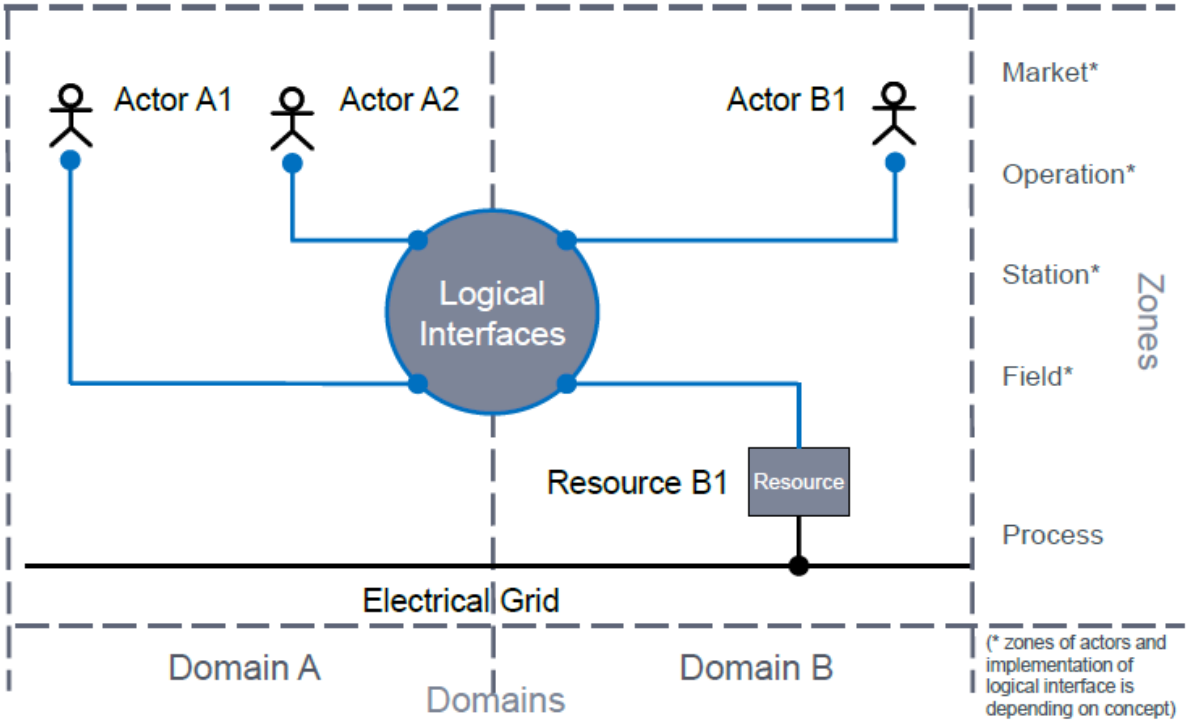

Figure 10 - Concept of logical interfaces in the context of domains and zones

3.5 FLEXGRID approach to SGAM

3.5.1 List of FLEXGRID HLUCs and UCSs

In this section, we address the relevant UCS for the software implementation, that will be described on section 5. The ones selected from D2.1 to be included in the SGAM architecture are listed below:

- ***HLUC 01:*** **FLEXGRID ATP offers advanced market clearing services to the Flexibility Market Operator (interaction between markets' and networks' operation)**

- *HLUC_01_UCS_02:* Market-based local congestion management using FLEXGRID ATP in distribution networks using output from AC-OPF model calculation as dynamic input for ATP.
- *HLUC_01_UCS_03:* Market-based local voltage control using FLEXGRID ATP in distribution network operation.
- *HLUC_01_UCS_04:* FLEXGRID ATP operates as a gateway to redirect local active power flexibility to TSO platforms (interaction with existing TSO balancing markets).
- ***HLUC_02:*** **FLEXGRID ATP offers advanced flexibility supply management services to Energy Service Providers (ESPs)**
- *HLUC_02_UCS_01:* ESP minimizes its OPEX by optimally scheduling the consumption of end users, production of RES and storage assets.
- *HLUC_02_UCS_02:* ESP minimizes CAPEX by making optimal investments (i.e. optimal siting and sizing) on RES and FlexAssets.
- *HLUC_02_UCS_03:* ESP maximizes its profits by co-optimizing its participation in several existing energy markets and distribution level flexibility markets.
- ***HLUC_03:*** **FLEXGRID ATP offers advanced flexibility demand management services to system operators**
- *HLUC_03_UCS_01:* Coordinated voltage/reactive power control either by aggregating flexibility from multiple FlexAssets or through a market-based mechanism.
- *HLUC_03_UCS_02:* TSO-DSO collaboration for coordinated management of aggregated FlexAssets and interaction between networks' and flexibility markets' operation.
- ***HLUC_04:*** **FLEXGRID ATP offers automated flexibility aggregation management services to ESPs/Aggregators (interaction with end users)**
- *HLUC_04_UCS_01:* ESP/aggregator efficiently responds to FlexRequests made by TSO/DSO/BRPs by optimally orchestrating its aggregated flexibility portfolio of end energy prosumers.
- *HLUC_04_UCS_02:* An aggregator operates an ad-hoc B2C flexibility market with its end energy prosumers by employing advanced pricing models and auction-based mechanisms
- *HLUC_04_UCS_04:* ESP exploits FLEXGRID's advanced forecasting services to predict market prices and FlexAssets' state and curves in the future.

Note: The selected UCS will be included in the S/W implementation and integration (WP6) as well as lab experimentations and pilot testing (WP7) reaching thus TRL 5-6 at the end of the project's lifetime. For the residual UCS that were not selected at this stage, advanced research work will be conducted reaching thus TRL 3-4 at the end of the project.

3.5.2 List of FLEXGRID actors

Based on the wholesale markets and the currently planned infrastructure of flexibility markets, three main user profiles for the FLEXGRID S/W platform are identified: Platform operator, flexibility suppliers (or supply side) and flexibility buyers (or buy side). Every user will log into the platform portal and will be redirected to the GUI specifically configured according to the user profile's main requirements.

- *Flexibility Market Operator (FMO) user:* This user shall be able to see all market activity across the various DSO regions. This user has a profile with administrative (admin) role for the FMO itself (able to use all operations). Additionally, a regular role will be assigned for the other users (System Operators, Aggregators) for them to be able to consult the data on their interactions with the Market (selected Read permissions and particular operations), following up and extending existing implementations from NODES Market user description. Moreover, FMO is responsible for registering and accepting new customers and assigning roles for them in the market. The FMO as operator of the platform is in charge of the correct functioning and maintenance of the ATP. In the FLEXGRID API, this includes the correct functioning of the market platform itself, identification andnotification of malfunctioning of the connected toolkits tothe responsible develo ping unit based on monitoring of automated processes. Furthermore, this user incorporates administrative rights to be to manage the validation and settlement process and to perform any action required in the context of client support/customer service.

- *FlexSupply users:* From D2.1, we identify 3 different profiles for aggregator users (retailer, ESP and independent aggregator) from FlexSupply side, which will be granted to access different pages or functionalities in the ATP GUI. All of them will have access for consulting the Market with a regular profile as well.
    - *Retailer:* It should be able to run AFAT retail pricing algorithms, and visualize the RES production and load consumption of all the FlexOwners that belong to its portfolio.
    - *ESP:* These users should be able to register the FlexAssets in the respective DSO zone and to run the algorithm-based FST to evaluate the minimization potential of the company´s CAPEX and OPEX considering the market activity. For monitoring and validation and settlement purposes they are also required to upload the asset or portfolio specific production/consumption forecasts or baselines, respectively.
    - *Aggregator:* Aggregators´ rights should be similar to the ESP. An independent aggregator is not responsible for the balance management of the zone, where its assets are located.The user requirements are thus similar to those of an Aggregator, however limited to the placement of offers from assets/ end customers that have the same BRP. This user should be able to login the Automated Flexibility Aggregation Toolkit (AFAT) and run an efficient automated flexibility aggregation algorithm.

- *FlexDemand users (DSO, TSO):* From the FlexDemand side, two users are identified.
    - *DSO:* Upon registration, this user has to be able to determine the spatial extension of his grid zone by submission of geographical coordinates. Furthermore, this user should be able to register congested regions (grid locations, nodes) in the market platform, to accept/confirm FlexAssets as belonging to its grid and under a particular grid location, to view baselines

submitted/updated by the respective FlexSuppliers/ESPs, and to access several functions[16] in the market (Buy orders). Ultimately, the DSO should be able to run the FMCT to identify/determine potential congestion.

- *TSO:* The TSO should be able to view/manage FlexRequests aggregated from the connected DSOs and get access to related prognoses/baselines entered in the DSO locations. Finally, TSO's balancing market will interact with the Distribution Level Flexibility Market (DLFM) (i.e. FlexOffers will be redirected from ATP to TSO balancing market, while TSO's balancing market results will be communicated to ATP).

3.6 List of FLEXGRID modules

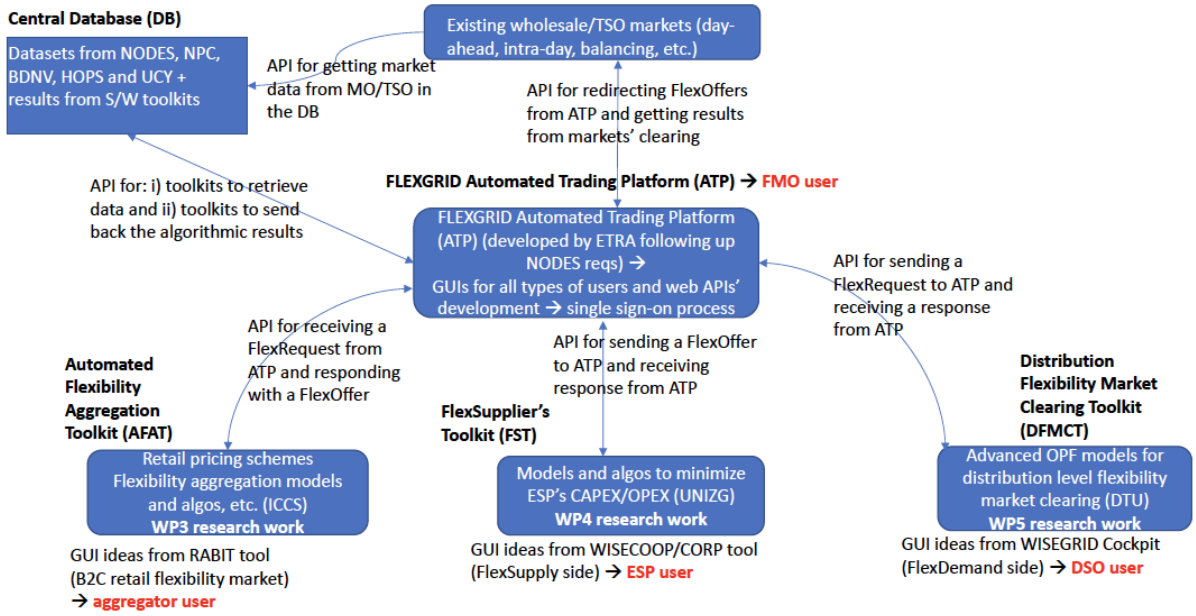

Figure 11 - Draft FLEXGRID S/W architecture design (Month 4)

As stated in section 6.2 of D2.1[17], the FLEXGRID platform will be composed by several S/W modules (Figure 11):

- **Automated Trading Platform (ATP):** It is actually the "frontend" system, in which the various types of users mentioned above (section 3.5.2) may login and navigate through various Graphical User Interfaces (GUIs). There are several web APIs for the interaction between the core ATP and the various subsystems (i.e. AFAT, FST and FMCT). This module will be developed by ETRA.
- **Automated Flexibility Aggregation Toolkit (AFAT):** It is the S/W tool that integrates the various WP3 research algorithms and will be implemented by ICCS and UCY. AFAT will receive a FlexRequest from ATP, will then run a retail flexibility pricing or flexibility aggregation algorithm and will then respond with a FlexOffer to the ATP. The retailer and aggregator user will use this toolkit. The retailer user will also be able to run

[16] i.e. to match FlexOffer and to place an initiator order.

[17] D2.1: "FLEXGRID use case scenarios, requirements' analysis and correlation with innovative models", available online at https://flexgrid-project.eu/deliverables.html

system-level simulations to decide on its optimal retail pricing scheme (i.e. automatically derive the most suitable FlexContract with the end energy prosumer).

- **FlexSupplier's Toolkit (FST):** It is the S/W tool that integrates the various WP4 research algorithms and will be implemented by UNIZG-FER. FST will run a specific algorithm to minimize ESP's OPEX and will then send an optimal FlexOffer to the ATP. Based on the market clearing results and the response sent by the ATP, the FST will be able to recalculate a better FlexOffer or re-schedule its FlexAssets in the optimal way (e.g. to participate in a subsequent intra-day and/or near-real-time balancing market). The ESP user will use this toolkit and two main algorithms are expected to be integrated, namely: i) optimal scheduling algorithm to minimize ESP's OPEX (either participating in just one or several markets at the same time stacking thus its revenues), and ii) optimal investment algorithm to minimize ESP's CAPEX.
- **Flexibility Market Clearing Toolkit (FMCT):** This is the S/W tool that integrates the various WP5 research algorithms and will be implemented by DTU. FMCT will run advanced market clearing algorithms (e.g. AC-OPF for distribution networks). It will be used by the DSO user in order to calculate the nodal prices and thus send a respective FlexRequest to the ATP. Moreover, it will be used by the FMO user in order to automatically match FlexSupply and FlexDemand at the distribution network level.

All the algorithmic results will be stored in the **central database (DB)** together with all real-life/realistic datasets from NODES, NPC, BDNV, HOPS and UCY business. These datasets will be used for the validation of the various research algorithms.

3.7 Relation with other projects

Some applications developed for other projects will be used as a baseline for the ATP GUIs, such as "WiseGRID COOP" [18] [19] and "WiseGRID Cockpit" [20] [21]. Other apps will be used as guideline for the user interface, like SOCIALENERGY "RABIT"[22] and "NODES Market"[23], taking ideas from their user interface for using an adequate design to display relevant information. A brief summary on these inclusions can be found in section 4.3.1, ATP internal architecture. It should be noted that the main focus of FLEXGRID S/W implementations will be to integrate intelligence from the three research toolkits (i.e. AFAT, FST and FMCT) inside the core FLEXGRID ATP.

[18] https://cdn.nimbu.io/s/76bdjzc/channelentries/u0pwift/files/D7.1_WiseGRID_WiseCOOP%20and%20WiseCORP%20Apps%20Design.pdf?hlwubsm

[19] https://cdn.nimbu.io/s/76bdjzc/channelentries/odiihmn/files/D7.2%20WiseCOOP%20and%20WiseCORP%20Apps%20implementation%20and%20lab-testing.pdf?c63g8h4

[20] https://cdn.nimbu.io/s/76bdjzc/channelentries/7d2h1pz/files/D13.1_WiseGRID%20Cockpit%20Design.pdf?gixxdlp

[21] https://cdn.nimbu.io/s/76bdjzc/channelentries/3bnbuhf/files/D13.2%20WiseGRID%20Cockpit%20implementation%20and%20lab-testing.pdf?n0i11vx

[22] P. Makris**,** D. J. Vergados, I. Mamounakis, G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, E. Varvarigos, "A Novel Research Algorithms and Business Intelligence Tool for Progressive Utility's Portfolio Management in Retail Electricity Markets", IEEE ISGT Europe 2019, Bucharest, Romania, 29 September-2 October 2019.

[23] https://nodesmarket.com/

4. FLEXGRID S/W product architecture

4.1 Introduction

Software architecture refers to the **design and creation of fundamental structures** composing a software system and their integration together, making fundamental structural choices that are costly to change once implemented. Each structure comprises software elements, relations among them and properties of both elements and relations. Software architecture choices include specific structural options from relevant possibilities in the design of the software platform.

For the integration to work properly, a suitable design is needed, and there exist several mature architectural styles that can comply with FLEXGRID requirements and goals for adequate interconnection. Following **best practices** on well-known S/W technologies, proven standards, design patterns and data formats will provide FLEXGRID with consistent data models, robust documentation, improved security and fast integration. This also allows more agile developing, evolving and replacing services that the software platform can offer.

Finally, designing a software architecture is usually too complex to address all details on its structures simultaneously, that's why many architecture design models require an iterative or incremental development process. This is a **cyclic process** of prototyping, testing, analyzing and refining the software platform, where the results of these activities enable feedback to evaluate changes and refinements on the design. This process is intended to improve the quality and functionality of the derived product. In WP6, we will follow two main cyclic processes. The first cycle will start in Month 13 and will end in Month 18 via the release of the first version of FLEXGRID S/W prototype. The second cycle will start in Month 19 and will end in Month 33 via the release of the final version of FLEXGRID S/W prototype. It should be noted that during the second cycle, several mathematical models and algorithms will be gradually integrated in the FLEXGRID S/W platform following up the progress from research WPs 3-5.

4.2 FLEXGRID modules integration via APIs

"An application programming interface (API) is an interface or communication protocol between different parts of a computer program intended to simplify the implementation and maintenance of software.
An API may be for a web-based system, operating system, database system, computer hardware, or software library." [24]

In FLEXGRID, we are going to implement **Web-based APIs** for the data interoperability between the different main modules of the project. They will allow communication and data exchange over the whole software platform. Using consistent data models, their objects or

[24] https://en.wikipedia.org/wiki/Application_programming_interface

resources (**subjects**) will be determined and the available operations (**verbs**) specified, as a first step to design these Web-based APIs.

These APIs should allow semantic operations over the accessible resources; in software programming, create, read, update and delete (**CRUD**) are the four basic functions of persistent storage. CRUD is a common data lifecycle pattern that will be applied on top of the API model, and their operations usually corresponds with network protocol methods underlying the API. CRUD is a cycle that can be mapped to many API models by design.

4.2.1 RESTful Web Services (REST APIs)

Since it was first proposed in 2000 by Roy Fielding [25], REST architectural approach for designing web services quickly became a de facto standard during the next years, and even with the fast evolution of web technologies in last decade, it is still a relevant API style.

REST stands for "REpresentational State Transfer", and it is a software architectural style that defines a set of constraints to be used for creating web services, the so-called RESTful Web Services or simply REST APIs. They provide interoperability between distributed hypermedia systems, allowing loosely coupled applications mainly over HTTP protocol although REST is network protocol independent. REST defines **six architectural constraints**, which make any web service a true RESTful API, briefly exposed:

1) *Client–server:* Portability across multiple platforms is improved by separating storage from user interface issues, as well as scalability. User interaction is client's domain and data access, workload management and security are server's domain. This loose coupling of the client and server enables each one to be developed, evolved and enhanced independently. The interaction model follows Request-Response scheme.

2) *Stateless:* Each client-server operation must include all information necessary to be processed, and any management or session state take place entirely on the client side, not the server side. One may not be confused between application state and resource state, REST statelessness means being free on application state (i.e. which is server-side data about previous interaction details and current context).

3) *Cacheable:* This constraint requires that all requestable resources have to be labelled as cacheable or non-cacheable; these resources should allow caching unless explicitly indicated that caching is not possible. When a response is cacheable, the client cache may reuse that response for later equivalent requests in order to reduce latency, improving performance and scalability. To be cacheable is a requirement for idempotent operations.

4) *Uniform interface:* Each resource should be uniquely identifiable using a single URL, and they can be manipulated through methods (such as DELETE, PUT and GET with HTTP communication) from the underlying network protocol. This way, the visibility of

[25] https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

interactions is improved, uniform interface simplifies and decouples the architecture. Pure REST implementations use HATEOAS principle (Hypermedia as the Engine of Application State), which is explained in more detail below.

5) *Layered system:* Having a layered architecture allows multiple hierarchical layers of servers that can constrain behaviour to control interactions (like one layer cannot be aware of others beyond the immediate one/s it is interacting with).

6) *Code on demand (optional):* Servers can send executable code to the clients in the form of applets or scripts. This extends and simplifies client functionality by reducing the number of features required to be pre-implemented.

The key information abstraction in REST is a **resource** and their APIs are designed around resources, which are any kind of object, data or service that can be accessed by the client. Resource identifiers are used to identify the particular resource involved in an interaction between components, usually through URIs (Universal Resource Identifiers).

The resource state at any particular timestamp is known as **resource representation**. A representation consists of data and metadata describing the data and hypermedia links, which can help the clients in transition to the next desired state. Resource representations shall be self-descriptive: the client does not need to know if a resource is one subject or another. It should act on the basis of media type associated with the resource.

The data format of a representation is known as a **media type**. The media type identifies a specification that defines how a representation is to be processed. Every media type defines a default processing model: for example, HTML defines a rendering process for hypertext and the browser behaviour around each element.

The format of a request or response body data is the media type (the format of client-server interaction). Web service operations can accept and return data in different formats, the most common being JSON, XML and images. Their usage can be specified, for instance in HTTP, with application/json, application/xml, text/html, text/plain, image/png, etc.

Media type defines data model, processing model, hypermedia controls (i.e. annotated links, input forms) and support to include additional application specific information described by link relations, element names, etc. In practice, developers end up creating lots of media types (like 'application/vnd[…]+xml', or 'application/vnd[…]+json'), usually one custom media type associated with one resource.

A pure RESTful API looks like **hypertext**. Every addressable unit of information carries an address, either explicitly (link and id attributes) or implicitly (derived from the media type definition and representation structure).

"Hypertext (or hypermedia) means the simultaneous presentation of information and controls such that the information becomes the affordance through which the user (or automaton)

obtains choices and selects actions. [...] Machines can follow links when they understand the data format and relationship types." [26]

For interacting with the API resources, **resource methods** are used to perform the desired action or transition. The resources are acted upon by using a set of simple, well-defined operations. Ideally, on pure REST APIs, everything that is needed to change the resource state shall be part of API response for that resource, including methods and in what state they will leave the resource representation. The clients and servers exchange representations of resources by using a standardized interface and network protocol.

Regarding the **uniform interface constraint**, it is fundamental for REST systems design and we have to consider four interface properties:

1) *Resource identification in requests:* Individual resources are identified in requests, for example using URIs. The resources themselves are conceptually separate from the representations that are returned to the client. For example, the server could send data from its database in a different format (e.g. JSON, XML) that the one used in the internal server's representation.
2) *Manipulation of resources through representations:* When a client holds a representation of a resource, including any metadata attached, it has enough information to modify or delete the resource.
3) *Self-descriptive messages:* Each message includes enough information to describe how to process the message. For example, which functionality to invoke can be specified by a media type.
4) *Hypermedia as the engine of application state (HATEOAS):* Any client, having accessed an initial URI on the REST API, should then be able to use server-provided links dynamically to discover all the available actions and resources it could navigate dynamically. When accessed, the server responds with text that includes hyperlinks to other actions that are currently available. The client does not need to be hard-coded with information regarding the structure or dynamics of the application.

The single most important reason for **HATEOAS** is loose coupling: If a consumer of a REST service needs to hardcode all the resource URLs, then it is tightly coupled with your service implementation. Instead, if you return the URLs that it could use for the actions, then it is loosely coupled. There is no tight dependency on the URI structure, as it is specified and used from the response.

With HATEOAS, responses to REST requests return not just the data, but also actions that can be performed with the resource (hypermedia information and controls). This helps in making the applications loosely coupled and able to dynamic navigation: By including unique URLs within a response package, hypermedia APIs can tell clients what capabilities are possible and in what scenarios.

[26] Roy Fielding: https://restfulapi.net/

In practice, most of the so-called REST implementations not fully complies with HATEOAS property. For this consideration, in 2008, Leonard Richardson proposed the following **maturity model** (RMM levels) for web APIs:

- *Level 0:* Single URI and a single verb. Assuming HTTP protocol being used, it defines one URI and all operations are POST requests to this URI, with the details of the query included in the body section.
- *Level 1:* Multiple URI-based resources and single verb. It creates separate URIs for individual resources, where operations are defined in the message itself.
- *Level 2:* Multiple URI-based resources and verbs. Assuming HTTP protocol, it uses different HTTP methods to define operations on resources and status codes are used appropriately.
- *Level 3:* Use of hypermedia (HATEOAS), where server responses include links that clients can use for dynamic navigation.

Level 3 corresponds to a truly pure RESTful API according to Fielding's definition. Many published REST APIs fall somewhere around level 2; following this Richardson's maturity model FLEXGRID REST APIs will fall on level 2 too, as HATEOAS is not really necessary due to FLEXGRID modules static navigation and should only apply whenever relevant (e.g. dynamic navigation).

Besides all these explained formal definitions, there are some technical issues and practices that should be observed in the scope of building FLEXGRID APIs:

First, the clients and servers have to exchange representations of resources by using a standardized interface and **network protocol**. REST is independent of any underlying network protocol; however, most common REST implementations are delivered over HTTP (HyperText Transfer Protocol), that is a request-response based protocol. Anyway, REST and HTTP are not the same.

This protocol is adequate for FLEXGRID REST APIs and they shall make use it, through its 1.1 version (HTTP/1.1). HTTP basically provides URLs that allow us to define resource paths. There are verbs (GET, POST, PUT, PATCH methods; related to CRUD) to enable acting on such resources, there are response codes, and there are headers that allow us to define client/server relationships. Besides, HTTP enables content negotiation, allowing APIs to negotiate what data formats (JSON, XML, etc), functionalities (compression) or languages the API can support and send to a client. HTTP also enables caching, ETags, conditional requests, and concurrency control. Some implementations are tunneling protocol semantics inside of HTTP, when they really could just use HTTP itself.

The HTTP protocol defines a number of methods that assign semantic meaning to a request; their verbs comprise a major portion of the "uniform interface" constraint and provide us the action counterpart to the subject-based resources. The most common used HTTP verbs (properly called methods) are POST, GET, PUT, PATCH, and DELETE. There are some other methods less frequently used, like OPTIONS and HEAD. Related to CRUD operations, HTTP methods or verbs are commonly used for:

- *Create:* POST creates a new resource at the specified URI. The body of the request message provides the details of the new resource. Note that POST can also be used to trigger operations that don't actually create resources.
- *Read:* GET retrieves a representation of the resource at the specified URI. The body of the response message contains the details of the requested resource.
- *Update/Replace:* PUT either creates or replaces the resource at the specified URI. The body of the request message specifies the resource to be created or updated.
- *Update/Modify:* PATCH performs a partial update of a resource. The request body specifies the set of changes to apply to the resource.
- *Delete:* DELETE removes the resource at the specified URI.

The API servers shall always return proper HTTP status codes (e.g. 200 OK, 201 Created, etc) for each call, especially considering 4xx series status codes for client issues and, if needed due to special network requirements (e.g. load balancers, proxies, gateways), extending to 5xx series status codes for server issues. Use status codes consistently in a way that its meaning is expected; lists and descriptions of HTTP status codes (standardized and unofficial) can be found on the Internet [27].

Second, the data format of API interactions should be a proven standard for lightweight **data-interchange format** (the media type itself), like JSON or XML. The trend is to use JSON, and very convenient to deal with for a diverse range of applications. JSON is an open-standard file format or data interchange format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value).

JSON (JavaScript Object Notation) grew out of a need for stateless, real-time server-to-browser communication protocol without using browser plugins (such as Flash or Java applets), and nowadays it is a very common data format standard. It is intended as a data serialization format, human readable, with low overhead and language-independent (it was derived from JavaScript, but many modern programming languages include code libraries to generate and parse JSON-format data).

The JSON schema to be used shall be defined based on the final FLEXGRID Data Model, which is part of WP6 work. There are JSON's related formats for encoding a variety of data structures (like encryption, geographical information, nucleic acid sequences, etc) that may help defining the schema. All API calls and responses shall use JSON as the interchanged data format, specifying in HTTP headers that we deal with this format ("Accept: application/json" for client requests, and "Content-Type: application/json" for server responses; if this header is not set on requests, server's responses shall throw a 415 Unsupported Media Type HTTP status code).

Furthermore, whenever an API call result in an error status code, if relevant the body of the server's response should include additional information about why the error happened, not

[27] https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

just what failed, oriented to developer work and not showing sensitive information (like usernames).

This custom error messages and codes will be delivered in consumable JSON representation, and specified in the API design. As a reference, RFC 7807 is an example "Error Response Format" that can be used [28].

The third issue is end-to-end compression; REST APIs can return resource representations in several formats (JSON, HTML, plain text, etc) and all of such forms can be compressed to a less number of bytes to allow saving bandwidth while being transmitted over the network. This is very convenient since bandwidth resource is far more expensive than CPU time, that is used for compressing and decompressing. This can save around 60% or even 80% of bandwidth, allowing faster communications as the latency is much lower. Actually, some API providers don't allow uncompressed communication with their API servers and FLEXGRID should not allow uncompressed communication as well.

Compression comes with minimal cost and without additional complexity, it is a reversable transformation of the representation of data in transit, and must be undone before the transmission endpoints can use this data representation. Compression can be performed by end-to-end or by hop-to-hop (on which servers use Transfer-Encoding header), that is not the better option for FLEXGRID operations. HTTP protocol allows several algorithms to be used for end-to-end compression, and some of the official ones maintained by IANA are:

- *bs:* A format using the Brotli algorithm, specifically designed for HTTP content encoding, defined in RFC 7932.
- *gzip:* A format using the Lempel-Ziv coding (LZ77), with a 32-bit CRC. This is the original format of the UNIX gzip program.
- *deflate:* Using the zlib structure (defined in RFC 1950) with the deflate compression algorithm.
- *compress:* A format using the Lempel-Ziv-Welch (LZW) algorithm. The value name was taken from the UNIX compress program, which implemented this algorithm. Like the compress program, which has disappeared from most UNIX distributions, this content-encoding is not used by many browsers today, partly because of a patent issue that expired in 2003.
- *zstd:* Zstandard compression, defined in RFC 8478.
- *identity:* This value means that there is no compression process involved.

The preferences of which algorithm to use can be set on the HTTP headers during content negotiation, through the optional parameter of quality value (qvalue, q) that indicates the relative importance of the negotiable parameters. This qvalue is a real number (in HTTP/1.1 as much as 3 decimal digits on it) in the range between 0 and 1, where 0 is the minimum and 1 the maximum value. If a parameter has a qvalue of 0, then content with this parameter is "not acceptable" for the client. For instance, the headers could be:

- *Client request:*

[28] https://tools.ietf.org/html/rfc7807

```
GET /employees HTTP/1.1
Host: www.domain.com
Accept: application/json
Accept-Encoding: br;q=1.0, gzip;q=0.8, deflate;q=0.5, compress;q=0.1, *;q=0
```

- *Server response:*

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Encoding: br
Vary: Accept-Encoding
```

The meaning of '*;q=0' in the "Accept-Encoding" line is that 'identity' value and the others that may exist and aren't set in this header (with q>0) are not allowed, therefore stating that the client only allows compressed communications. The client preferences in this example for compression algorithms are, in order: br, gzip, deflate and compress.

The "Vary: Accept-Encoding" HTTP response header determines how to match future request headers to decide whether a cached response can be used rather than requesting a fresh one from the origin server, based on Accept-Encoding request header. This means we can make compressed responses cacheable.

Fourth, it is predictable that some API calls will require processing activities that take a while to finish. If the server waits for completion before sending a response to the client, it may cause unacceptable latency and it is possible that the request timeout finishes before the response is ready. If so, we need to consider making the operation asynchronous: for this long-time-consuming requests, the APIs need to be prepared for **asynchronous interaction**. There are several methods to allow asynchronous operations, like HTTP Asynchronous Request-Response pattern or the usage of Message Queues Broker (i.e. RabbitMQ, NATS).

Broker systems are more interactive than HTTP asynchronous polling, they are easier to deploy and more interactive, able to inform clients directly through the message queue. For these API long-time operations, a message queue should be created for each module that can transmit data asynchronously to the FLEXGRID Central Database and the module acting as the client, in other words the ATP GUI. While the server is processing the request, it sends via this queue the current status of the operation and a unique identifier for the result to be found on the database; the messages shall be consumed by the client, that will act accordingly the received message, and the final result stored in the FLEXGRID Central Database.

For instance, first the GUI calls a module API that returns an HTTP status code like "202 accepted" (to indicate the request was accepted for processing but is not completed), and the server starts sending to its queue the description of the operation status (like "The algorithm is running", "Loading data" or "End of run") when it changes, this messages being consumed by the client. When the job is done, the server may send to the queue the appropriate status and the result data **or** an identifier to find these results on the database.

Finally, some additional good practices to be taken into account for FLEXGRID S/W development and integration:

- **API versioning:** Versioning helps to iterate faster when some needed changes to the API are identified. There are several versioning philosophies, we can just embrace the most straightforward 'URI Versioning' including the version in the root API URL, for example:
  - http://api.exampleA.com/v1.5.2/
  - http://api.exampleB.com/v1/
  - http://apiv1.exampleC.com

  The version need not be numeric (like Major.Minor.Patch numbers), instead it can include dates, project names, seasons or other identifiers that are meaningful enough to the developers.

- **Health/Status resource:** It is a good idea to implement a health resource in the API, to know at any time if it is accessible and ready to operate or not.

- **Pretty printing** by default: All APIs should implement a variable like pretty=true or pretty=1 in the backend, for the clients being able to request JSON responses in human readable format (with blank/newline spaces). With this variable set to false (or 0) the API should behave cutting blank spaces and newlines, to reduce the response size. By default, set pretty variable to true (or 1).

4.2.2 API Web Authentication (API Keys) and Security principles

There are multiple authentication standards available today to secure a RESTful API, such as HTTP Basic Auth, API Keys, JWT, OAuth, etc. But, for being sure that RESTful APIs is stateless, the requests of authentication and authorization should not depend on cookies or sessions. Instead, each and every API request should come with some sort of authentication credentials, which must be validated on the server. **API Keys** are simple to use, they are short, static and don't expire unless revoked. They provide an easy way for multiple services to communicate but correct management for them is essential.

The API Key itself is an identity by which to identify the application or the user. It has to be unique, random and non-guessable or at least very difficult to guess. A good tool for generating the keys is RFC 4122, also known as Universally Unique Identifier (**UUID**). They can be generated by this tool and then incorporated into the application; there are several variants of UUID:

- *version 1:* time and node based,
- *version 3:* name based with MD5 hashing algorithm,
- *version 4:* random number based,
- *version 5:* name based with SHA-1 hashing algorithm.

Since the API Keys provide direct access to APIs' usage, they are very similar in concept to passwords, so there is a need for **securely store** them. This is because the application needs to make sure that the API Key in the request is valid and issued by the intended user, not by any malicious subject impersonating them. The API doesn't need to know the raw keys, but

just needs to validate that the received key is correct. For doing so, instead of storing them in plain text, they shall be stored as hashed values in the API server's database.

These **hashed values** protect the API usage in case someone gains unauthorized access to the API server's database, the raw keys are not leaked just the hashed value and it is almost impossible to guess the raw value from the hashed one. Then, for validating the raw keys that users are sending in their requests the API server hashes them and compare the value with the hashed keys it has stored, if they match anyone the received key gains validation.

A good practice is to assign each API key different permissions set and choose for each key specific actions that they are able to carry out. This can be done by providing **scopes**, where each scope represents a set of specific access permissions, grouping this way the API Keys affiliating them to scopes. In summary, use unique API keys per system functions set.

For instance, one key being used by FMO admin profile would have different access permissions than the FMO regular profile, that should use a different key from a different scope. For example:

- *Create* and *Read* FMO resources: FMO regular profile (i.e. 'FMO.reg_profile' scope) assigned to users able to check their Market operations.
- *Create, Read, Update* and *Delete* FMO resources: FMO admin profile (i.e. 'FMO.adm_profile' scope).

Authentication is verification of credentials or proving correct identity. **Authorization** is verification that the connection attempt is valid and certain actions are allowed. Authorization occurs after successful authentication: An API might authenticate an end user, but not authorize that end user to make certain requests. In our architecture, authentication shall be verified on ATP GUI through password, that will allow after success to use the API Key's scopes authorized for the logged user, which will be validated on the API servers on requests.

There are ways to check for proper authorization, such as via content-based access control (CBAC), role-based access control (RBAC) or policy-based access control (PBAC), ensuring that project data remains fully protected against unapproved access. **Role-based access control (RBAC)** fits well with FLEXGRID's API design, especially when combining API Key scopes with user profiles.

API Keys will travel raw in plain text on every call, that's why in order to protect them we need end-to-end encryption. HTTP can be enhanced to use SSL/TLS (secure transport layer technologies) public key certificates in order to encrypt the data in transit, becoming into **HTTPS protocol**. This is a fundamental security requirement.

Some additional best practices for securing REST APIs to consider:

- *Never expose sensitive information on URLs or responses:* Hide all API Security clues, like usernames, passwords, session tokens and API keys. This information should not appear in the URL, as this can be captured in web server logs, which makes them easily exploitable.

- *Input Parameter Validation:* Validate request parameters to sanitize the user input on the very first step, before it reaches to application logic. This measure helps avoiding potential malicious attacks (like injection or buffer overflow).
- *Containerize API server:* The use of containers (e.g. Docker) for encapsulate the API server bring several advantages like isolation, enhanced security, availability, portability, fast deployment, standardization, compatibility, portability, maintainability, continuous testing, etc.
- *Consider Adding Timestamp in Request:* This will help preventing very basic replay attacks from people who are trying to brute force the system. This point is not very prioritized since the API Keys shall travel through encrypted channels.
- *Rate limiting API keys (optional):* It is important to rate limit requests made with specific API Keys to ensure no bad actor can take down your API servers or cause performance issues that affect your other consumers. Having a proper rate limiting and monitoring solution keeps the API service functioning healthy.

4.2.3 OpenAPI Specification (OAS)

The OpenAPI Specification (**OAS)** [29] , formerly known as Swagger Specification, is a specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services. It is an API description format and defines a standard, language-agnostic interface to RESTful APIs. OpenAPI 3 is the latest and current version of this specification. This definition allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation or through network traffic inspection. When properly defined, a consumer/client can understand and interact with the remote service/server with a minimal amount of implementation logic.

An OpenAPI definition can then be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases. OpenAPI promotes a **contract-first approach**, rather than an implementation-first approach, which fits very well to FLEXGRID's scope.

Contract-first means that you design the API contract (the interface) first and then write code that implements this contract with logic. An API contract is a shared understanding of what the capabilities of a software interface are, allowing applications to be programmed on top of it. A **contract** should reflect a balance between the provider and the consumer interests, and provides a machine and human readable agreement that reflects the shared understanding of what an API may deliver.

An API contract is the documentation of the API itself, where the developer side declares its behaviour. There are several formats and tools allowing to create the contract and obtain the documentation, automated tests, mock servers, etc. All of these capabilities are covered by

[29] https://swagger.io/specification/

OAS, leaning on several **related tools** classified into categories (mock servers, documentation rendering, text and GUI editors, parsers, converters, etc) [30].

As stated before, the OAS document defines and describes an API conforming to OAS specification. Two important points are:

- *Media type:* The media type definitions shall be in compliance with RFC6838 (which defines procedures for the specification and registration of media types for use in HTTP, MIME, and other Internet protocols.)
- *HTTP Status Codes:* The HTTP Status Codes are used to indicate the status of the executed operation on the response. The available status codes are defined by RFC7231 [31], and the registered ones are listed in the IANA Status Code Registry [32].

4.3 FLEXGRID modules description

This section explains an overall description about the functionality, architecture and internal details for each FLEXGRID S/W module. For each one of them, a brief description of the internal pieces conforming their module, a functional diagram observing their interactions and the technologies used to build each module piece are included.

4.3.1 Automated Trading Platform (ATP)

4.3.1.1 Internal architecture

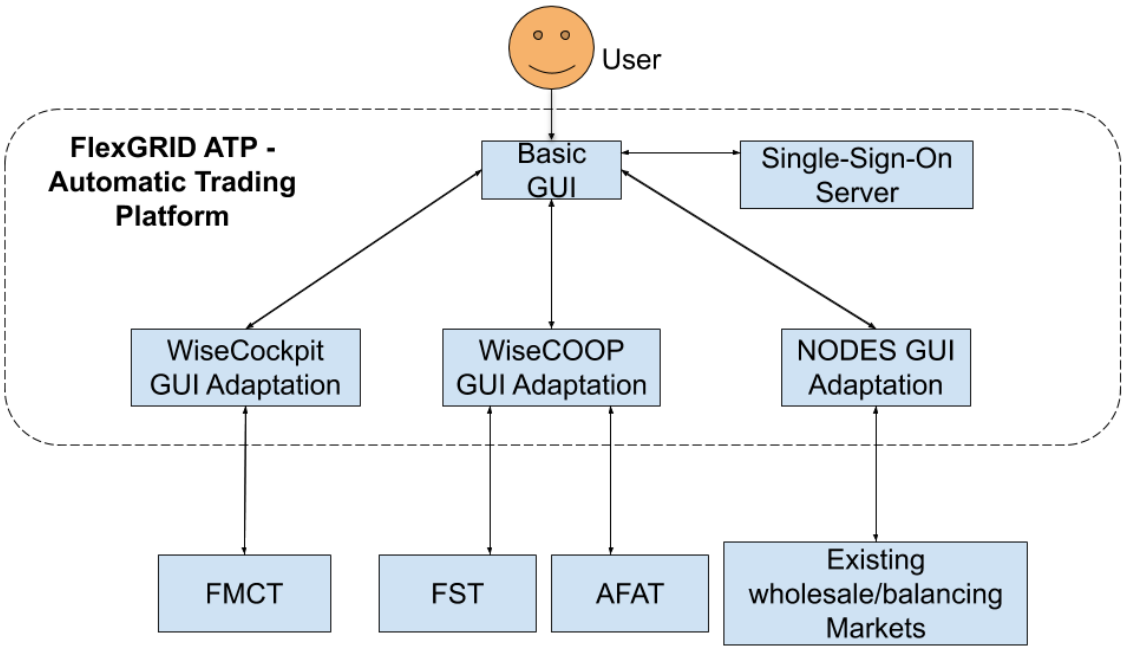

Figure 12 - The Automated Trading Platform (ATP) internal architecture

[30] https://openapi.tools/
[31] https://tools.ietf.org/html/rfc7231
[32] https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

4.3.1.2 System technical specifications

The ATP GUI will be composed by the submodules below:

- *Basic GUI:* This submodule will serve as a basic login facility, that will compare credentials with the SSO server, in order to determine authentication and authorization for the logged user. Upon successful log-in, the user will be redirected to the GUI subset it has access to, depending on user role. This will be built using Meteor.
- *Single-Sign-On (SSO) Server:* This one will evaluate if the logged user has credentials to access any of the ATP submodules, informing the Basic GUI where inside the ATP the user should be redirected if s/he has valid credentials. Implemented via OAuth and Keycloack.
- *WiseCockpit:* From WiseGrid project, this tool shall be adapted for use in FLEXGRID platform, being able to get information from the Central Database. The submodule will work as the application for DSO and TSO users. FLEXGRID will add intelligence to this tool through the use of FMCT module and respective novel mathematical models and algorithms residing therein.
- *WiseCOOP:* From WiseGrid project, this tool shall be adapted for use in FLEXGRID platform, being able to get information from the Central Database. The submodule will work as the application for Aggregator users, taking some UI ideas from SOCIALENERGY RABIT[33], too. FLEXGRID will add intelligence to this tool through the use of FST and AFAT modules and respective novel mathematical models and algorithms residing therein.
- *NODES GUI Adaptation:* GUI subset that will be built upon the actual NODES market application design, that will show the information from existing Markets (like market clearing results (i.e. prices) from wholesale and TSO balancing markets)[34].

All ATP submodules shall include a backend that will act as an API client for accessing Central Database, Market and AFAT, FST and FMCT modules. Adapted WiseGrid and Market applications shall be built by microarchitecture design, developed using: Meteor, NodeJS, MongoDB, Python3, SCALA, AKKA Streams, Spark, and some additional technologies.

4.3.2 Automated Flexibility Aggregation Toolkit (AFAT)

4.3.2.1 Internal architecture

Figure 13 illustrates the architecture of the Automated Flexibility Aggregation Toolkit (AFAT), with all the submodules that it includes.

[33] SOCIALENERGY RABIT is S/W tool that has been developed in the context of H2020-GA-731767 SOCIALENERGY project coordinated by ICCS.

[34] Real-life market clearing results will be provided by NPC based on Nord Pool's data portfolio. For FLEXGRID research purposes, a S/W agent will be implemented to emulate the operation of the Market Operator and the TSO. APIs will also be implemented to emulate the data exchange between the ATP and existing markets (i.e. day-ahead, intra-day, balancing, etc.).

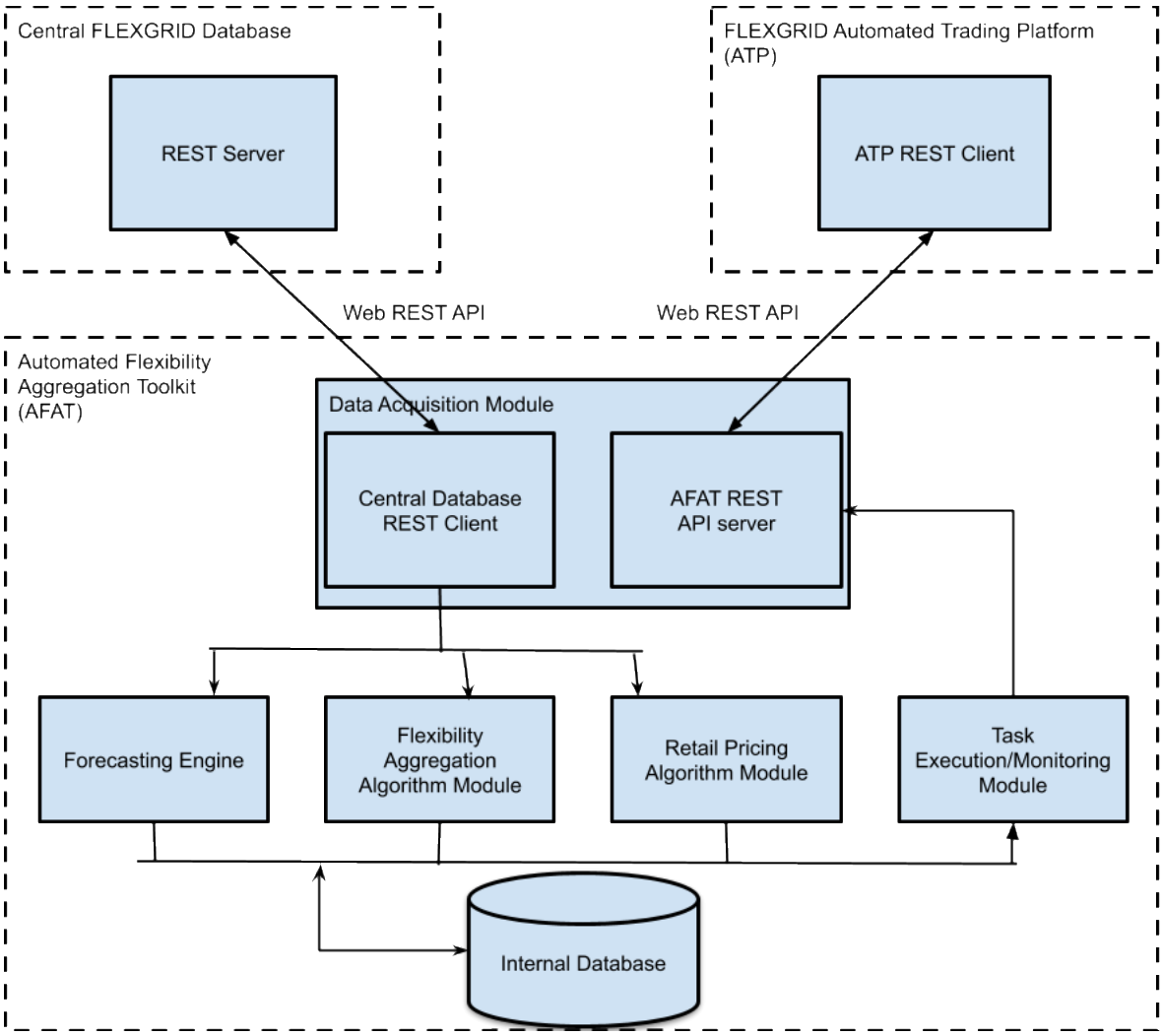

Figure 13 - The Automated Flexibility Aggregation Toolkit (AFAT) internal architecture

4.3.2.2 System technical specifications

The AFAT module/subsystem will be comprised by the following submodules:

- The *data acquisition submodule* will be responsible for communicating with the other modules of the FLEXGRID system (i.e. core FLEXGRID ATP and central database). Communication will be realized through JSON REST APIs. For the AFAT module specifically, there will be data exchange with the FLEXGRID Automated Trading Platform (ATP), and the Central FLEXGRID Database. Thus, the data acquisition submodule of the AFAT will contain:
  - The *central database REST client*, which is responsible for obtaining energy, flexibility and market data from the central FLEXGRID database, in order to use this information for the forecasting engine, the Flexibility Aggregation and Retail Pricing algorithm submodules. Communication with the Central Database module will be initiated by the AFAT module. Therefore, on the AFAT side, there will be a JSON REST web client.
  - The *AFAT REST API server*: Communication between ATP and AFAT is usually initiated by the ATP, in order to issue new FlexRequests, or new simulation runs, or to obtain

the status of the already submitted jobs. A REST or other client may also be included in this submodule, so that AFAT may inform the submitted jobs' status in real time (e.g. "the algorithm has received the input data", "the algorithm is running; please wait", "the algorithm's results are now being loaded in ATP GUI", etc.), when there are status updates.

The data acquisition module will be implemented using Ruby on Rails, taking advantage of existing software modules and experience available from the ICCS research team. The Ruby on Rails platform allows for the rapid development of web services, using best practices in web development with integrated REST support and also includes a REST client for consuming services from other modules.

- The *forecasting engine*: This module is used for generating forecasts with respect to energy consumption, energy production and storage, as well as market data (prices), that will be used by the flexibility aggregation algorithm submodule, in order to produce the Demand Response activation schedule, as a reaction to a FlexRequest that is received by the ATP.
- The *flexibility aggregation algorithm module*: This module is responsible for generating the Demand Response activation schedule, which consists of a set of end prosumers that will participate in the event by reducing their consumption or by raising their production. For each participating prosumer, a time-series of the demand reduction/production increase of each time interval will be produced, as well as an execution plan that will show which devices will participate for each prosumer at each timeslot. This submodule will be implemented in python3.
- The *retail pricing algorithm module* is responsible for providing an API that will be used for optimizing the pricing models and their parameters that the retailer will make available to its clients. This module will also be implemented using python3. This is because python3: a) has very mature optimization libraries, that will enable the production of optimal results with the least effort, b) the language and its libraries are both open source, which allows the algorithms to be used by any other researchers or interested parties without the need of acquiring any commercial licenses, c) the language is included by default in almost all Linux distributions, meaning that anybody can execute the module without the need of downloading and installing external libraries.
- The *task execution/monitoring submodule* will be responsible for the execution and monitoring of the tasks that have been submitted for execution in AFAT. This submodule will allow for submitting, canceling, and viewing the status of each submitted job by the user. It may also send notifications when there is a status update. This submodule will be integrated inside the Ruby on Rails web application and will schedule tasks using the ActiveJob API and the sidekiq execution engine.
- The *internal database* will be responsible for supporting the execution of all the submodules of AFAT, including the history of submitted jobs, parameters and results of the simulation runs. This internal database will be implemented in PostgreSQL.

4.3.3 FlexSupplier's Toolkit (FST)

4.3.3.1 Internal architecture

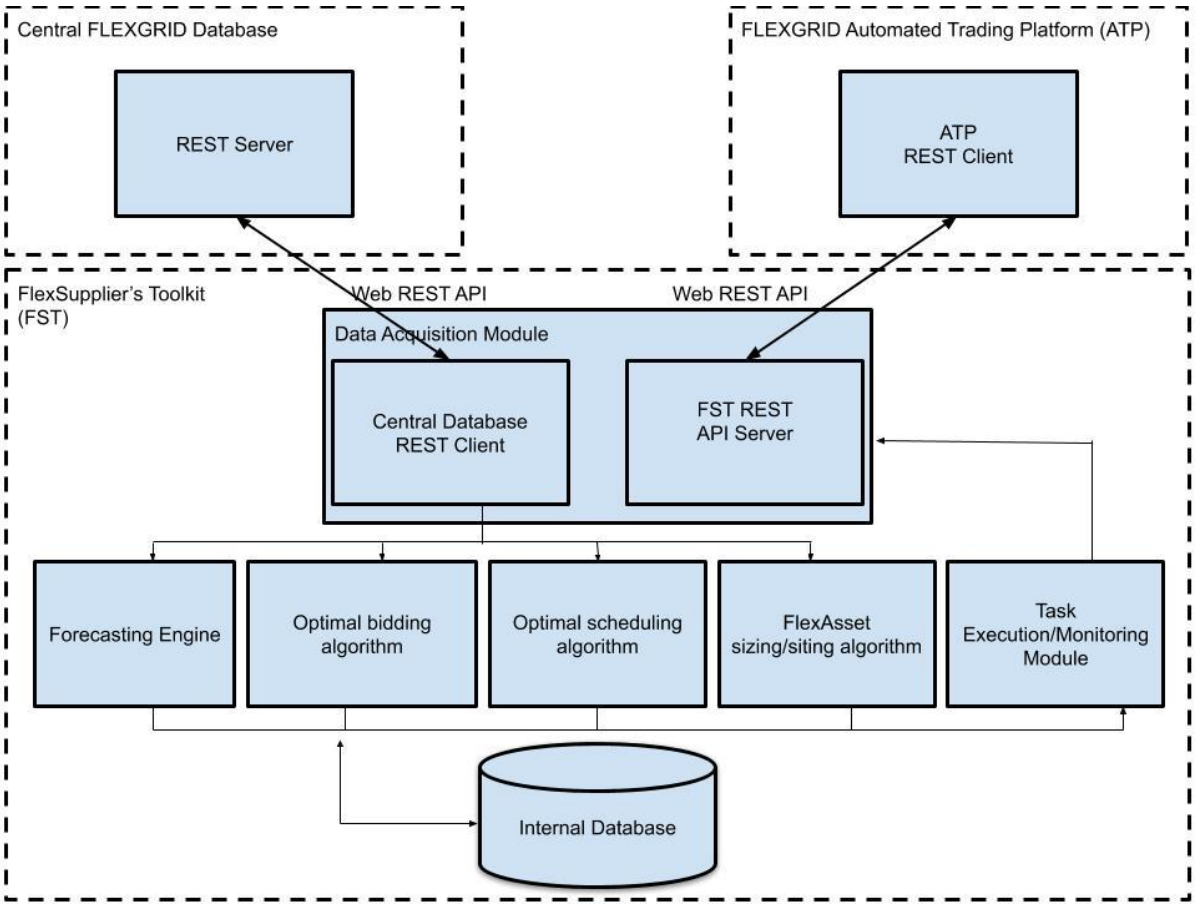

Figure 14 - The FlexSupplier's Toolkit (FST) internal architecture

Figure 14 illustrates the architecture of the FlexSupplier's Toolkit (FST), with all the modules that it includes.

4.3.3.2 System technical specifications

The FST module/subsystem will be comprised by the following submodules:

- The *data acquisition submodule* will be responsible for communicating with the other modules of the FLEXGRID system such as core FLEXGRID ATP and central database. Communication will be realized through JSON REST APIs. The FST module will exchange data with the FLEXGRID Automated Trading Platform (ATP) and Central FLEXGRID Database. Thus, the data acquisition submodule of the FST will contain:
    - The *central database REST client,* which is responsible for obtaining historical market data from the central FLEXGRID database. This data will then be used both by the forecasting engine and ESP CAPEX/OPEX minimization devoted submodules. FST module initiates the communication with the Central Database module so JSON REST web client will be on the FST side.
    - *The FST REST API server:* Communication between ATP and FST is usually initiated by the ATP, in order to issue new FlexRequests, or new simulation runs, or to obtain the status of the already submitted jobs. A REST or other client may also be included in this submodule, so that FST may inform the submitted jobs' status in real time (e.g. "the algorithm has received the input

data", "the algorithm is running; please wait", "the algorithm's results are now being loaded in ATP GUI", etc.), when there are status updates.

The data acquisition module will be implemented using Ruby on Rails.

- The *forecasting engine:* This submodule is used for generating forecasts. Forecasts include not only RES production (depending on the weather conditions), but also market prices, production and consumption in general.
- The *optimal bidding algorithm:* This submodule is one of the most vital parts of the FST. It generates optimal business strategies for the respective ESP considering different objectives such as CAPEX minimization or OPEX minimization. It includes bi-level optimization models, which take as input, among others, AC-OPF market clearing developed in FMCT as the lower-level problem. It results with the optimal bidding strategy considering respective constraints. The whole optimization problem will be developed in Python3.
- The *optimal scheduling algorithm:* This submodule is complementary to the Optimal bidding algorithm. They both present the core of the FST. Its task is to optimally schedule FlexAssets considering given constraints and objectives. The whole optimization problem will be developed in Python3.
- *The FlexAsset sizing/siting algorithm*: It will be responsible for sizing and siting of FlexAssets in an optimal manner according to the ESPs' input parameters. The input data will be given through easy to use GUI at ATP side and, after the algorithms are executed, the results will be sent back to the core ATP module.
- The *task execution/monitoring submodule* will be responsible for the execution and monitoring of the tasks that have been submitted for execution in FST. This submodule will allow for submitting, cancelling and viewing the status of each submitted job by the user. It may also send notifications when there is a status update. This submodule will be integrated inside the Ruby on Rails web application and will schedule tasks using the ActiveJob API and the sidekiq execution engine.
- The *internal database* will be responsible for supporting the execution of all the submodules of FST, including the history of submitted jobs, parameters and results of the simulation runs. This internal database will be implemented in PostgreSQL.

4.3.4 Distribution Flexibility Market Clearing Toolkit (FMCT)

4.3.4.1 Internal architecture

Figure 15 illustrates the architecture of the Flexibility Market Clearing Toolkit (FMCT), with all the modules that it includes.

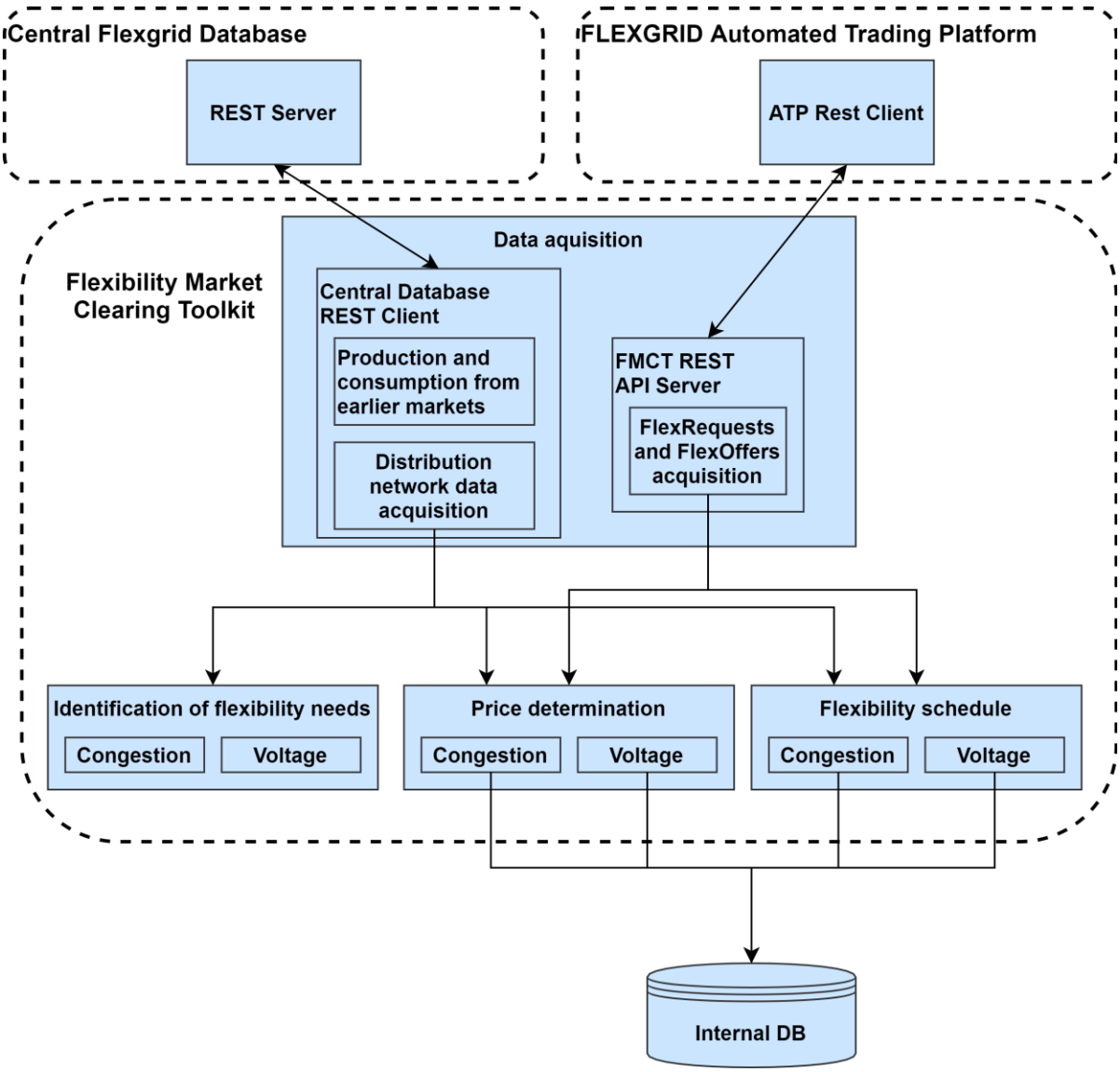

Figure 15 - The Flexibility Market Clearing Toolkit (FMCT) internal architecture

The different sub-modules in the Flexibility Market Clearing Toolkit (FMCT) are the following:

- *The FMCT REST API server:* Communication between ATP and FMCT is usually initiated by the ATP, in order to issue new FlexRequests, or new simulation runs, or to obtain the status of the already submitted jobs. A REST or other client may also be included in this submodule, so that FMCT may inform the submitted jobs' status in real time (e.g. "the algorithm has received the input data", "the algorithm is running; please wait", "the algorithm's results are now being loaded in ATP GUI", etc.), when there are status updates.
- *Data acquisition:* This module communicates with central FLEXGRID database (or other outside sources) to obtain the data necessary to run the market clearing algorithms. It also comprises the translation of the data in a format that can be used by the algorithms. The datasets to be acquired are:
    - The distribution network topology with nodes and lines, including the operating limits

    - The production and consumption schedules that were decided at earlier stages (e.g. in day-ahead market)
    - FlexRequests formulated by DSOs and FlexOffers formulated by ESPs

  There will be data exchange with the FLEXGRID Automated Trading Platform (ATP), and the Central FLEXGRID Database. Thus, the data acquisition submodule of the FMCT will contain:
    - The *central database REST client*
    - The *FMCT REST API server*
- *Identification of flexibility needs:* This sub-module helps the DSO for the creation of the FlexRequests. It identifies the potential line congestions and voltage deviations, based on the production and consumption schedules for a given time period and the distribution network data.
- *Flexibility schedule:* In this sub-module, the FlexRequests and FlexOffers are implemented and matched. The schedule is returned and sent to the general database.
- *Price determination:* This sub-module corresponds to the determination of market prices for congestion and voltage management. The prices determined are sent to the central database and are thereby stored.

4.3.4.2 System technical specifications

All the algorithms will be implemented using Python3[35].

[35] The optimization solver to be used for all S/W toolkits will be specified in subsequent deliverables D3.1, D4.1 and D5.1 for AFAT, FST and FMCT modules respectively.

5. FLEXGRID Use Cases architecture

5.1 Introduction

Only relevant UCSs have been selected to be developed for the S/W design, for being included into the SGAM diagrams that describe the platform architecture (i.e. TRL 5-6). This was done in order to narrow down the scope of S/W implementations (WP6 and WP7) emphasizing on FLEXGRID services that can be brought closer to the market stakeholders' needs at the end of the project's lifetime. The residual UCS described in D2.1 will be developed at lower TRLs 3-4 and will include ground-breaking and innovative research contributions in the context of WPs 3-5. In this section, the summary of each HLUC and their relevant UCSs are included, and also the associated SGAM diagrams for Component, Communication and Information layers.

5.2 HLUC_01 Description: FLEXGRID ATP offers advanced market clearing services to the Flexibility Market Operator (FMO)

This HLUC focuses on FLEXGRID ATP's operation and its interaction with incumbent markets and the underlying physical network operation. The initial idea is based on NODES business model in collaboration with Nord Pool Consulting (NPC) aiming at defining and developing advanced mathematical models and research algorithms. The aim is to define and develop advanced clearing models for the FMO that go beyond the state-of-the-art as described further in D2.1.

5.2.1 HLUC_01_UCS_02: Market-based local congestion management using FLEXGRID ATP in distribution networks using output from AC OPF model calculation as dynamic input for ATP

5.2.1.1 Use Case Scenario summary and context

As the existing electricity markets do not consider the constraints of local distribution networks, line congestions that were not accounted for in market clearing can arise especially when considering high RES penetration at the distribution network level in the future. As a result, gradually increasing re-dispatch volumes may be necessary, which imply that more expensive flexibility units will have to be running and renewable energy production could be curtailed.

Using an AC-OPF model makes it possible to evaluate the flow on each line of the distribution network and thus to identify potential constraints. This information can be used to relieve line congestions through the FLEXGRID ATP. This is possible because the AC-OPF model gives a good description of the distribution network, including line constraints and power losses on the lines.

The DSO must have filled in the platform its network topology details[36]. The DSO inputs the production and consumption schedules[37] and runs the OPF to estimate its need for flexibility to prevent line congestions. These production schedules are the results of wholesale markets (e.g. day-ahead market), which ran before the distribution level flexibility market (DLFM)[38]. They are given with the location of the producer or consumer in the network. The DSO uses the output of the OPF associated with bidding prices to constitute the FlexRequests. The ESPs on their side, give their FlexOffers on the ATP. The FMO runs the AC-OPF to clear the proposed distribution level flexibility market[39]. The outputs are the flexibility schedules and the prices for the trades[40]. These prices referred to as Distribution Locational Marginal Prices (d-LMPs).

5.2.1.2 SGAM Component Layer

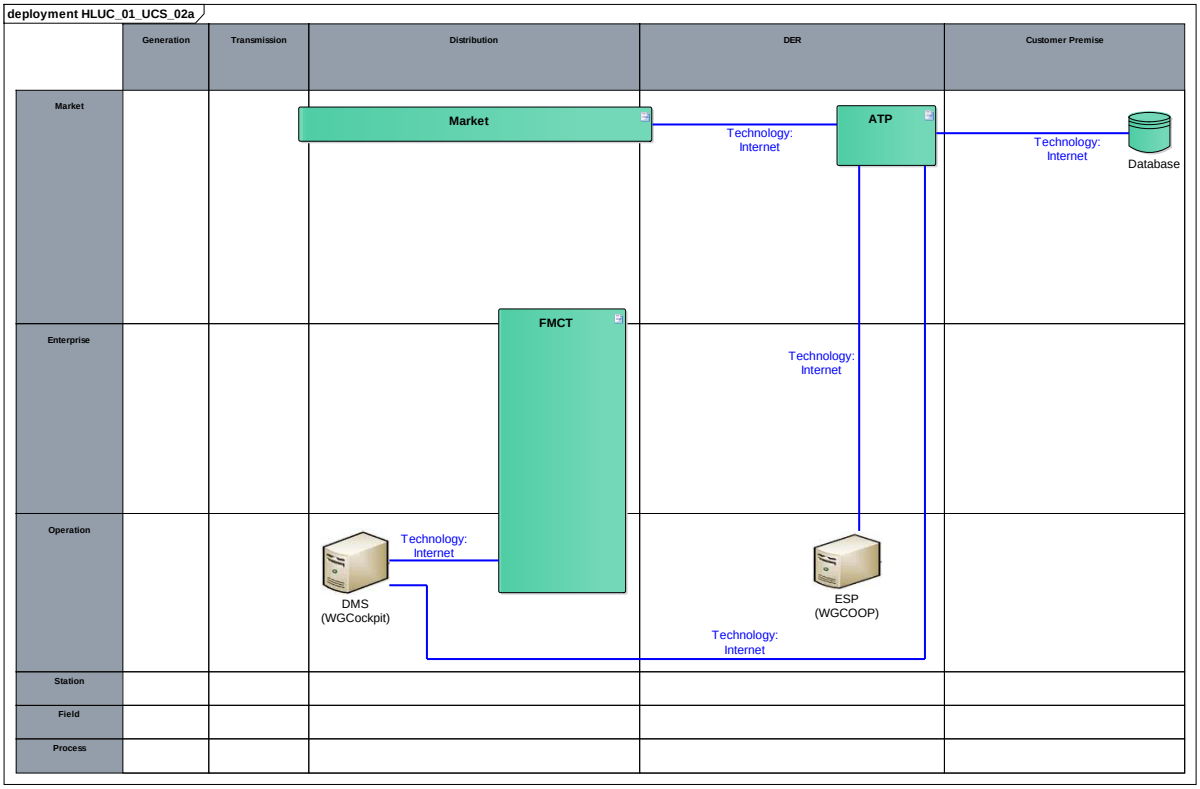

Figure 16 - HLUC_01_UCS_02 SGAM Component Layer

[36] See more in section 7.3 below. More specific details about respective data modelling will be provided in D5.1 (due to Month 12).

[37] Not possible in markets where schedules are notified on portfolio level. For example, Nord Pool does not have unit-level information. TSOs might share this information with DSOs where applicable.

[38] For more details, please read the proposed DLFM architectures and their interaction with existing markets in chapter 2 of this document.

[39] Please note that AC-OPF will be used by DSO in order to create the FlexRequests and then by the FMO in order to derive the market clearing results (i.e. the objective is to minimize the flexibility cost).

[40] Scheduling may not be on a FlexUnit level, but on FlexAsset/end prosumer level. The aggregator/ESP should decide which units to schedule. The FMO may not schedule units on household level, e.g. individual water boilers or heating panels.

5.2.1.3 SGAM Communication Layer

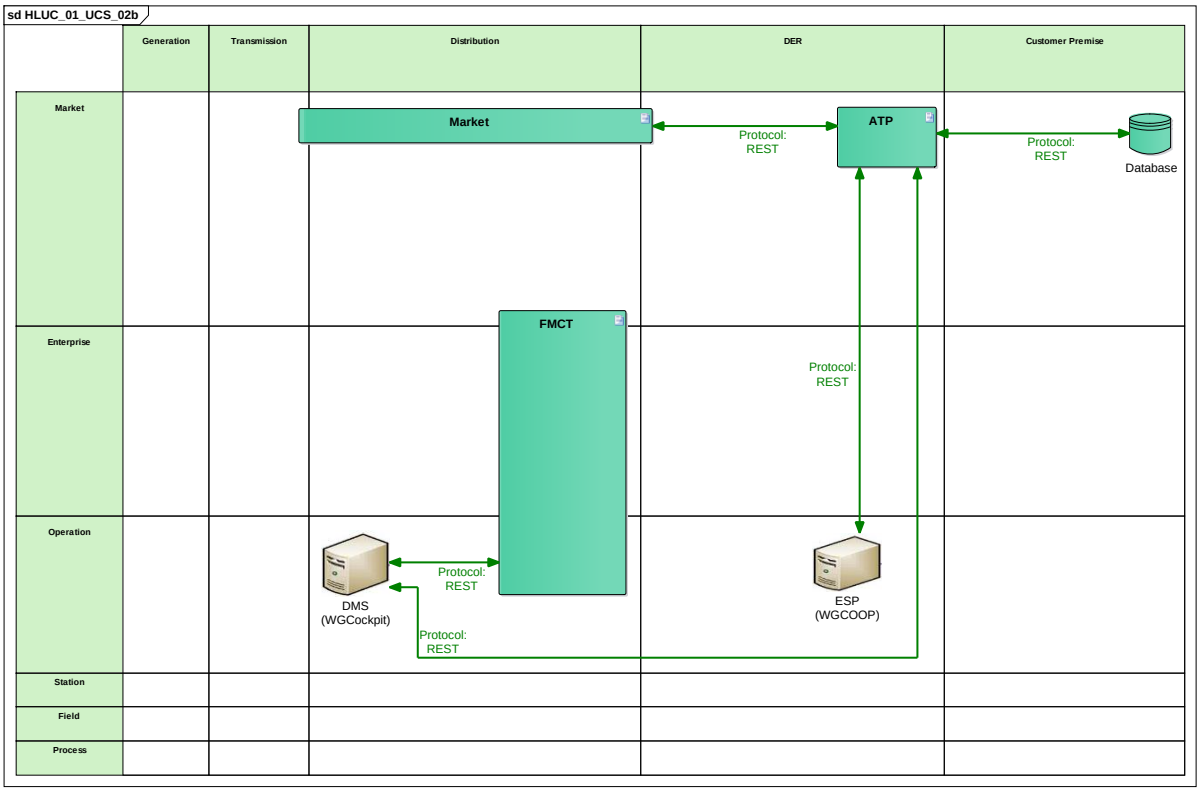

Figure 17 - HLUC_01_UCS_02 SGAM Communication Layer

5.2.1.4 SGAM Information Layer

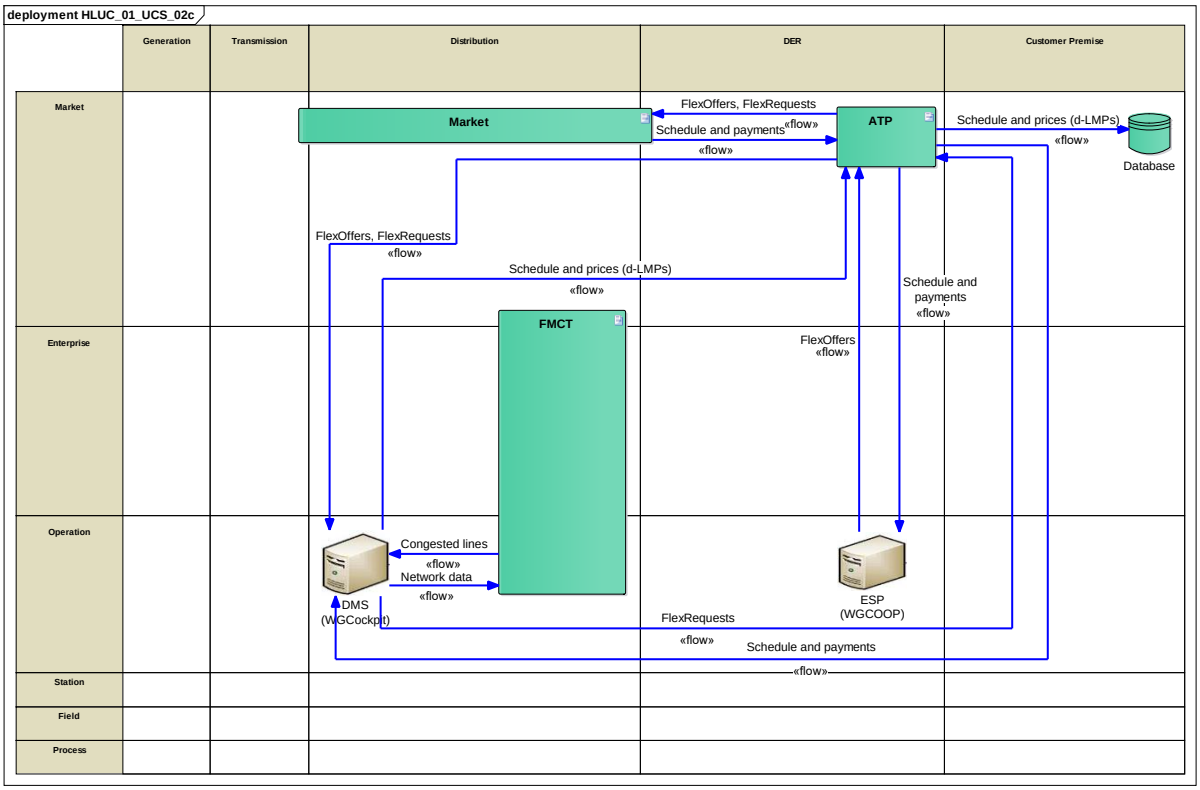

Figure 18 - HLUC_01_UCS_02 SGAM Information Layer

5.2.2 HLUC_01_UCS_03: Market-based local voltage control using FLEXGRID ATP in distribution network operation (Q-LMP market clearing)

5.2.2.1 Use Case Scenario summary and context

As the existing electricity markets do not consider the constraints of local distribution networks, voltage deviations that were not foreseen can arise. In particular, the models used in the existing markets, do not take into account reactive power. As a result, re-dispatches may be necessary, which imply that more expensive units will have to be running to provide reactive power. Using an AC-OPF model makes it possible to anticipate/estimate the voltage level at every node of the distribution network and thus to identify/forecast voltage deviations. This information can be used to relieve voltage deviations through the FLEXGRID ATP.

The DSO must have filled in the platform its network topology and details (see more details in Annex 7.4)[41]. The DSO inputs the production and consumption schedules and runs the OPF to determine the need for flexibility to prevent voltage deviations. These production schedules are the results of wholesale markets, which ran before the flexibility market. They are given with the location of the producer or consumer in the network. The DSO uses the output of the OPF associated with bidding prices to constitute the FlexRequests. The ESPs on their side, give their FlexOffers on the ATP. The FMO runs the AC-OPF to clear the market. The outputs are the flexibility schedules for the provision of reactive power and the prices for the trades. These prices referred to as Locational Marginal Prices for Reactive Power (Q-LMPs).

[41] It should be noted that the more accurate the distribution network topology datasets are, the better and more accurate AC-OPF results will be. FLEXGRID envisages cases in which DSOs will provide from basic network topology information (like the way it is done in NODES) to more complex and real-time information, which will allow the even better operation of the proposed distribution level flexibility market (DLFM).

5.2.2.2 SGAM Component Layer

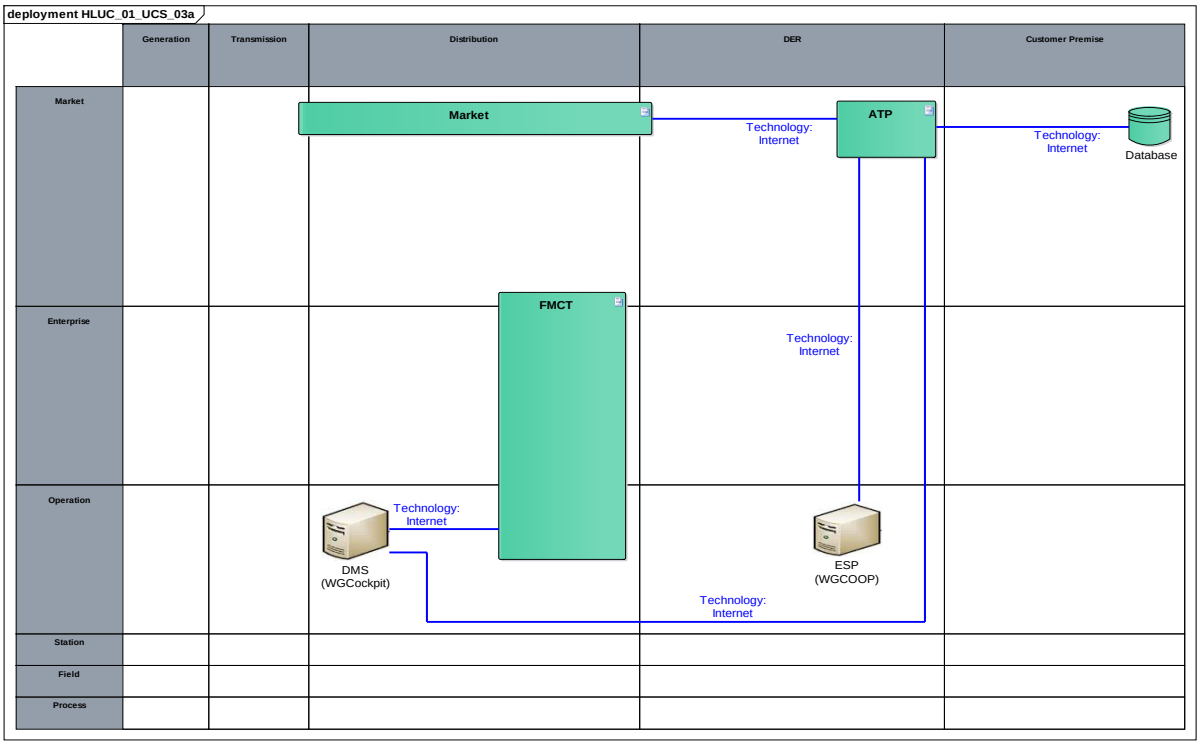

Figure 19 - HLUC_01_UCS_03 SGAM Component Layer

5.2.2.3 SGAM Communication Layer

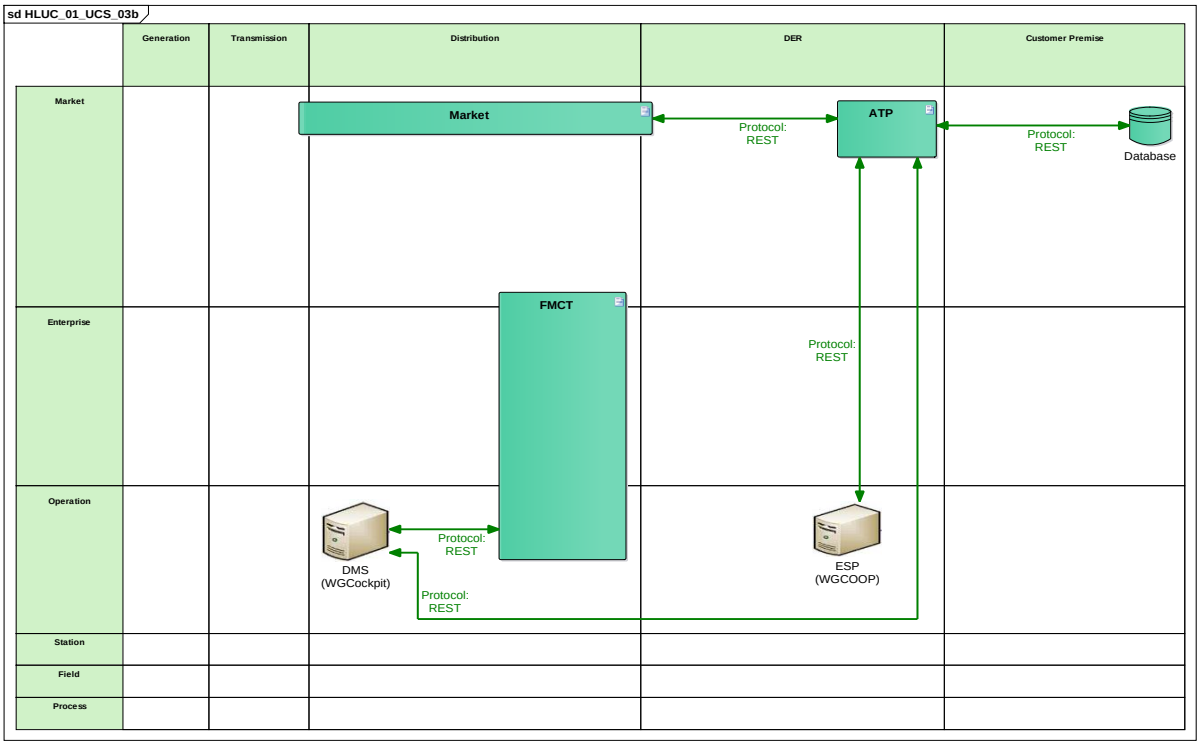

Figure 20 - HLUC_01_UCS_03 SGAM Communication Layer

5.2.2.4 SGAM Information Layer

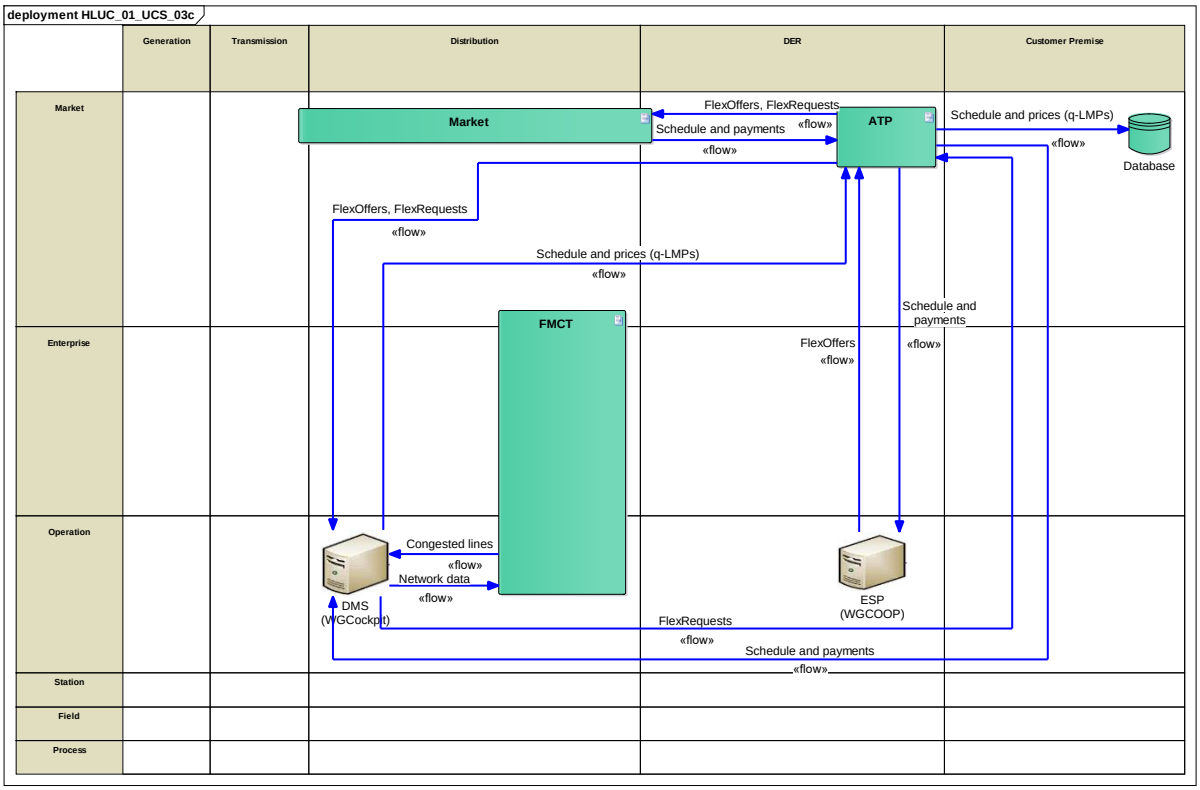

Figure 21 - HLUC_01_UCS_03 SGAM Information Layer

5.2.3 HLUC_01_UCS_04: FLEXGRID ATP operates as a gateway to redirect local active power flexibility to different TSO platforms (interaction with existing TSO markets)

5.2.3.1 Use Case Scenario summary and context

In the upcoming future of high RES penetration contexts, TSO and DSO are expected to compete[42] for the same flexibility in order to fulfil their mission. FLEXGRID project assertion is that to the extent possible this competition should happen through the use of flexibility provided via a flexibility market within the grid location determined by the DSO and/or TSO. In cases where distributed flexibility has the highest value for the DSO and the local congestion/voltage management, flexibility stays local[43]. And vice versa, in cases where distributed flexibility has the highest value for the TSO, flexibility is procured in the balancing markets operated by the TSO. The key feature of the proposed FLEXGRID ATP is the possibility to identify (through a location tag) and give a value (by putting a price tag on flexibility) to FlexSuppliers (or else ESPs). This opens new opportunities for grid operators (i.e. both DSO and TSO) that can procure distributed flexibility that lies at the distribution network level. DSOs are enabled to contract local flexibility to solve grid issues, while TSOs get access to smaller flexibility units that are currently excluded from traditional TSO markets. This

[42] For the purpose of grid security, there should be some kind of coordination to prevent competition between the TSOs and DSOs (see more in UCS 3.1 and 3.2 below).

[43] FLEXGRID models and algorithms will define this value of flexibility in every part of the grid and in every timeslot.

integrated approach ensures that flexibility can be purchased and activated where it has the highest value (i.e. for local congestion in the DSO grid or balancing market for the TSO).

5.2.3.2 SGAM Component Layer

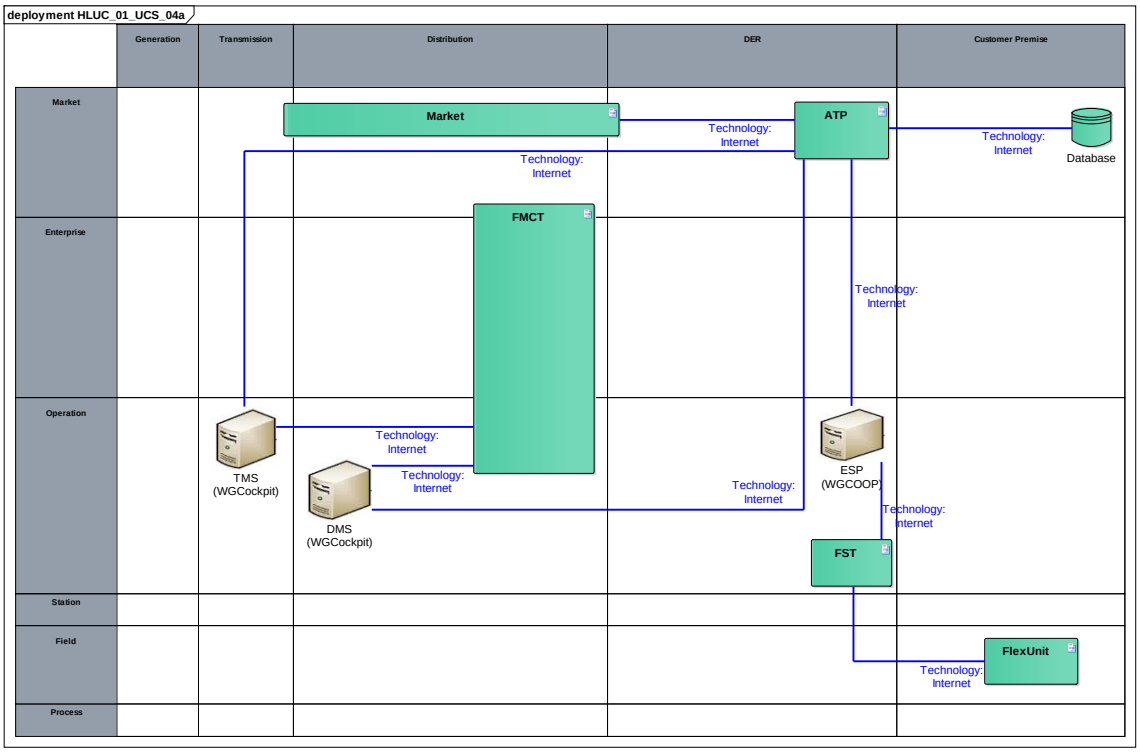

Figure 22 - HLUC_01_UCS_04 SGAM Component Layer

5.2.3.3 SGAM Communication Layer

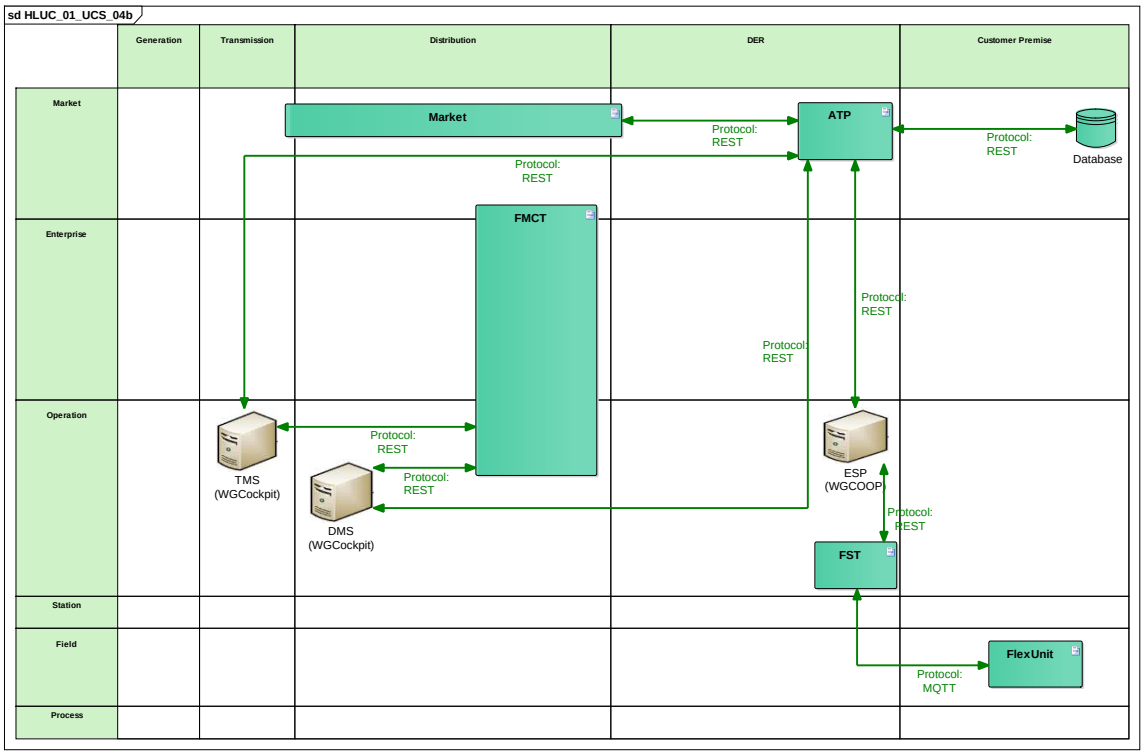

Figure 23 - HLUC_01_UCS_04 SGAM Communication Layer

5.2.3.4 SGAM Information Layer

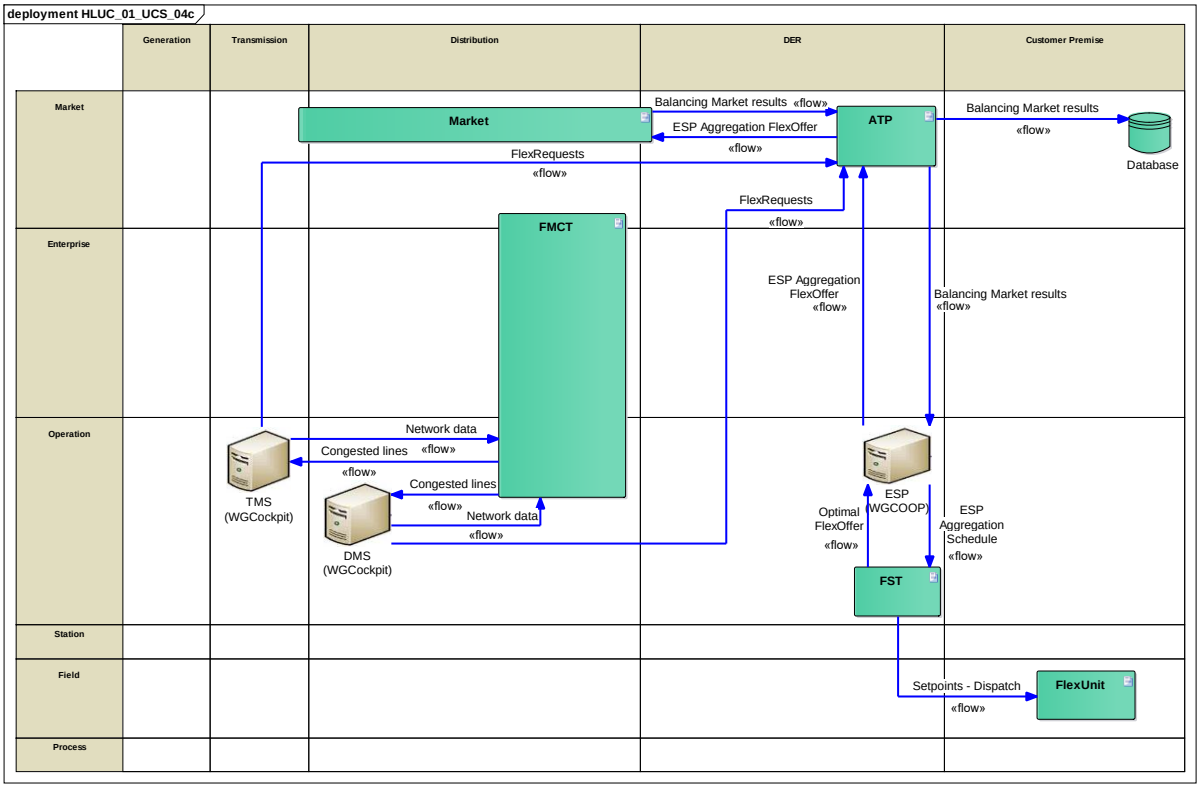

Figure 24 - HLUC_01_UCS_04 SGAM Information Layer

5.3 HLUC_02 Description: FLEXGRID ATP offers advanced flexibility supply management services to Energy Service Providers (ESPs)

This HLUC focuses on FLEXGRID ATP's operation (in collaboration with the proposed FlexSupplier's Toolkit - FST) and its support to 'Energy Service Providers' (i.e. FlexSupply side of the proposed flexibility marketplace). The increasing share of renewable energy in all European markets and the respective continuously increasing energy re-dispatching costs (i.e. costs incurred due to growing discrepancy between market outcome and physical reality, especially in bidding zones with high iRES) leads to more or less pronounced price fluctuations in the spot market [44]. In 2018, the costs for redispatch and feed-in management of renewables, including the costs for maintaining the so-called grid reserve, amounted to almost EUR 1.5 billion. In 2017, costs also reached EUR 1.5 billion, while in years 2012-2014, respective costs did not surpass EUR 0.4 billion.

[44] L. Hirth, I. Schlecht, C. Maurer, B. Tersteegen, "Cost or market-based? Future redispatch procurement in Germany", October 2019, available online: https://www.bmwi.de/Redaktion/EN/Publikationen/Studien/future-redispatch-procurement-in-germany.pdf?__blob=publicationFile&v=3

Operators of decentralized controllable devices can benefit from these price fluctuations by shifting the electricity generation or electricity consumption in times of economically advantageous market prices. Ideally, they use an external service provider for this – a so called 'Energy Service Provider' (ESP). ESP is a general term that is used in the FLEXGRID project. It means a profit-oriented company, which may make contractual arrangements with various types of flexibility assets owners (e.g. DSM, RES, storage). An ESP may offer various types of services to TSOs/DSOs and BRPs.

5.3.1 HLUC_02_UCS_01: ESP minimizes its OPEX by optimally scheduling the consumption of its end users, the production of its RES and its storage assets

5.3.1.1 Use Case Scenario summary and context

Energy Service Providers (ESPs), as profit-oriented companies seek to optimize their asset portfolio to maximize their profit by participating in a number of capacity and/or energy markets. Generally, an ESP operates flexible loads, inflexible loads, Energy Storage Systems (ESSs) and production of Renewable Energy Sources (RES). Each of these elements has its own unique set of characteristics, resulting in different benefits and challenges to the ESP's business strategy. In order to exploit advantages and minimize negative impacts (e.g. high OPEX/CAPEX) of certain technologies, all relevant aspects, including high-fidelity models[45], need to be considered as well as specific characteristics and opportunities that various markets present. FlexSupplier's (FST) toolkit provides ESPs the needed mathematical models and algorithms to approach the problem in a holistic and novel way. The algorithm on the FlexSupplier side aims at minimizing OPEX. By optimally scheduling market actions (buying/selling) and operation of its resources (e.g. energy storage charging/discharging, shifting the consumption of flexible loads), the developed tools will result in an operating schedule, which finds the minimum OPEX, while respecting the technical and market constraints. To do so, topology data combined with high-end forecasting models are of the utmost importance[46].

[45] Means a mathematical model that represents as close as possible the real life situation.

[46] According to the Art. 32 of the recently released Clean Energy Package, DSOs are expected to be obliged to make public some basic information about their network topology and constraints in order for RES and FlexAsset investments to be incentivized. Moreover, DSOs are expected to derive more advanced mathematical models in the future in order to make public information, which is related to geographical locations, where RES and FlexAsset investments should be prioritized (and maybe other locations where more RES and FlexAsset installations may not have a good Return of Investment or even cause more technical problems than the ones that they aim to heal).

5.3.1.2 SGAM Component Layer

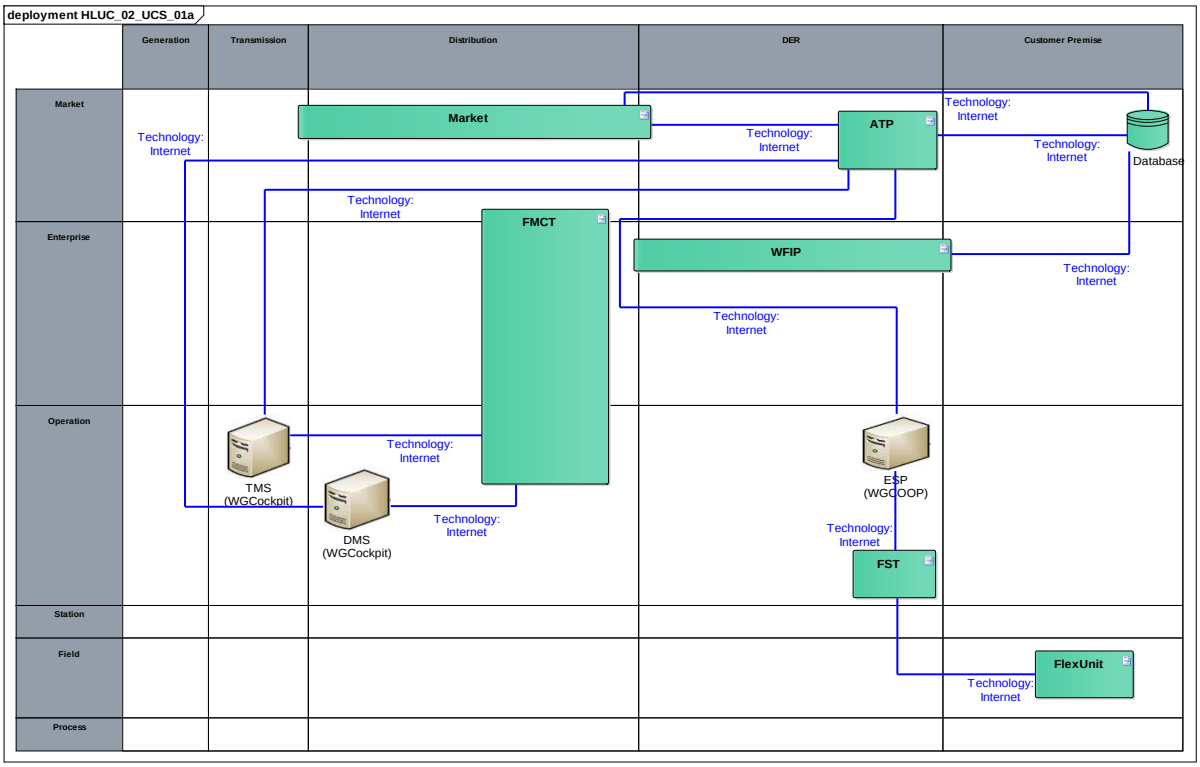

Figure 25 - HLUC_02_UCS_01 SGAM Component Layer

5.3.1.3 SGAM Communication Layer

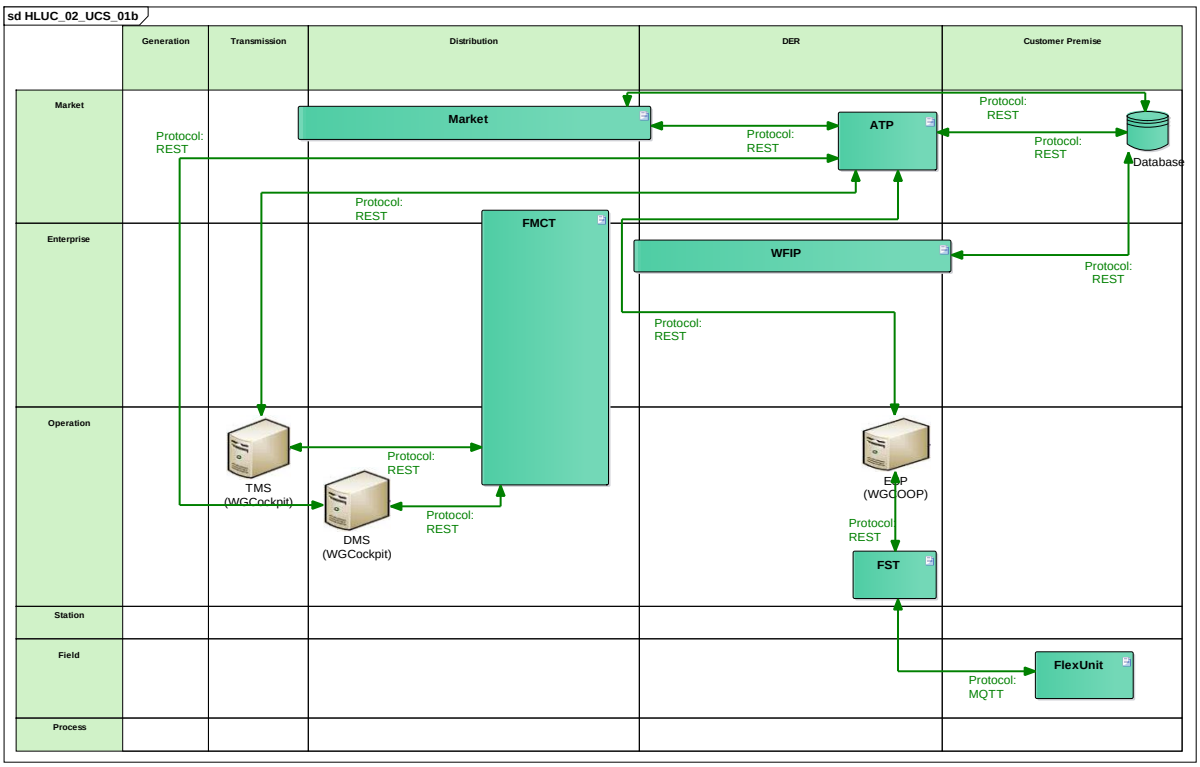

Figure 26 - HLUC_02_UCS_01 SGAM Communication Layer

5.3.1.4 SGAM Information Layer

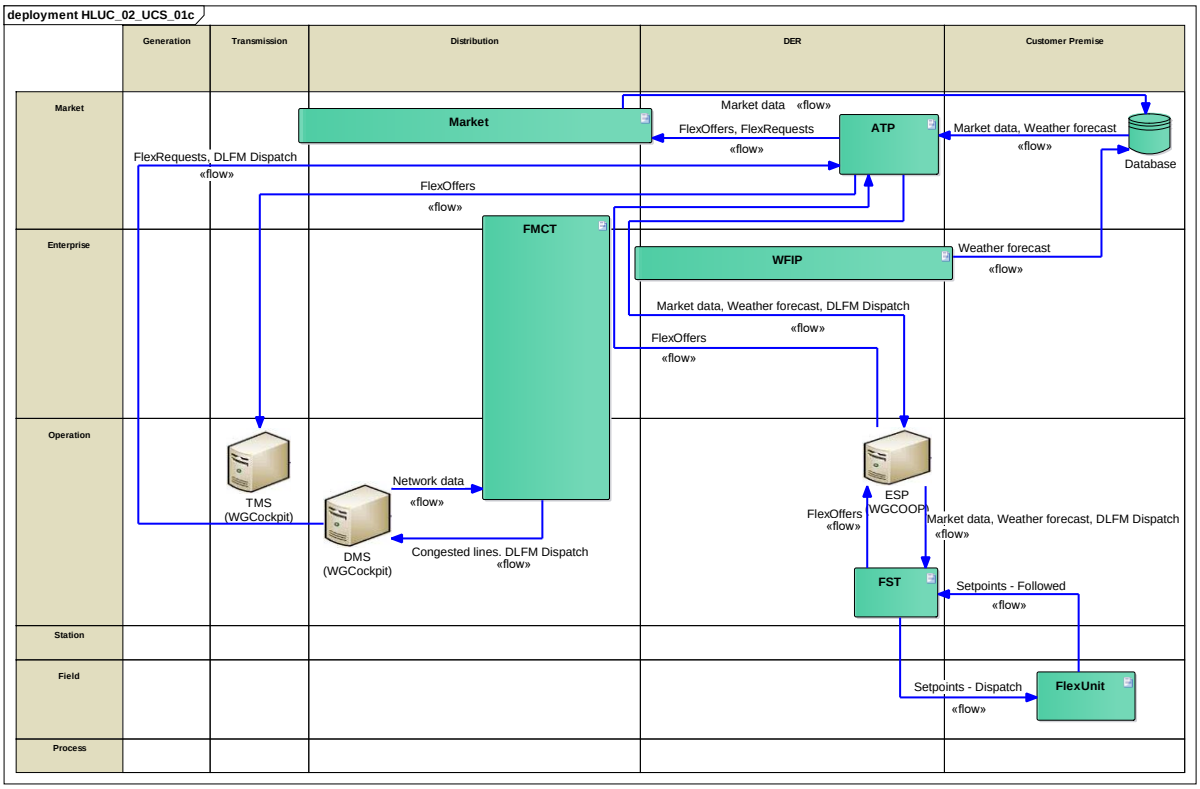

Figure 27 - HLUC_02_UCS_01 SGAM Information Layer

5.3.2 HLUC_02_UCS_02: ESP minimizes CAPEX by making optimal investments on RES and FlexAssets

5.3.2.1 Use Case Scenario summary and context

ESPs' portfolio includes flexible loads, inflexible loads, RES and Energy Storage Systems (ESSs). The latter two present a massive capital expenditure for an investor. Thus, there is a strong incentive to minimize the CAPEX. As capital investments are long-term decisions, multi-stage planning is also a possibility to hedge against potential inaccuracies in predictions regarding the future market trends, demand curves, weather conditions and other variables that could influence ESP's earnings. The RES and FlexAsset investment problem is observed from multiple angles, because RES and FlexAsset investment success is dependent on a wide variety of uncertain parameters. Thus, FlexTools provide a deep techno-economic analysis of the current electricity market characteristics and future trends[47]. FlexSupplier's Toolkit (FST) provides a comprehensive interpretation of the developed models and co-observes them with optimal bidding strategies in various markets. Such thorough analysis identifies the most attractive electricity markets to participate in, considering technical constraints and CAPEX-to-profit ratio. Furthermore, due to possible future techno-economic trends, it considers a

[47] More details about the nature and characteristics of this techno-economic analysis will be provided in subsequent deliverable D4.1 (due to Month 12).

multi-stage investment plan[48] in order to intelligently acquire the assets with the lowest CAPEX possible while obeying all the constraints and requirements on the optimal sitting and sizing of the relevant assets.

5.3.2.2 SGAM Component Layer

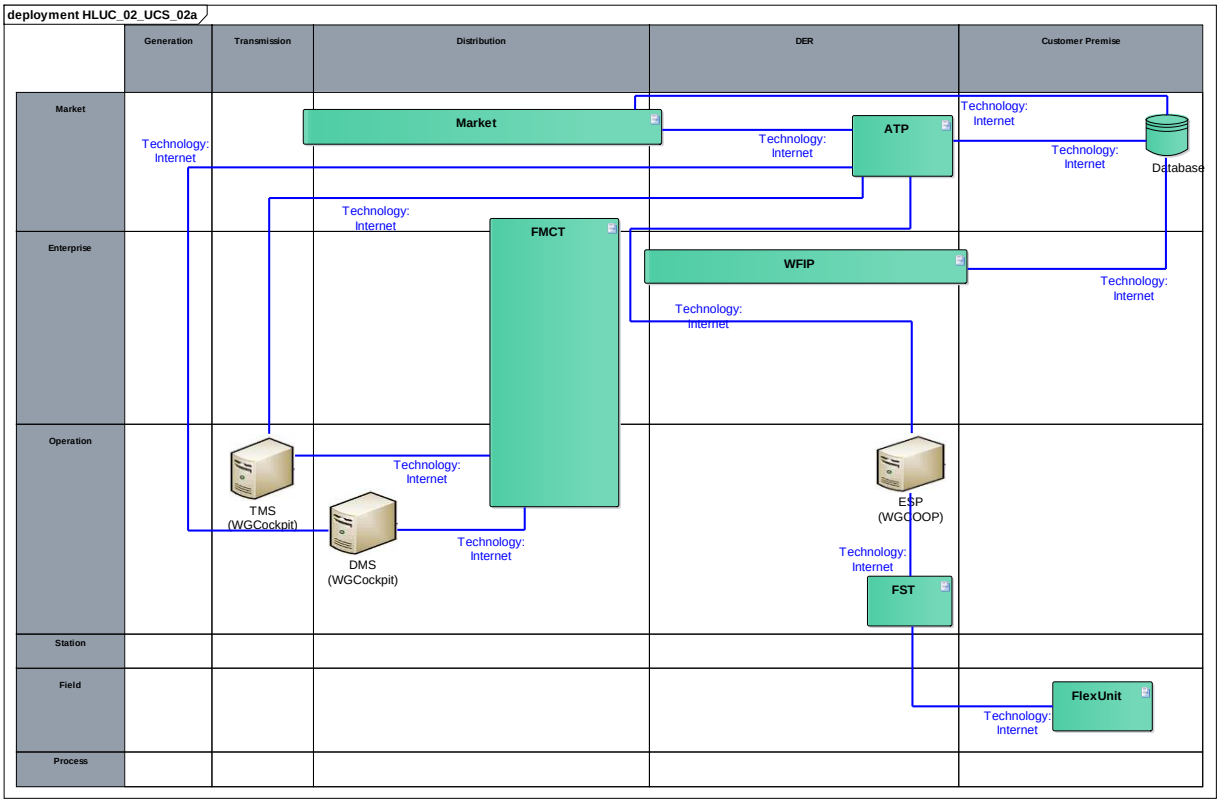

Figure 28 - HLUC_02_UCS_02 SGAM Component Layer

[48] A multi-stage investment plan executes optimization procedure in multiple steps as some variables become input parameters. Furthermore, as the time passes, new relevant information/trends could appear and should be modelled, which heavily affect former optimization process and re-optimization is needed. Accurate answer of the investment horizon will be defined in a later stage of the research as part of WP4 deliverables.

5.3.2.3 SGAM Communication Layer

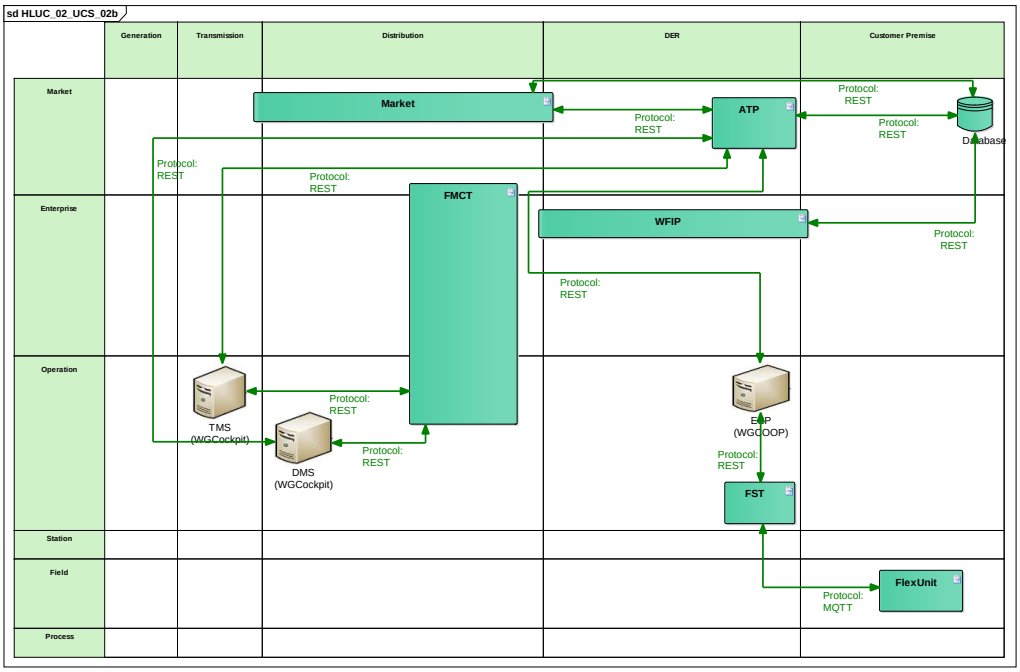

Figure 29 - HLUC_02_UCS_02 SGAM Communication Layer

5.3.2.4 SGAM Information Layer

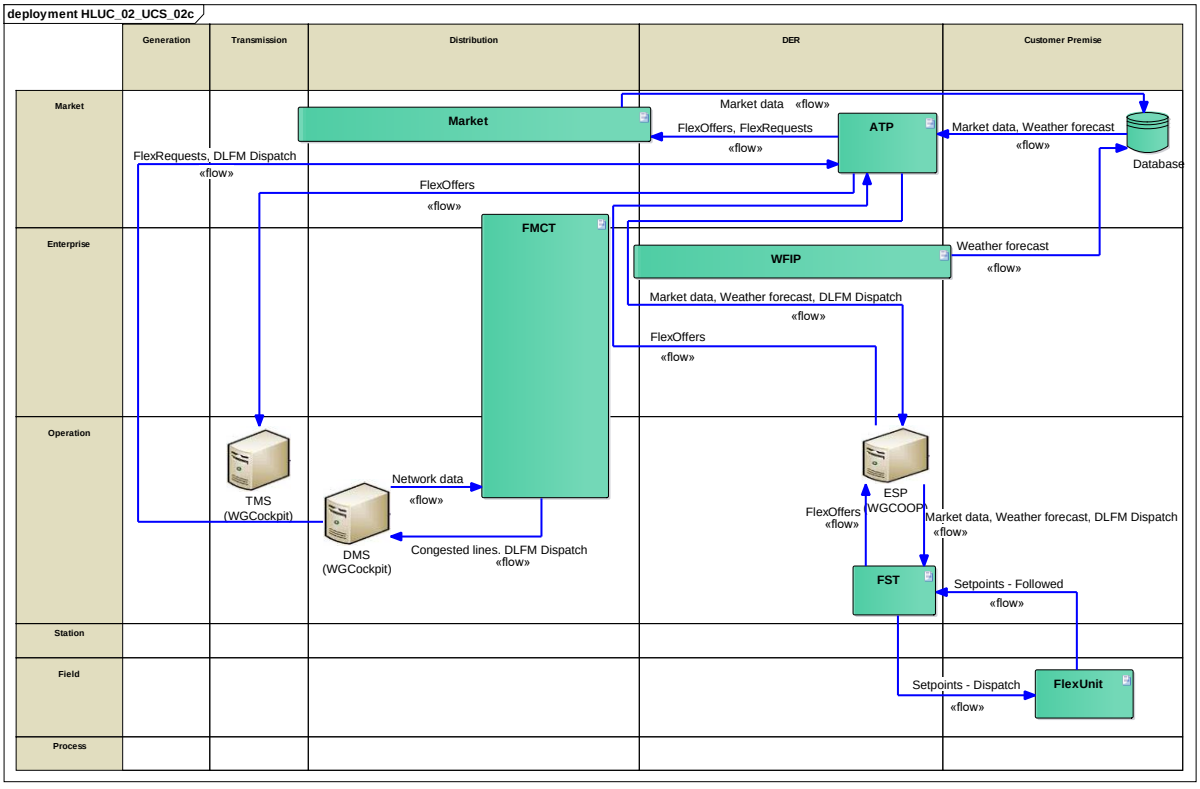

Figure 30 - HLUC_02_UCS_02 SGAM Information Layer

5.3.3 HLUC_02_UCS_03: ESP maximizes its profits by co-optimizing its participation in several energy and local flexibility markets

5.3.3.1 Use Case Scenario summary and context

Congestion management and frequency/voltage control issues caused by high RES penetration increase the volatility of energy prices in various existing energy markets (e.g. day-ahead, intra-day, balancing, reserve markets) as well as in emerging local flexibility markets (e.g. Distribution Level Flexibility Markets - DLFMs proposed by FLEXGRID)[49]. This volatility offers the potential for energy arbitrage (buy power when the price is low and sell it during high-price time periods) and respective revenues for ESPs, who own and invest on Energy Storage Systems (ESS).

In this Use Case Scenario (UCS), we consider a profit seeker Energy Service Provider (ESP), who owns a set of Energy Storage Units (ESUs) located in different transmission network buses and nodes of a radial distribution network. In order to maximize its profits, ESP joins several energy markets and dynamically optimizes its bidding strategy. In more detail, it exploits: market price forecasters, energy prosumption forecasters and information relevant with the underlying network topology in order to derive an optimal scheduling and bidding strategy to maximize its business profits[50]. Without lack of generality, we assume ESP's participation in three markets, namely: 1) Day Ahead Energy Market (DA-EM), 2) (day-ahead) Reserve Market (RM), and 3) Day Ahead Distribution Level Flexibility Market (DA-DLFM).

The objective function of the ESP's problem is to maximize its aggregated profits from the three aforementioned markets. As far as the day-ahead wholesale energy market is concerned, the ESP decides the ESUs' operation by taking as input the nodal price, which corresponds to the node of the transmission grid on which the distribution network connects with. Secondly, the ESP earns a profit by providing upward and downward reserves in the day ahead reserve market. 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 DSO's day-ahead flexibility market, in which it gets paid for its flexibility services relevant with active and reactive power (P-flexibility and Q-flexibility) based on nodal prices that are calculated in the day-ahead DLFM[51].

[49] We assume that in the proposed DLFM, where many small and distributed RES and FlexAssets will exist in small-scale geographical areas and distribution networks, the volatility of energy prices will be high. Within FLEXGRID, extensive simulations of many DLFM variants and conditions will take place.

[50] More details will be provided at later stages of research via the delivery of D4.1 (due to Month 12).

[51] DLFM is, for the time being, only theoretical market, thus extensive simulation analysis and "what-if" scenarios will take place. Multiple simulation scenarios will be conducted and afterwards all of them will be analysed. Simulation will be held in a manner that it reaches a convergence point.

5.3.3.2 SGAM Component Layer

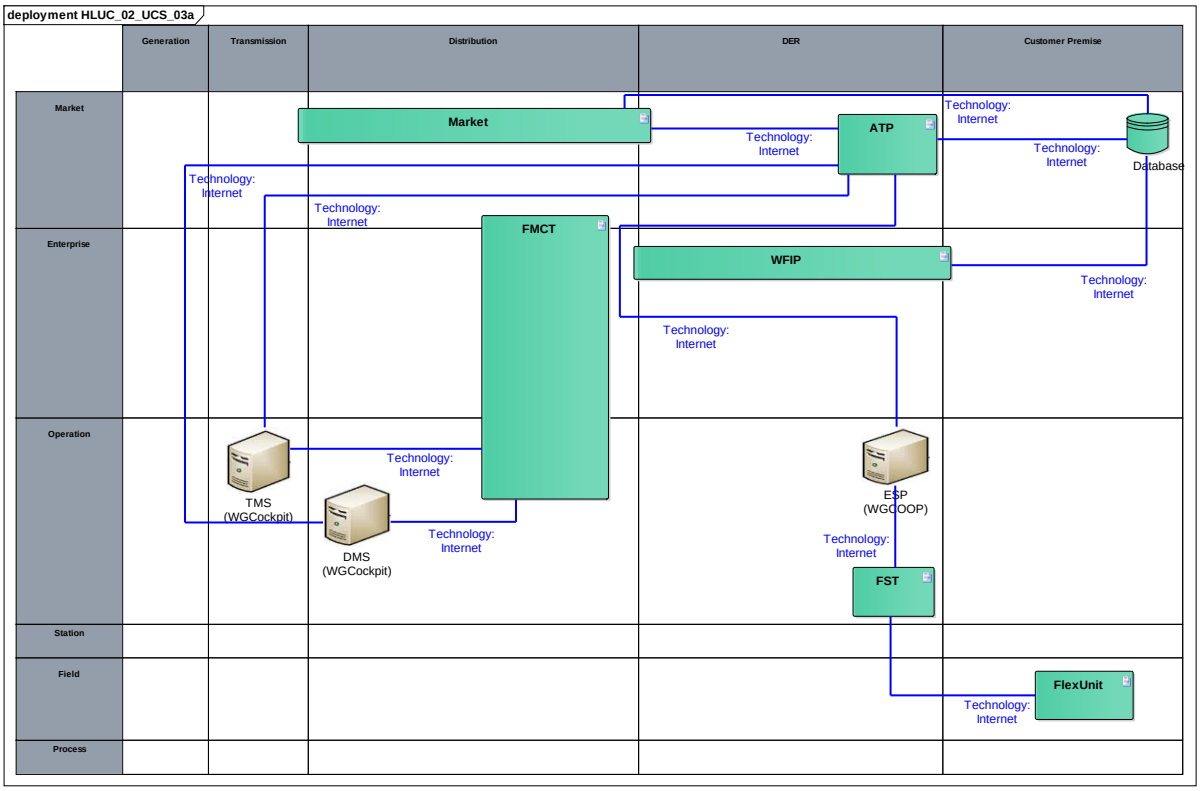

Figure 31 - HLUC_02_UCS_03 SGAM Component Layer

5.3.3.3 SGAM Communication Layer

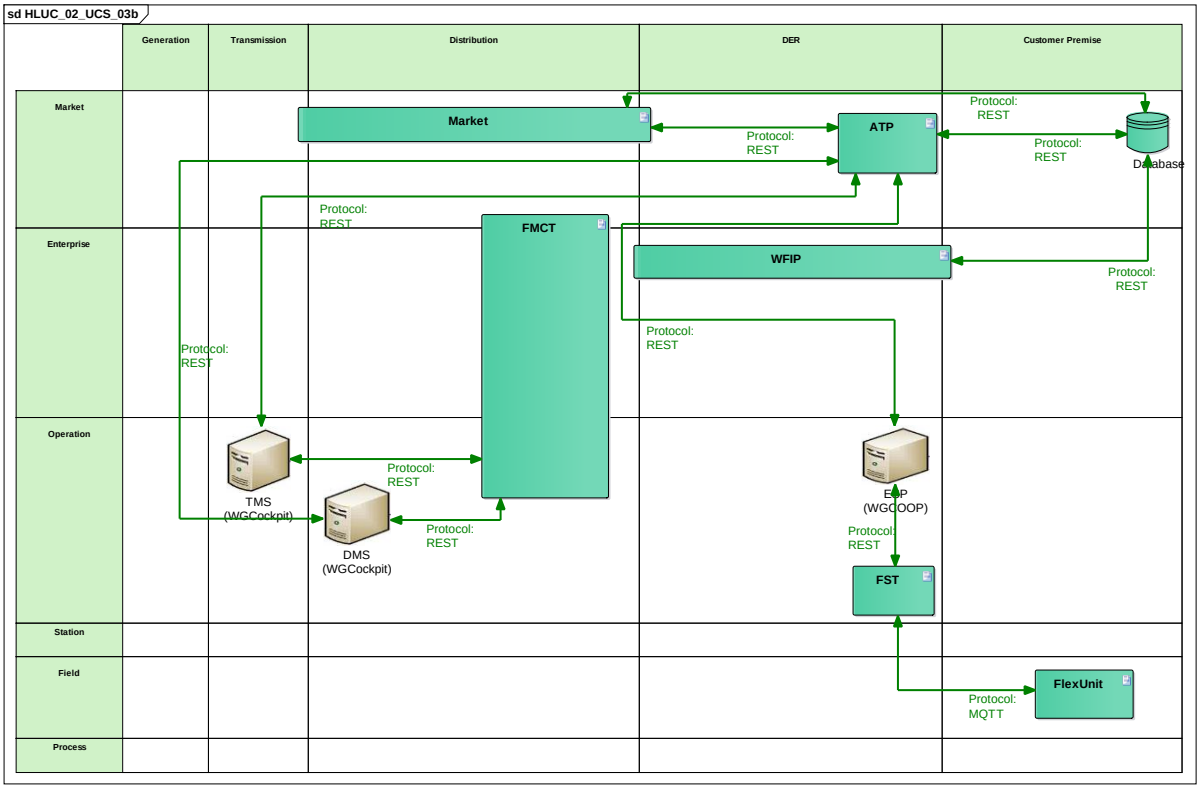

Figure 32 - HLUC_02_UCS_03 SGAM Communication Layer

5.3.3.4 SGAM Information Layer

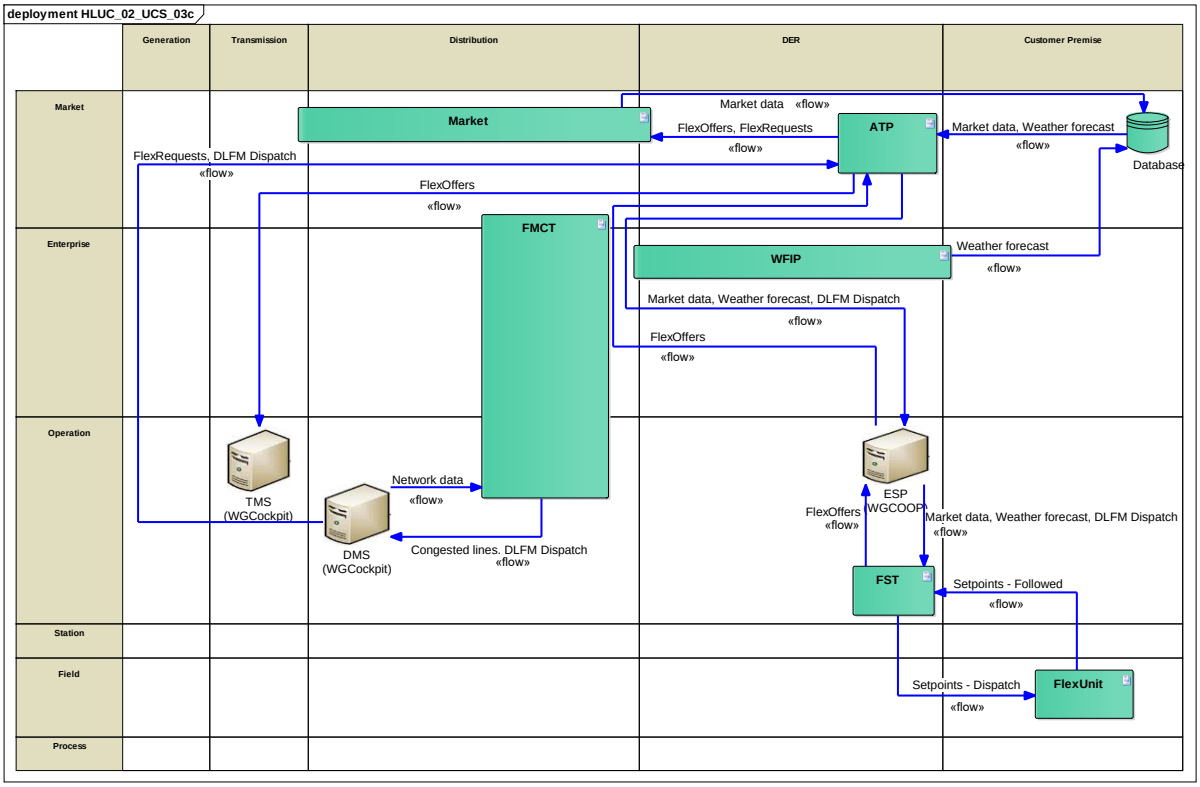

Figure 33 - HLUC_02_UCS_03 SGAM Information Layer

5.4 HLUC_03 Description: FLEXGRID ATP offers advanced flexibility demand management services to system operators

The purpose of HLUC_03 is to develop and test various ways of simultaneous use of FLEXGRID ATP together with existing market-based mechanisms. It should be beneficial for TSOs and DSOs to decide whether it is more cost-effective and efficient to purchase flexibility services on flexibility markets via the FLEXGRID ATP or on the existing markets. HLUC_03 will also compare future investment scenarios for DSOs. There are two main alternatives for DSOs: i) continue with their Business-As-Usual (BAU) scenario in which they invest on new CAPEX in order to reinforce their network, or ii) purchase flexibility from ESPs and aggregators[52]. The second alternative is FLEXGRID project's proposal and can be further divided in two sub-scenarios, namely: i) either purchase flexibility via long-term and bi-lateral contracts with local flexibility providers or ii) purchase flexibility through a (DLFM) via auction-based mechanisms. In order to run the system in a safe and the most economical way, TSO/DSO cooperation is also crucial. Hence, several TSO-DSO coordination schemes will be studied and compared.

[52] DSOs are already today procuring flexibility from their customers, e.g. through interruptible tariffs. The main difference to the proposed DLFM is that the current schemes are not market based.

5.4. HLUC03_UCS_01: Coordinated voltage/reactive power control either by aggregating flexibility from multiple FlexAssets or through a market-based mechanism

5.4.1.1 Use Case Scenario summary and context

The scope of this UCS is to utilize reactive power provision capabilities of renewable energy resources (RES) and distributed energy resources (DER) as well as emerging technologies in the distribution grids to increase the hosting capacity and to improve voltage profiles both in transmission and distribution grids. This UCS utilizes untapped reactive power provision capability of RES and DER resources in the distribution grid (DSO level) in addition to conventional resources controlled by TSOs, taking into consideration new control schemes utilizing all possible flexibilities, which could be introduced by distributed generators, DER and emerging technologies. The proposal is that the future DSO, who tries to solve over/under-voltage issues has two basic alternative solutions (i.e. advanced control mechanisms vs. flexibility market-based mechanism). In case it is cheaper and more effective, the DSO will use a flexibility market-based mechanism. DSO will have the intelligence to decide dynamically on what control policy to follow based on the type of event/problem. Since reactive power must be transported to the TSO grid, intensive cooperation between DSO and TSO demanding simultaneous coordination is necessary. Thus, the optimal planning and operation of DG and RES located in distribution grids in order to contribute to local and regional reactive power management and reactive power provision to the transmission grid must be enabled.

5.4.1.2 SGAM Component Layer

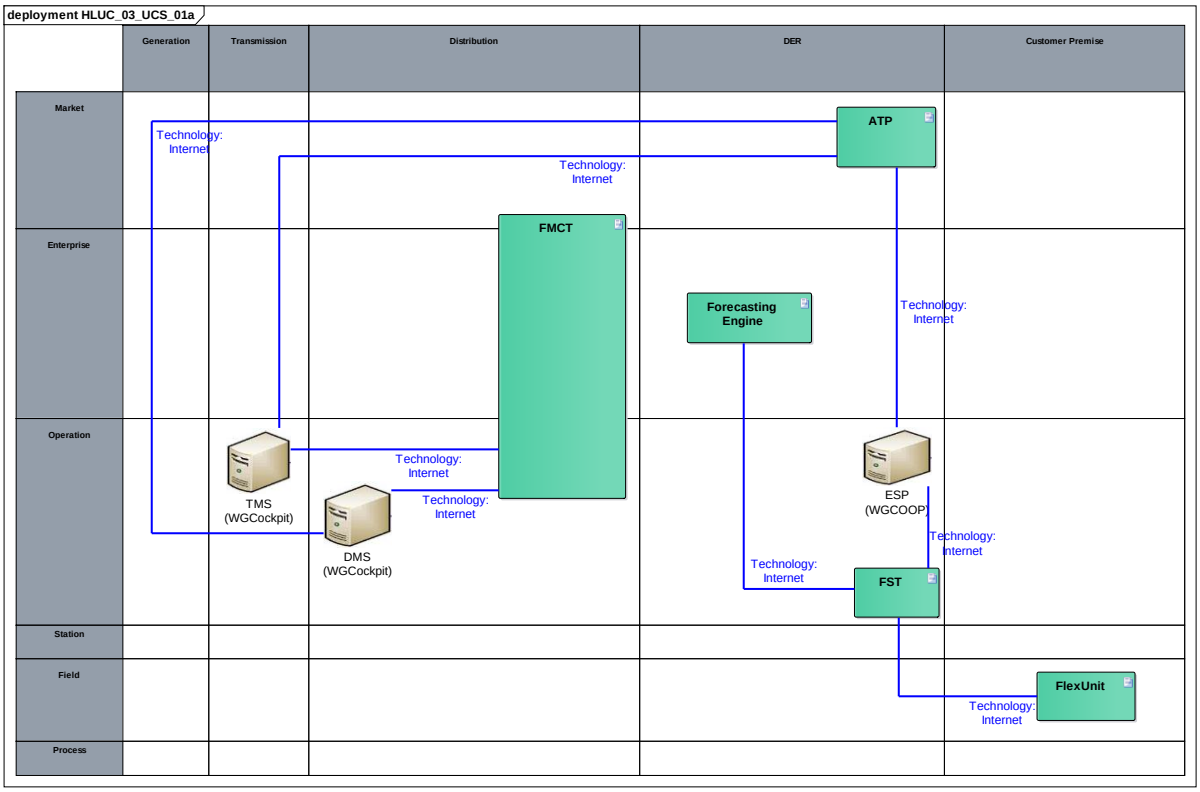

Figure 34 - HLUC_03_UCS_01 SGAM Component Layer

5.4.1.3 SGAM Communication Layer

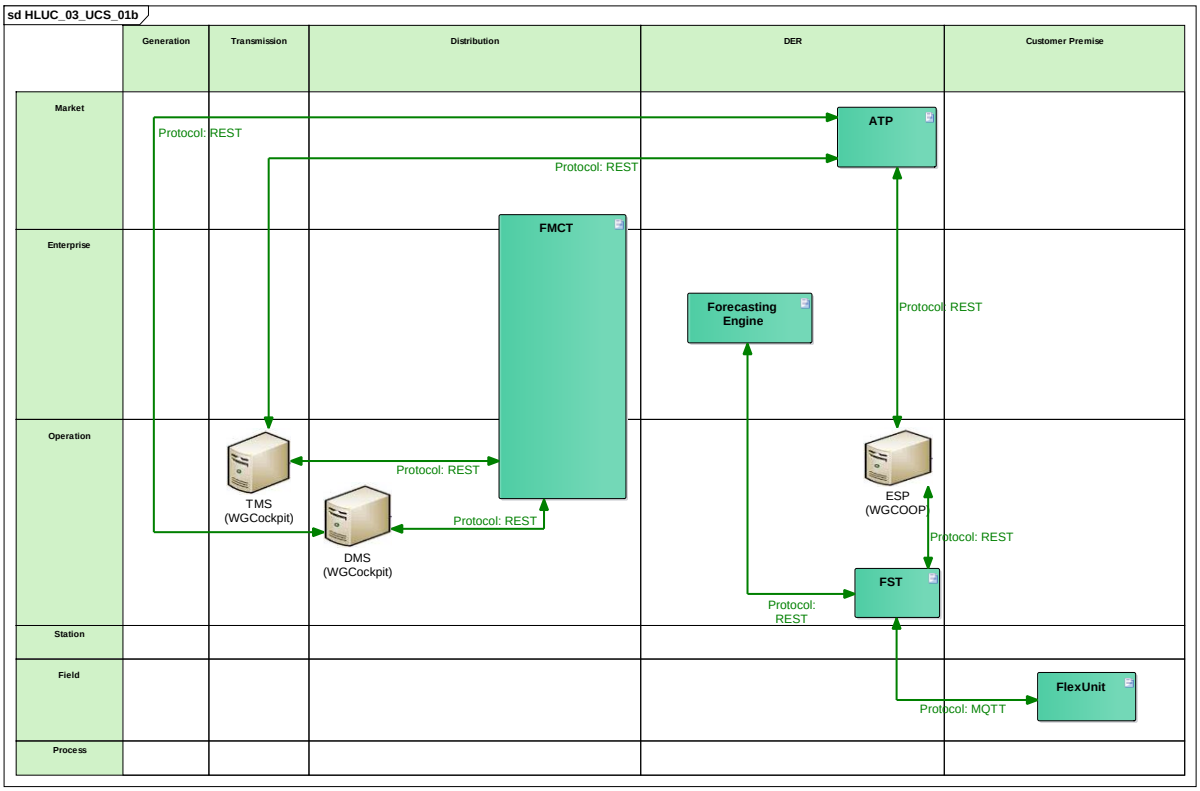

Figure 35 - HLUC_03_UCS_01 SGAM Communication Layer

5.4.1.4 SGAM Information Layer

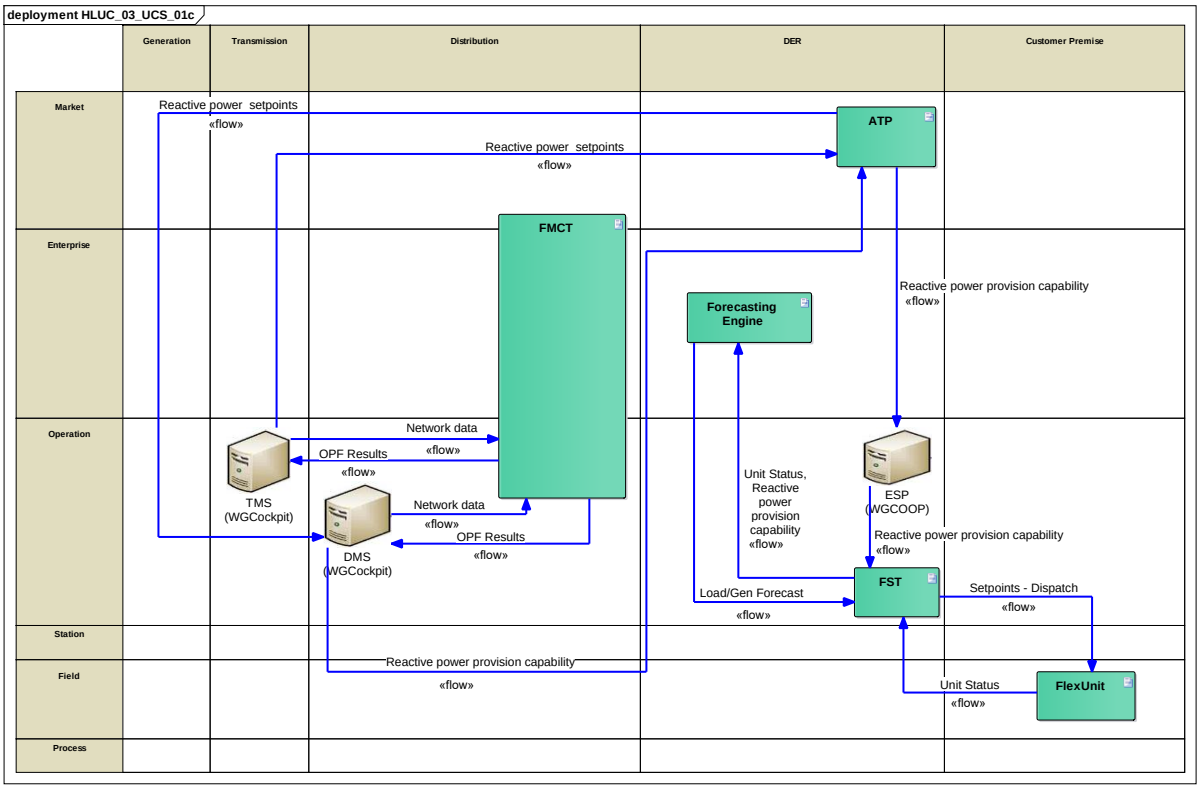

Figure 36 - HLUC_03_UCS_01 SGAM Information Layer

5.4.2 HLUC_03_UCS_02: TSO-DSO collaboration for coordinated management of aggregated FlexAssets and interaction between networks' and flexibility markets' operation

5.4.2.1 Use Case Scenario summary and context

Customers and distributed third-party energy resources that have the ability to change their consumption or generation (including energy storage systems/units) for short time could be aggregated and their flexibility could be offered as ancillary service to TSO or to be used for DSO grid purposes. For commercial purposes, the Flexibility Market Operator (FMO) offers ancillary services to the TSO (frequency control services and balancing services). The FMO pools flexibility of DER with the commercial FlexAssets' management system operated by the FlexSupplier stakeholder. On the other hand, the same DERs' flexibility could be used by the DSO for non-frequency DSO needs (i.e. solving local congestions and voltage problems).
The main innovation is to supply reliable and efficient (technically and competitively priced) flexibility to TSO or DSO from geographically distributed third-party energy resources through the use of FLEXGRID ATP (operated by a novel FMO entity)[53].

5.4.2.2 SGAM Component Layer

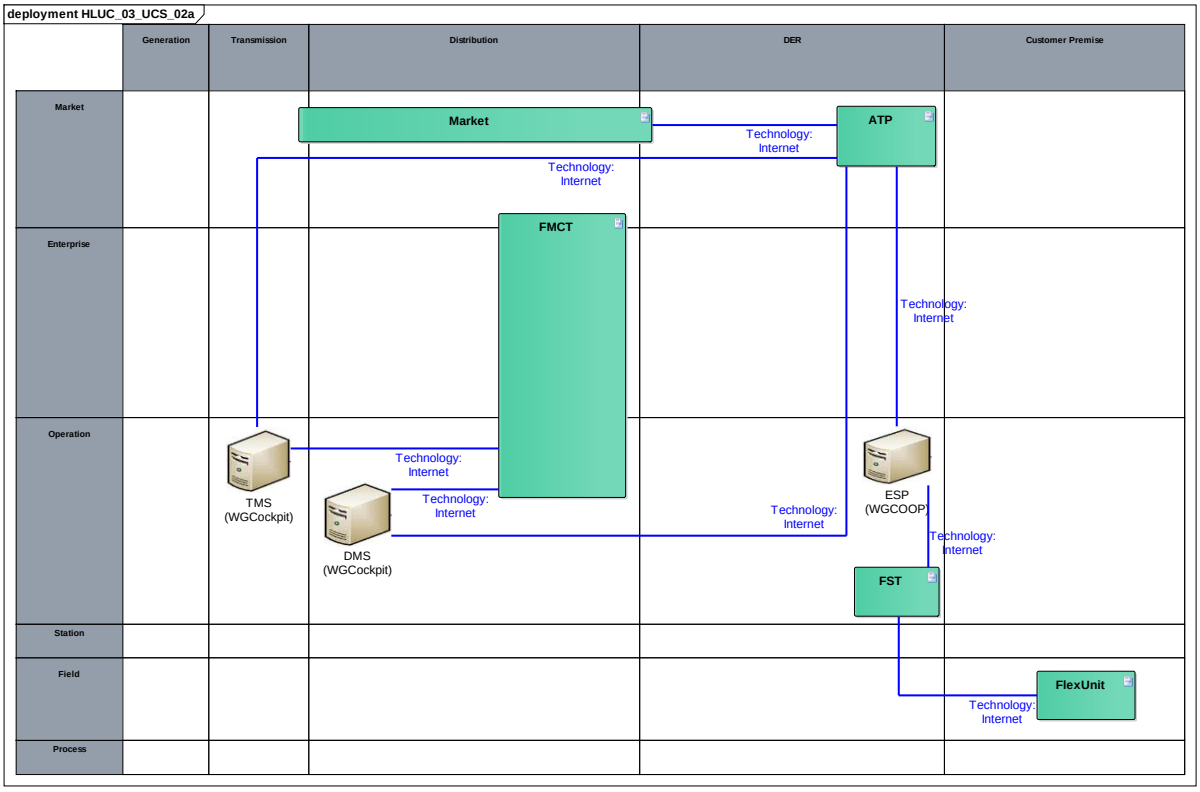

Figure 37 - HLUC_03_UCS_02 SGAM Component Layer

[53] As part of FLEXGRID research, there will be a couple of UCS like thisUCS 3.2, where the FMO is understood as an active participant in the market playing the role an aggregator facilitating thus an efficient TSO-DSO collaboration.

5.4.2.3 SGAM Communication Layer

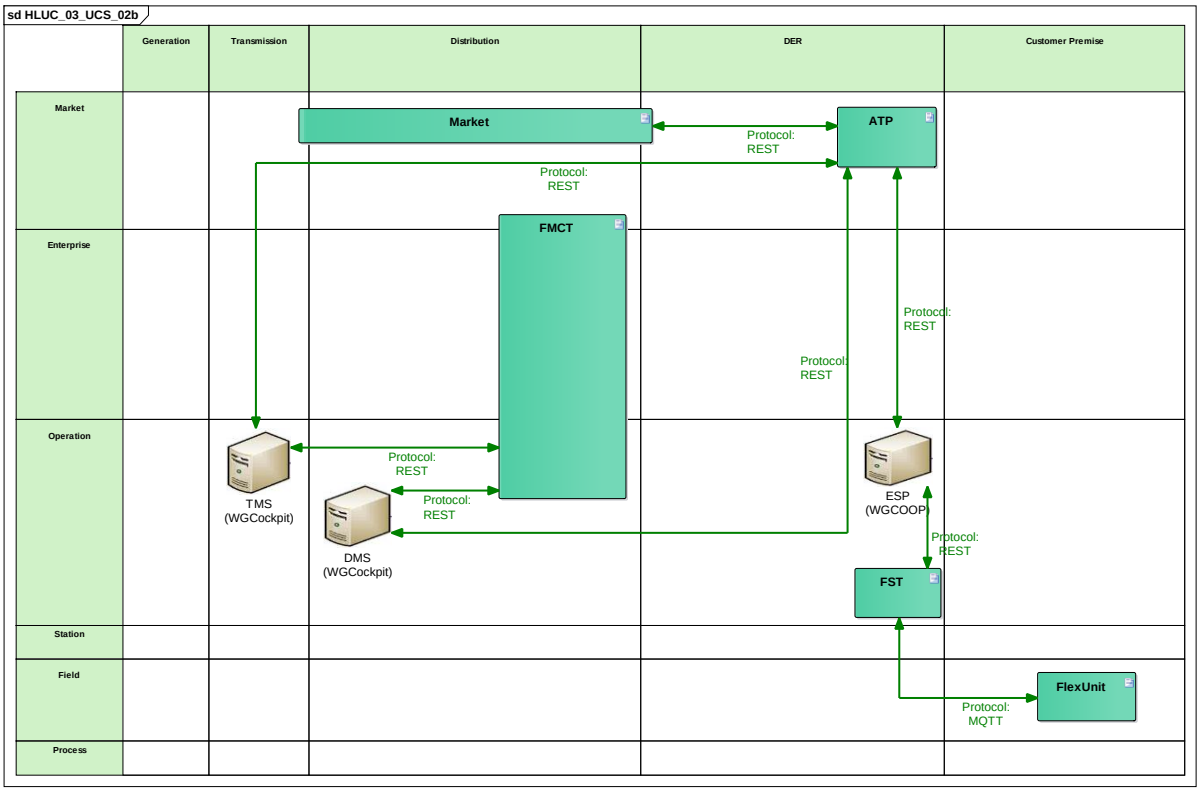

Figure 38 - HLUC_03_UCS_02 SGAM Communication Layer

5.4.2.4 SGAM Information Layer

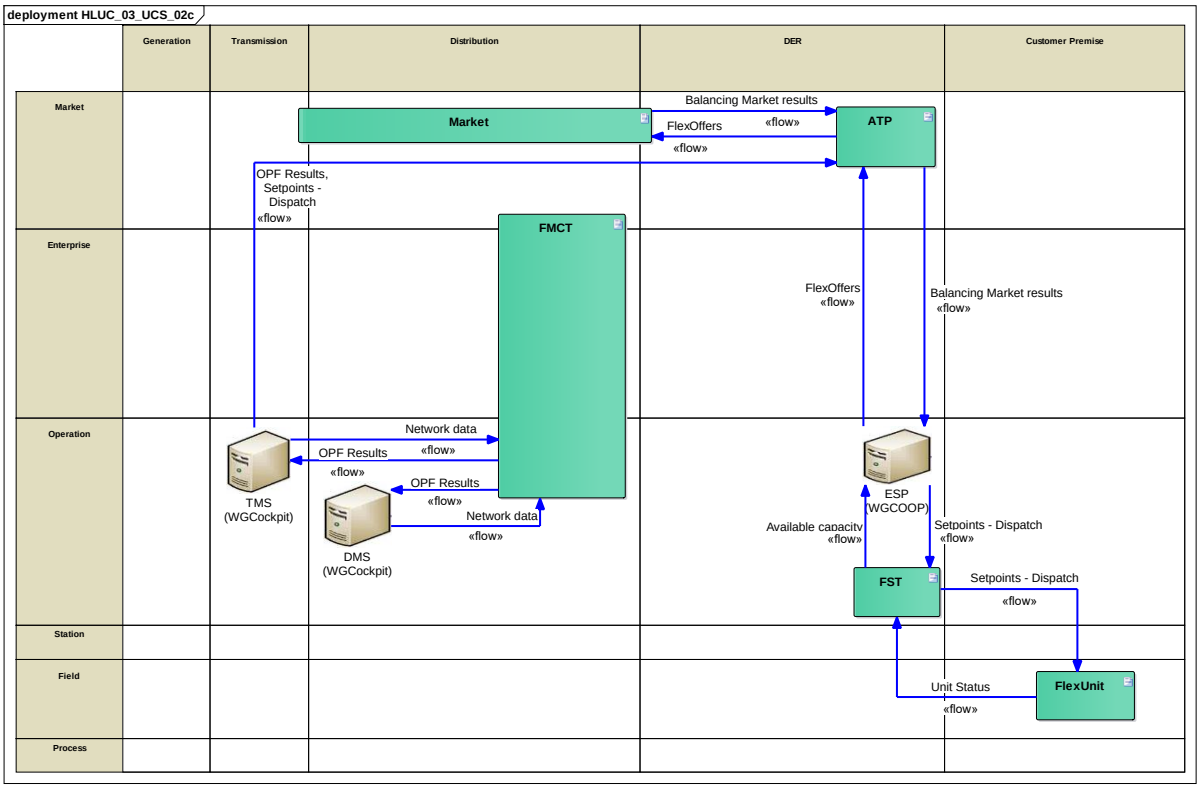

Figure 39 - HLUC_03_UCS_02 SGAM Information Layer

5.5 HLUC_04 Description: FLEXGRID ATP offers automated flexibility aggregation management services to ESPs/aggregators

This HLUC focuses on the operation of the automated flexibility aggregation, which is modeled as a novel ad-hoc energy market development and management as a service to be offered to independent aggregators and/or ESPs. It deals with the B2C interaction between an ESP/aggregator entity and its business portfolio, which comprises of a large amount of end energy prosumers together with their FlexUnits (i.e. DSM, RES and storage flexibility assets). In the context of FLEXGRID project, a S/W toolkit will be developed, which is called Automated Flexibility Aggregation Toolkit (AFAT) and its requirements are described in D2.1, section 6.2.3. The AFAT will integrate several retail flexibility market pricing schemes, flexibility aggregation models and respective algorithms. There will also be an API in order for the AFAT to interact dynamically with the core FLEXGRID ATP. More specifically, the FLEXGRID ATP will send all available FlexRequests being made by TSO/DSO/BRP. Then, the AFAT will run a specific automated flexibility aggregation algorithm and the result will be an optimal FlexOffer that will be sent back to the ATP. In exercising this business approach, aggregators/ESPs can utilise advanced forecasting services to predict market prices and net load profiles of available end users' assets to facilitate optimal use of resource availability for maximising profits for all participants in the portfolio.

5.5.1 HLUC_04_UCS_01: ESP/aggregator efficiently responds to FlexRequests made by TSO/DSO/BRP by optimally orchestrating its aggregated flexibility portfolio of end energy prosumers

5.5.1.1 Use Case Scenario summary and context

In this scenario, the interaction between the ESP/aggregator and the end- user will be studied. More specifically, there are FlexContracts between end users and ESP/aggregator, where the end users state their preferences. Then, the ESP/aggregator considers the available flexibility and selects the most appropriate mix of end users to satisfy the FlexRequest by choosing the most profitable solution, but at the same time considering the technical constraints (i.e. end-user's utility function and welfare). Optimisation in this case is carried out centrally in contrast with the UCS 4.2 and 4.3 described below, where decentralized modelling and approaches are assumed[54]. Interaction between ESP/aggregator and end users will be easy and effective, so that they can participate in future dynamic energy markets thereby increasing their profits.

[54] In the centralized modelling, all end users provide their flexibility preferences and utility functions to the aggregator and the latter gathers all data and solves the problem centrally. In decentralized models, there is an iterative process of message exchanges, in which end users exchange data with the aggregator until this process converges to a global optimum (cf. UCS 4.2 and 4.3 below).

5.5.1.2 SGAM Component Layer

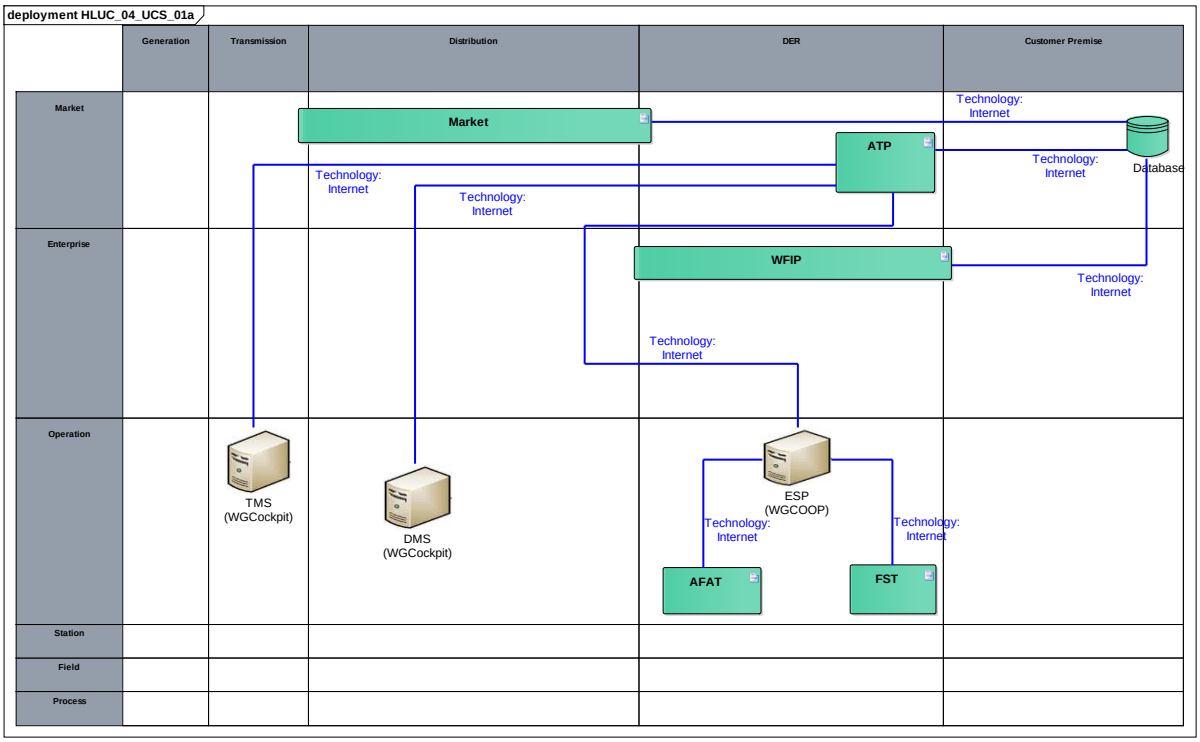

Figure 40 - HLUC_04_UCS_01 SGAM Component Layer

5.5.1.3 SGAM Communication Layer

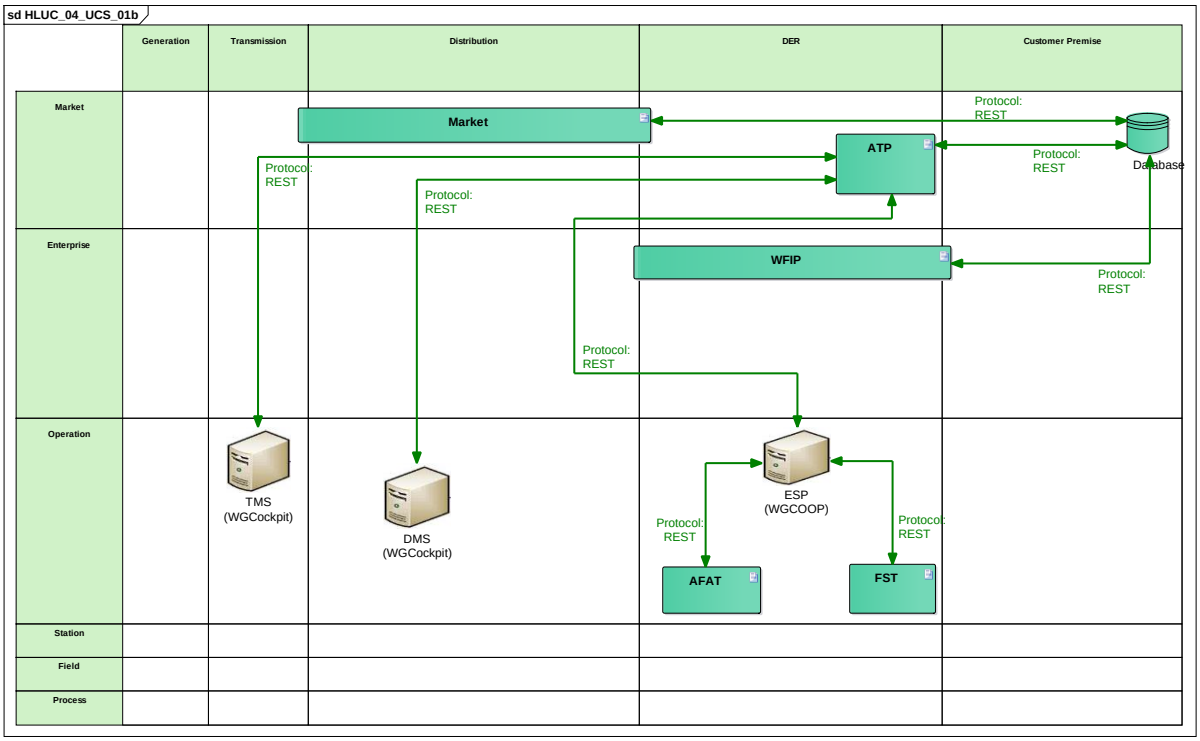

Figure 41 - HLUC_04_UCS_01 SGAM Communication Layer

5.5.1.4 *SGAM Information Layer*

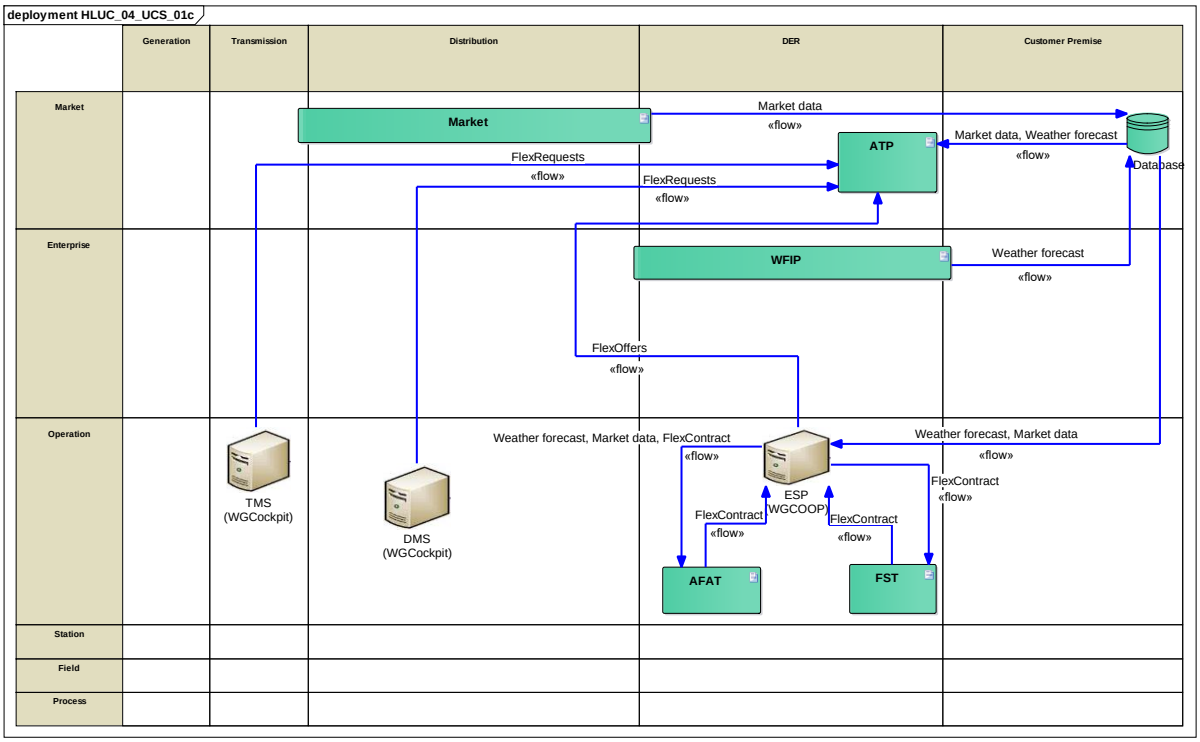

Figure 42 - HLUC_04_UCS_01 SGAM Information Layer

5.5.2 HLUC_04_UCS_02: An aggregator operates an ad-hoc flexibility market with its end energy prosumers by employing advanced pricing models and auction-based mechanisms

5.5.2.1 *Use Case Scenario summary and context*

The aggregation of small-scale distributed flexibility assets (end user electric appliances with controllable loads, EVs, batteries, etc.) requires the development of a retail flexibility market through which ESP/aggregator trades dynamically with end users the value of the flexibility assets (FlexAssets) that the latter dispose. The development of dynamic pricing schemes and auction-based processes has several requirements, because these systems have to be: real time, efficient, strategy proof, competitive, scalable, fair and privacy protecting[55]. Moreover, the uncertainty in the constraints and preferences that the end user introduces is a critical research thread towards their development.

In this UCS, we consider the case of an aggregator that is responsible for coordinating the energy prosumption of end users within its portfolio, while facing costs and constraints on the aggregate energy prosumption. The aggregator takes on the task of satisfying the system constraints in the most efficient way (i.e. its objective is to maximize social welfare[56]). In FLEXGRID, we consider both day-ahead and near-real-time cases.

[55] In most FLEXGRID B2C flexibility market models, we assume that a S/W agent resides at every home/building energy management system and is able to communicate in real-time with a centralized S/W agent that resides at the aggregator's/retailer's side.

[56] In this case, social welfare KPI includes the profits of the aggregator, which acts as a profit-based company trying to find an optimal trade-off between maximizing its own profits and provide the best quality of services to its end users (i.e. minimize inconvenience of end users based on the latter's preferences).

5.5.2.2 SGAM Component Layer

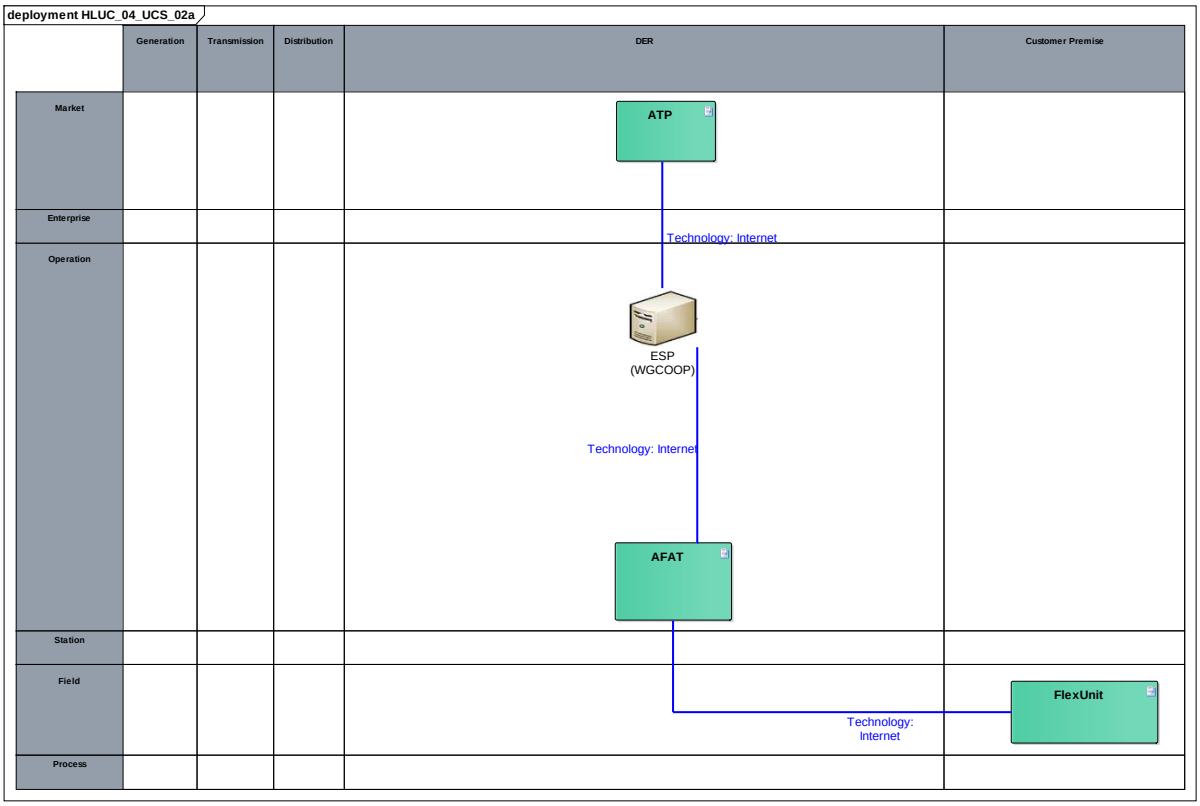

Figure 43 - HLUC_04_UCS_02 SGAM Component Layer

5.5.2.3 SGAM Communication Layer

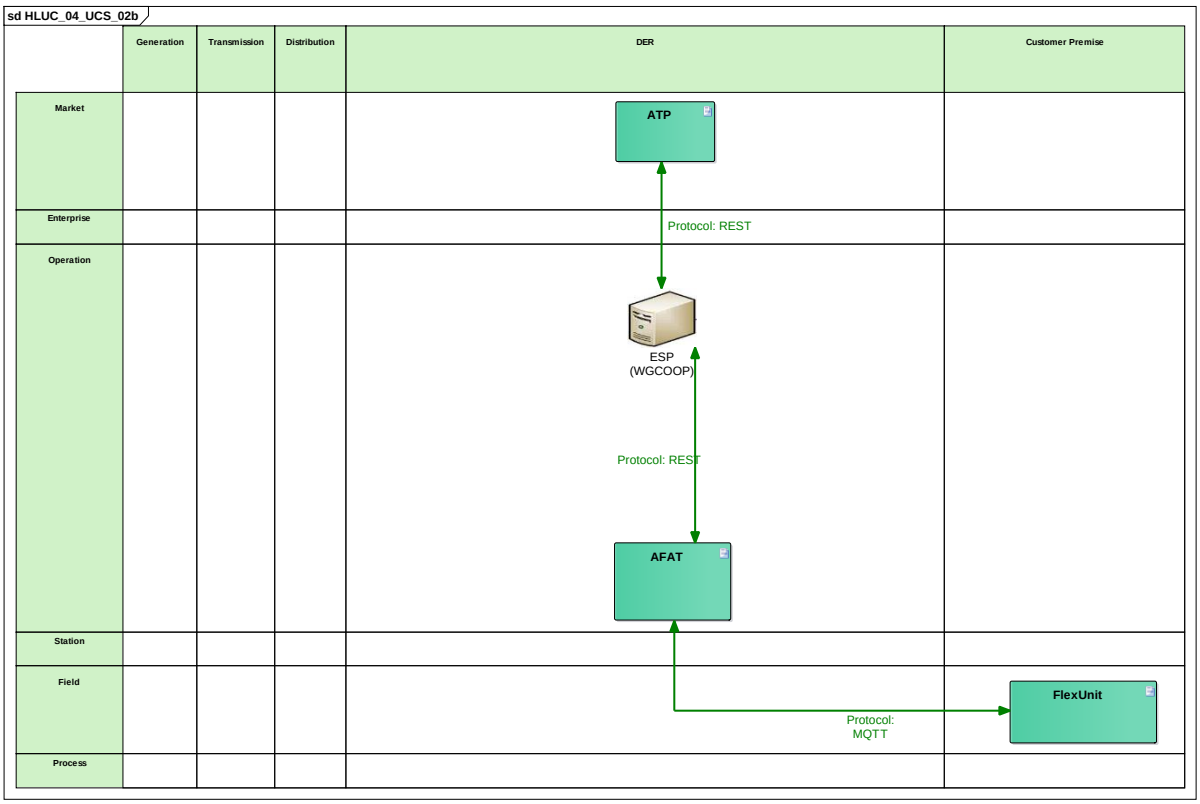

Figure 44 - HLUC_04_UCS_02 SGAM Communication Layer

5.5.2.4 *SGAM Information Layer*

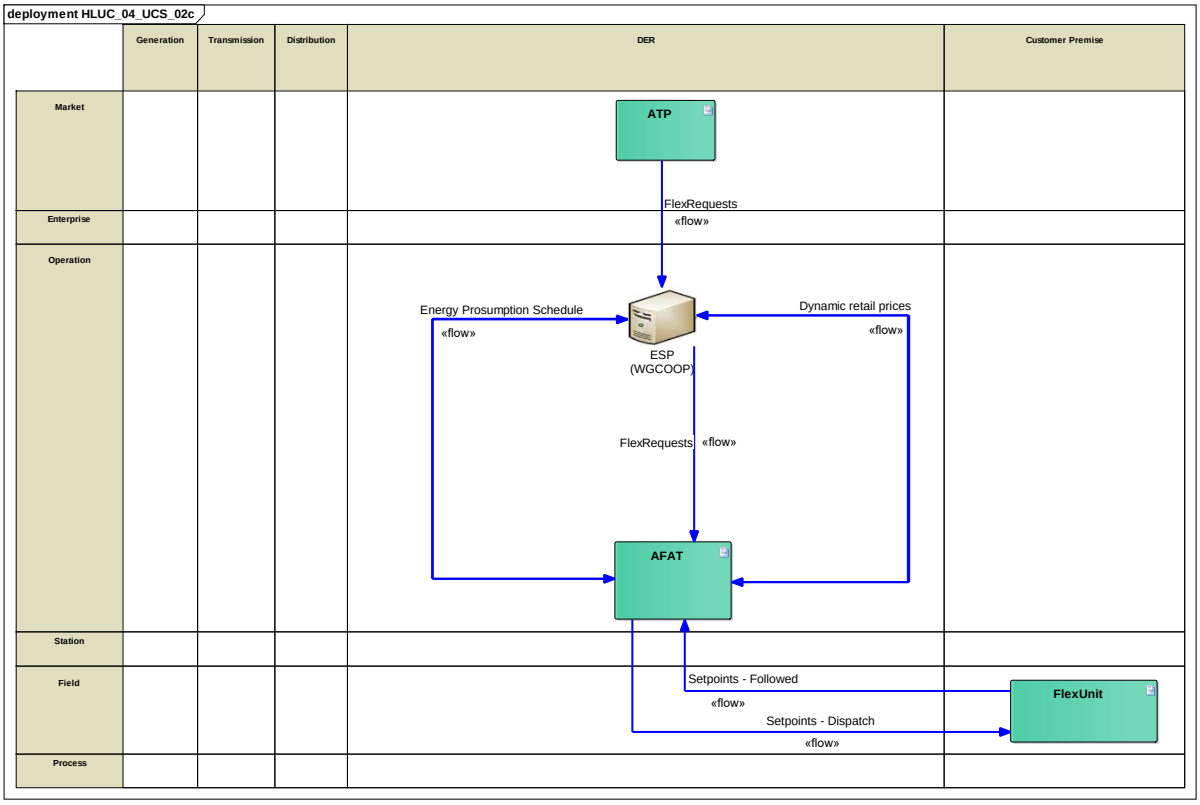

Figure 45 - HLUC_04_UCS_02 SGAM Information Layer

5.5.4 HLUC_04_UCS_04: ESP exploits FLEXGRID's advanced forecasting services to predict market prices and FlexAssets' state and curves in the future

5.5.4.1 *Use Case Scenario summary and context*

This scenario concerns forecasting services' provisioning for ESP and aggregator actors. These services will include forecasting the ESP's/aggregator's generation by aggregating its end-users' generation (both day ahead and intra-day – predominantly from PV systems), market price forecasting[57] will also be considered in order to facilitate the optimal FlexOffer process towards efficient ESP's/aggregator's participation in distribution level flexibility market and wholesale/balancing markets (i.e. transmission system level).

[57] For wholesale market price forecasting, there already exist a bunch of available tools on the market. More details on how FLEXGRID research will outperform existing solutions will be provide in subsequent D4.1 (due to Month 12).

5.5.4.2 SGAM Component Layer

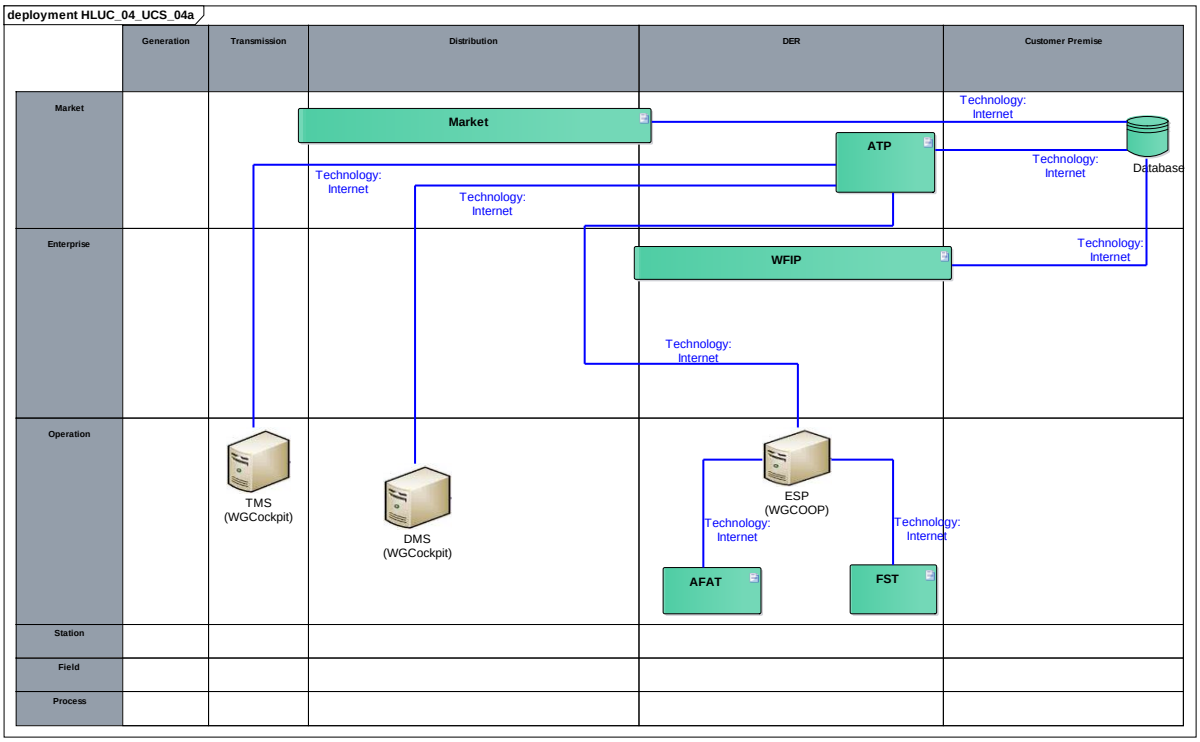

Figure 46 - HLUC_04_UCS_04 SGAM Component Layer

5.5.4.3 SGAM Communication Layer

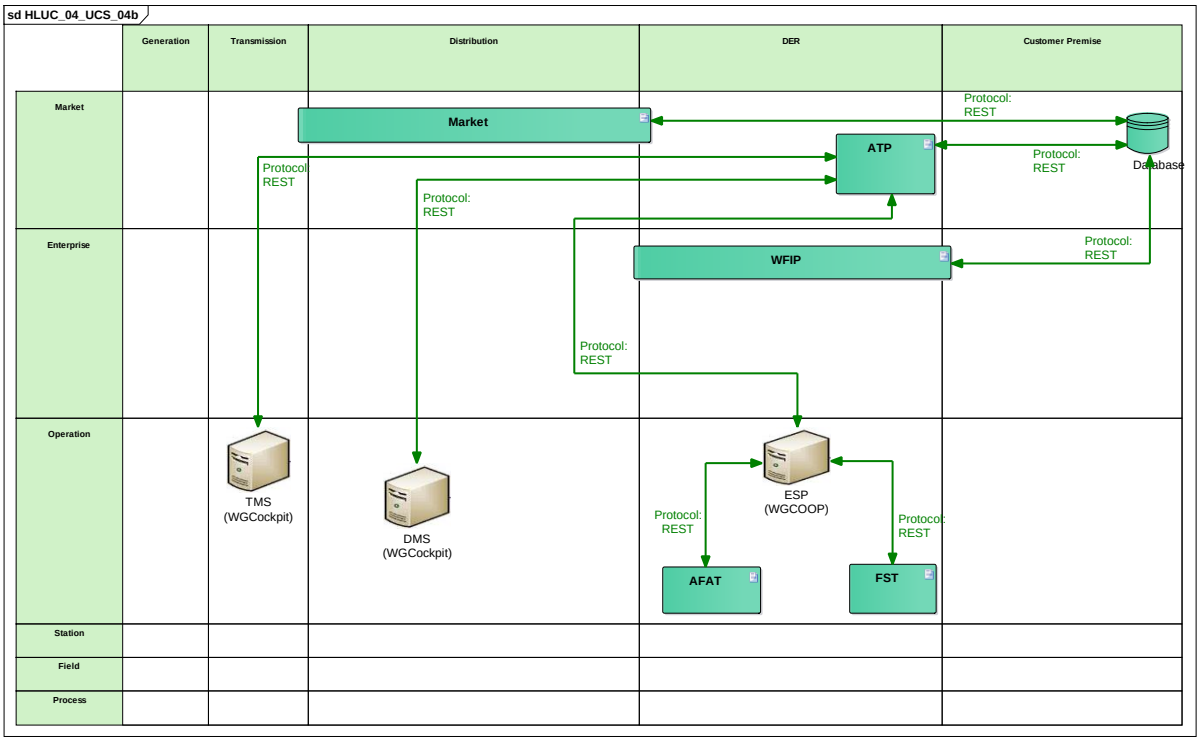

Figure 47 - HLUC_04_UCS_04 SGAM Communication Layer

5.5.4.4 SGAM Information Layer

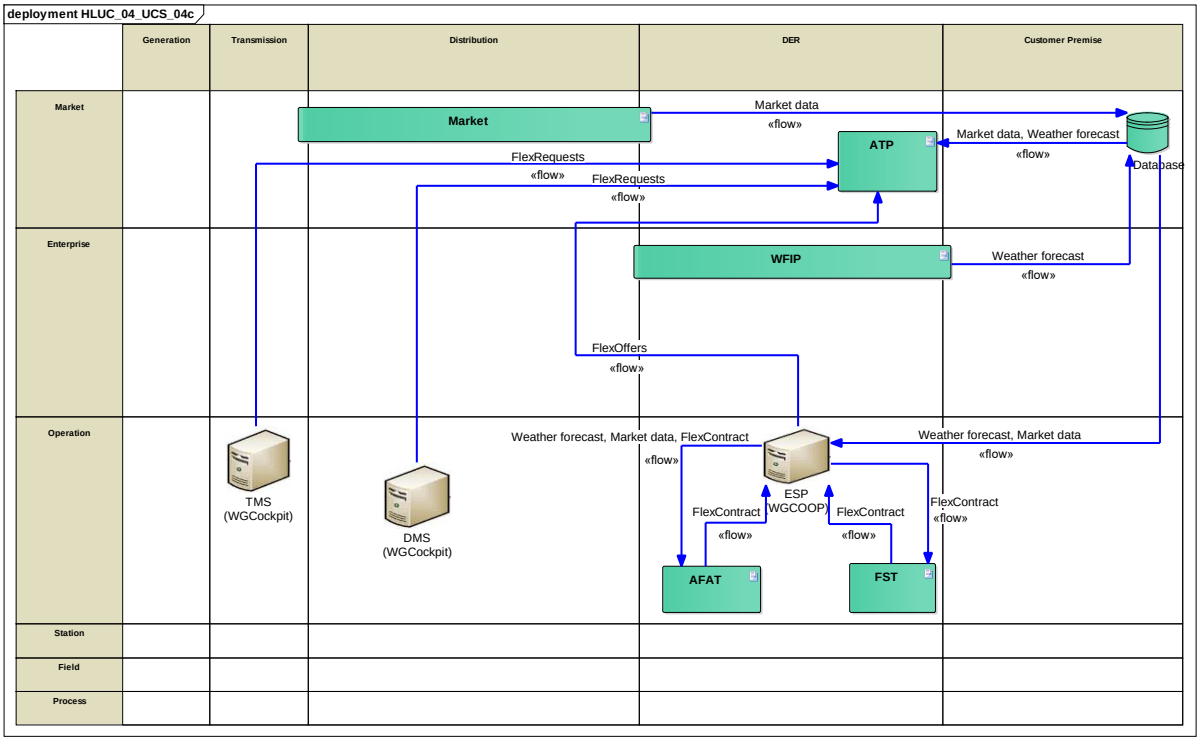

Figure 48 - HLUC_04_UCS_04 SGAM Information Layer

6. Conclusions

Conclusively, during the next months, FLEXGRID consortium will elaborate on the current work presented in this deliverable, towards implementing the final FLEXGRID S/W platform and starting the respective research for each module functionality on the Flexibility Markets proposed. The work schedule plan is the following:

- All academic partners have already started the R&I work in WPs 3-5. Each academic partner works on its own research threads, as these have been explicitly and clearly defined in the description of the Use Cases Scenarios (D2.1, Section 5). Collaboration with specific industrial partners (who lead respective High Level Use Cases - HLUCs) is also taking place.
- Until M12, the consortium shall develop the design documents including technical specifications for all FLEXGRID S/W modules, with research objectives and challenges addressed by this project activities.
- From M13 onwards, all partners will start collaborating towards integrating each individual subsystem into the single modular-by-design FLEXGRID S/W platform, taking into consideration the design decisions addressed in this deliverable.
- Information scheme will be developed from Appendix elements, in order to design the final Data Model for the FLEXGRID S/W platform, as well as the rest of WP6 work from M13 onwards.

The figure below shows the current project's timeline schedule. Milestones #2 (D8.1) and #3 (D2.2) have been achieved, while there is one more milestone (#4) to be achieved before the end of Year 1 (i.e. by 30/9/2020).

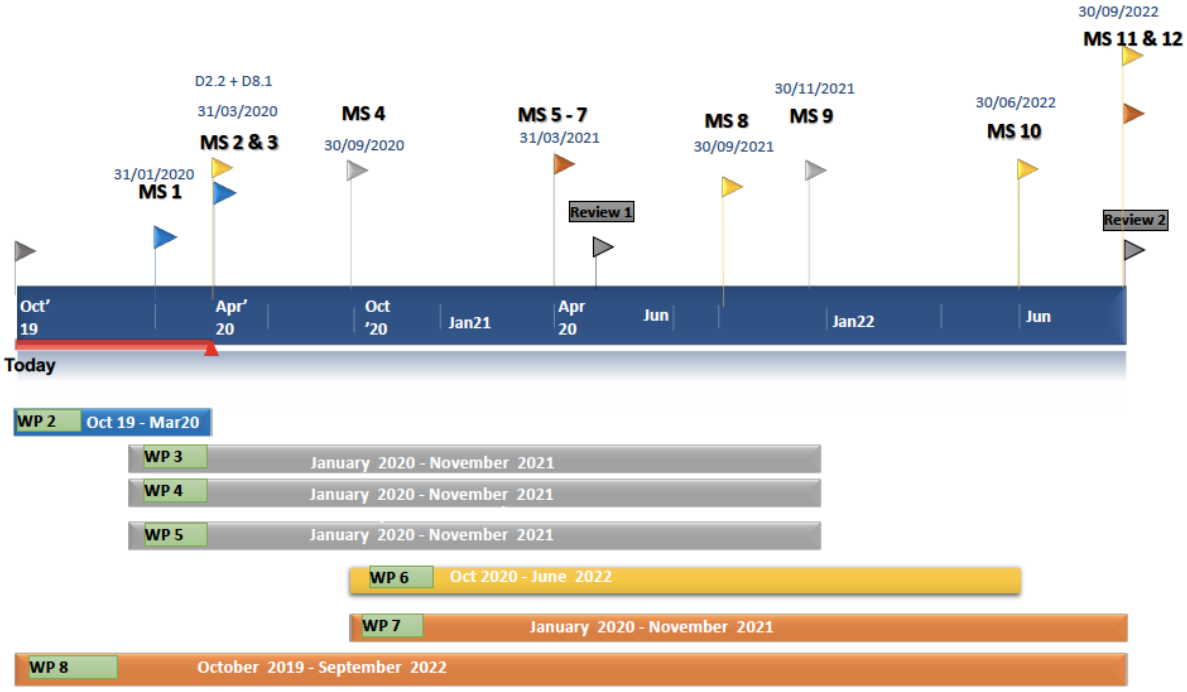

Figure 49: Current FLEXGRID project's timeline schedule (MS 2 & 3 have been accomplished)

7. ANNEX: Exchanged Data between FLEXGRID modules

The Data Model is crucial for the S/W development. This appendix will define the information managed by each module, categorized as input or output, to be translated into the formal Data Model that will take place in WP6. The data shown in this appendix and their components will be uniformized into a coherent information scheme, the one that module's APIs shall deal with for data interoperability.

7.1 AFAT information

Input data for the "Retail Pricing Algorithm" module:

- Set of end energy prosumer ids
- Real historical Energy Consumption Curves (ECCs) and RES Production Curves (RESPCs) of all end energy prosumers
- Start and end time of simulation
- Time interval (e.g. 15-min, 1-hour, daily, weekly, monthly)
- Type of Energy Consumption and RES Production Curves (ECC & RESPC)
- Energy cost function parameters for the aggregator/ESP to purchase energy from the wholesale market
- Retailer's profit margin parameter
- Flexibility factor or else utility functions of each end energy prosumer
- Parameter that models the type of FlexContracts to be compared
- Types of FlexContracts to be compared

Output data for the "Retail Pricing Algorithm" module:

- Energy cost
- Aggregated Users' Welfare (AUW)
- Retailer company's profit
- Total energy prosumption
- Total bills paid by end users
- Social Welfare (SW) of the B2C flexibility market

Input data for the "Flexibility Aggregation Algorithm" module:

- Set of end energy prosumer ids
- Start and end times of DR event (i.e. scheduling horizon)
- Time interval (e.g. 5-min, 15-min, 1-hour)
- Real historical Energy Consumption Curves (ECCs) and RES Production Curves (RESPCs) of all end users
- Flexibility factor or else utility functions of each end energy prosumer
- Forecasted ECCs, RESPCs and batteries' State-of-Charge (SoC) from AFAT forecasting engine's results
- Forecasted market prices (according to the respective market participation case) from AFAT forecasting engine's results

- Aggregator's technical constraints imposed by the DR event specifics (e.g. total power to be decreased/increased should not be higher than a threshold for one timeslot or a given time horizon)
- Aggregator's financial constraints (e.g. profits should be at least 10%)

Output data for the "Flexibility Aggregation Algorithm" module:

- Optimal energy prosumption schedules for each end energy prosumer (i.e. setpoints for a given time horizon)
- Aggregated Users' Welfare (AUW) and per end user
- Social Welfare (SW) of the B2C flexibility market
- Aggregator's profit

7.2 FST information

Input data for the "Optimal bidding algorithm" module:

- Distribution network technical data
    - Topology
    - Admittance
    - Capacity (MVA)
    - Tap ratios
    - Shunt capacities
    - Loads
    - Voltage constraints
- Generation
    - Locations (nodes)
    - Min./max. power
    - Costs
    - Type (PV, solar, …)
    - Dispatched generation for the respective time period
    - RES production curves
- Market data (intra-day, day-ahead, balancing markets)
    - Historical prices
    - Historical volumes
    - Forecasted market prices
- Energy consumption data
    - Historical energy consumption data
    - Dispatched energy consumption for the respective time period
    - Forecasted energy consumption
- Storage
    - Location (nodes)
    - Capacity per asset (MW)
- AC-OPF market clearing
- FlexAssets' characteristics
    - Technical characteristics:
        - Ramp-up/down time

        - Flexibility factors (FF) (for instance household might have higher FF during the night for EV charging)

Output data for the "Optimal bidding algorithm" module:

- FlexOffers differentiated per market (e.g. day-ahead, intraday, balancing)

Input data for the "FlexAsset sizing/siting algorithm" module:

- Existing FlexAsset situation
    - Locations (nodes) of assets
    - Capacity of each asset (MW)
    - Technical characteristics:
        - Ramp-up/down time
        - Flexibility factors (FF) (for instance household might have higher FF during the night)
- Other stakeholders
    - Current assets
        - Locations (nodes)
        - Capacity (MW)
    - Future investment plans
        - Stochastic scenarios
- Distribution network technical data
    - Topology
    - Capacity (MVA)
    - Admittance
    - Voltage constraints
    - Tap setting
    - Loads
    - Shunt capacities
- AC-OPF
    - Congested lines
- FlexRequests
    - Active power requested
    - Reactive power requested
    - Location of the nodes which are generating the requests
    - Historical data
    - Biding price for the active power
    - Biding price for the reactive power

Output data for the "FlexAsset sizing/siting algorithm" module:

- Optimal size of the FlexAssets
- Optimal siting of the FlexAssets

Input data for the "Optimal scheduling algorithm" module:

- Distribution network technical data
    - Topology
    - Admittance

    - Capacity (MVA)
    - Tap ratios
    - Shunt capacities
    - Loads
    - Voltage constraints
- Market results (AC-OPF)
    - Market clearing
    - Congested lines
    - Generation dispatch
- Generation
    - Location (nodes)
    - Type (PV, solar, …)
        - Characteristics
    - Capacity (MW)
    - Dispatched generation for the respective period
    - Forecasted RES production
- Consumption data
    - Dispatched consumption for the respective period
    - Forecasted consumption

Output data for the "Optimal scheduling algorithm" module:

- Optimal schedule for the respective (Flex)units

7.3 FMCT information

Input data for the AC-OPF flexibility market clearing algorithm:

- Network description:
    - Lines data:
        - Identification (name)
        - From node
        - To node
        - Admittance
        - Capacity (MVA)
    - Nodes data:
        - Tap ratios
        - Shunt capacities
        - Connected generators
        - Connected loads
        - Connected flexibility sources
        - Connected lines
        - Voltage limits
- Generation, with schedules from earlier stages markets. For each generator:
    - Identification (name)
    - Location in the network (node)
    - Production dispatched for the given time period (MW)

- Consumption, with schedules from earlier stages markets. For each load:
    - Identification (name)
    - Location in the network (node)
    - Consumption dispatched for the given time period (MW)
- FlexRequests:
    - Active power requested
    - Reactive power requested
    - Node
    - Bidding price for active power
    - Bidding price for reactive power
- FlexOffers:
    - Active power offered
    - Reactive power offered
    - Node
    - Bidding price for active power
    - Bidding price for reactive power
    - Block offer scheme

Output data for the AC-OPF flexibility market clearing algorithm:

- Congested lines identified:
    - Line identification
- Voltage deviation identified:
    - Node
- Market clearing for congestions:
    - Flexibility schedule (dispatch)
    - Prices for congestion management
- Market clearing for voltage deviations:
    - Flexibility schedule (dispatch)
    - Prices for voltage management

7.4 Real-life datasets

There is an extensive description on these datasets in Section 3 of FLEXGRID D8.1 delivered in Month 6 as part of the project's Data Management Plan (DMP).