This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957781.

active communities & energy prosumers for the energy transition

GRANT AGREEMENT NUMBER 957781

WP2 – Foundations

D2.3 – ACCEPT system architecture description v1

Responsible organization	Hypertech
Contributing organization(s)	CIRCE, QUE, CERTH, Witside, Mytilineos, EDBR, MIWenergia, LaSolar, AEM
Due date of Deliverable	31/10/2021
Actual date of submission	28/10/2021
Type of deliverable	Report
Dissemination level	Public

Disclaimer: ACCEPT is a project co-funded by the European Commission under the Horizon 2020 - Call: H2020-SC3-2018-2020 EC-3 Consumer engagement and demand response Programme under Grant Agreement No. 957781. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Communities. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use, which may be made, of the information contained therein.

Authors

Name	Organization	e-mail	Role
Ismini Dimitriadou	Hypertech	i.dimitriadou@hypertech.gr	Leading author
Dimosthenis Tsagkrasoulis	Hypertech	d.tsagkrasoulis@hypertech.gr	Leading author
Ioannis Koskinas	CERTH	jkosk@iti.gr	Contributing Author
Maria Diamantaki	CERTH	mardiama@iti.gr	Contributing Author
Paschalis Gkaidatzis	CERTH	pgkaidat@iti.gr	Contributing Author
Dimos Ioannidis	CERTH	djoannid@iti.gr	Contributing Author
Panagiotis Andriopoulos	QUE	panos@que-tech.com	Contributing Author
Panagiotis Moraitis	QUE	p.moraitis@que-tech.com	Contributing Author
Stelios Genouzos	Witside	stelios.genouzos@witside.com	Contributing Author
Aitor Alcrudo Sangros	CIRCE	aalcrudo@fcirce.es	Contributing Author

Reviewers

Name	Organization	e-mail
Aitor Alcrudo Sangros	CIRCE	aalcrudo@fcirce.es
Daniele Farrace	AEM	dfarrace@aemsa.ch

Version history

Version	Date	Comments
0.0	23.08.2021	Table of Contents
0.1	15.09.2021	Contributions on functional specifications
0.2	23.09.2021	Updated Table of Contents, system architecture, SGAM, component template
0.3	26.09.2021	Filled in templates
0.4	01.10.2021	Filled in templates, populated sections, 1st consolidated version
0.5	05.10.2021	Integration comments, updates
0.58	07.10.2021	Intermediate version after additional inputs from QUE, CERTH
0.6	08.10.2021	2nd consolidated version after additional inputs from CIRCE, QUE, CERTH and Witside
0.7	15.10.2021	Integration of partner contributions and updates
0.8	18.10.2021	List of authors, content updates
0.82	21.10.2021	Version for peer review
1.0	29.10.2021	Final version ready for submission

Project Consortium

Austria
- European Center for Renewable Energy Güssing GmbH

Cyprus
- Witside International Markets LTD

Denmark
- Geco Global APS

Greece
- **Hypertech** (Project Coordinator)
- Ethniko Kentro Erevnas Kai Technologikis Anaptyxis
- Mytilinaios Anonimi Etaireia
- QUE Technologies Kefalaiiouchiki Etaireia

Ireland
- University College Cork National University of Ireland, Cork

Italy
- Rina Consulting SPA

Netherlands
- Bedrijfsbureau Energie Samen BV
- Cooperatief Energie Dienstenbedrijfs Rivierenland BA

Spain
- My Energia Oner SL
- La Solar Energia Sociedad Cooperativa
- Fundacion Undacion Circe Centre De Investigacion de Recursos Y Consumos Energeticos
- Viesgo Distribucion Electrica SL

Switzerland
- Azienda Elettrica Di Massagno SA

1. Table of Contents

Glossary

Application Programming Interface	An Application Programming Interface is a connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software.
Communication Protocol	A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods.
Component Diagram	In Unified Modelling Language, a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.
Deployment Requirements	Deployment requirements describe the desired configuration of a software system.
Functional Specification	A functional specification in software development is a textual description that specifies the functions that a system or component must perform.
Non-functional Specifications	Non-functional Specifications define system attributes such as security, reliability, performance, maintainability, scalability, and usability
Payload	In computing and telecommunications, the payload is the part of transmitted data that is the actual intended message.
Software Architecture	Software architecture refers to the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.
Software Component	An individual software component is a software package, a web service, a web resource, or a Component that encapsulates a set of related functions (or data).

Table of Abbreviations and Acronyms

API	Application Programming Interface
ATs	Aggregator Tools
ASE	ACCEPT Solution Emulator
BAM	Building Asset Manager
BDT	Building Digital Twin
BI	Business Intelligence
BIML	Building Information Management Layer
CADR	Community Asset Portfolio Registry
CDT	Consumer Digital Twin
CiDT	Citizen Digital Twin
DAM	District Asset Management
DHW	Domestic Hot Water
DIML	District Information Management Layer
DR	Demand Response
D-SRI	Dynamic SRI Performance Rating
EC	European Commission
ECTs	Energy Community Tools
EFT	Energy Flexibility Transactions
ESCo	Energy Service Company
ETs	ESCo Tools
EU	European Union
FFC	Flexibility Forecast Collection
HVAC	Heating, Ventilation and Air-Conditioning
H-ECTs	Horizontal Energy Community Tools
IoT	Internet of Things
KPI	Key Performance Indicator
ODFM	On-Demand Flexibility Management
P2P	Peer-to-Peer
P2P-EP	P2P Exchange Platform
PMCD	Portfolio Monitoring and Control Dispatch
RTs	Retailer Tools
SCP	Smart Contract Platform
SGAM	Smart Grid Architecture Model

SLA	Service Level Agreement
SRI	Smart Readiness Indicator
UC	Use Case
UI	User Interface
WP	Work Package
WSN	Wireless Sensor Network

The following naming conventions have been adopted for the ACCEPT architecture:

- The District Asset Management component has been renamed to District Asset Manager. A new component, namely the District Information Management Layer (which is the equivalent to the BIML for district assets) has been introduced to cover the functionalities of the District Asset Data Connector.
- The tools of the Citizen Application covering all the functionalities described in the DoW are the Building Monitoring and Control Module, the Notification and Alerting System, the Collaboration Forum Module, the Statistical Analysis Module and the Optimal Energy Schedule and Recommendation module.
- The Horizontal Energy Community Tools are supported by the registries and repositories described in the DoW and their functionalities are included as specifications for the components VPP Manager, Demand Elasticity Estimator, Energy Community Flexibility Manager and the P2P Supply Shadow Administrator.

Executive summary

This report presents the work carried out in Task 2.3 of ACCEPT regarding the system architecture design process and results up to Month 10 of the project. In particular, the deliverable includes the high-level architecture model of the complete system, with the initial point being the design as presented in the Description of Work, and further refined and updated based on the work carried out in this period. This high-level architecture has also been mapped to the various SGAM layers to assist the future integration and development activities.

The main bulk of the document concentrates on detailing the specifications and characteristics of the various system's software components. A dedicated template was created and circulated to all relevant partners in order to provide the components' functional and non-functional specifications, highlight interdependencies with other components, and declare the input/output data requirements (Annex I – Component's Specifications and Requirements Template). Additionally, detailed component diagrams were created, and finally, the connection with Deliverable 2.1 (D2.1) was performed via a mapping of components to Use Cases that required relevant functionalities. Sequence diagrams were already created for D2.1 and are not presented here.

This is the first version of ACCEPT's system architecture. It will drive the development of the first iteration of the system and its constituent components, up to Month 16 of the project. Following that, a pre-validation stage will evaluate the system prototype, resulting to refinements, changes and updates that will be reflected on the Deliverable 2.4, which will present the final version of the architecture.

1. Introduction

1.1. Scope of the Deliverable

The scope of the present deliverable is to present the work carried out in the context of Task 2.3 of ACCEPT project. More precisely, it is focused on system architecture design process up to Month 10 (M10) of the project, including the high-level architecture model of the complete system. The ACCEPT system conceptual architecture was briefly presented in the Description of Work (DoW) and was further updated based on work carried out during this period with the involvement of all relevant technology providers. This high-level architecture has also been mapped to the various SGAM layers in order to assist the future integration and development activities.

The main part of the document is focused on a detailed presentation of the system's software components. The functional and non-functional specifications of the components, their interdependencies with other components and the required input/output data for each of them, were provided from all relevant partners, through a dedicated template that was created and circulated among them. In addition, detailed component diagrams were created and linked to Deliverable 2.1 (D2.1), correlating each component to relevant Use Cases (UCs) and describing relevant functionalities for each of them.

1.2. Structure of the Deliverable

The deliverable is structured as follows:

- Chapter 2 described the methodological process adopted, and presents the high-level system architecture and its mapping to the SGAM model;
- Chapter 3 presents the detailed(non-)functional specifications, interdependencies, UC correlation, implementation view and input/output requirements of the system's constituent components.
- Chapter 4 concludes this document;
- Annex I include the component characterization template.

1.3. Interdependencies with Other Tasks and Deliverables

The deliverable reports on activities that have been performed in the context of "T2.3 System architecture design, configuration & elaboration (M4-M18)", with main outcome the first version of ACCEPT's system architecture. It accompanies and heavily depends on the work performed in T2.1 of the project and its associated deliverable "D2.1 ACCEPT business scenarios, use cases & requirements". The work carried out here will drive the development of the first prototype of the software components in WP4 and WP5, as shown in Figure 1.

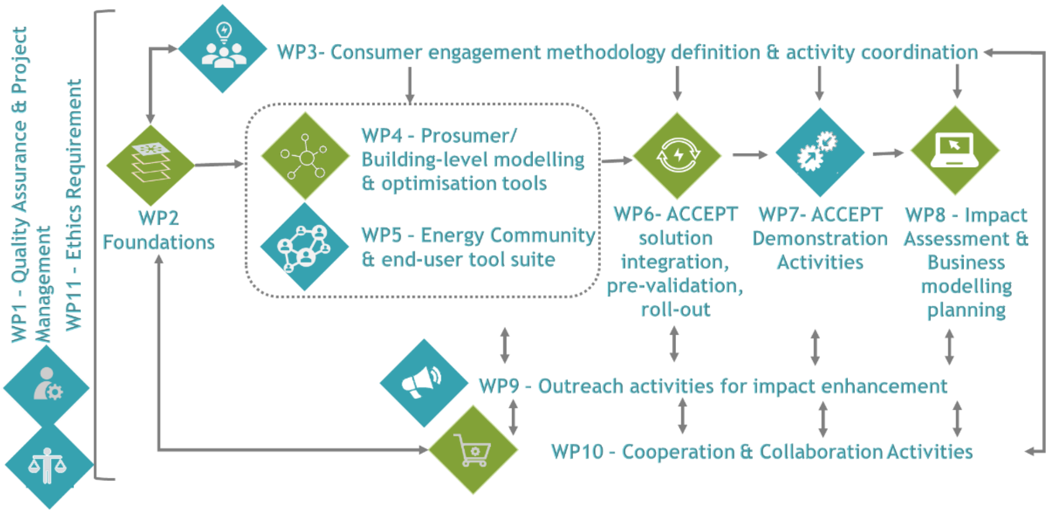

WP1 - Quality Assurance & Project Management
WP11 - Ethics Requirement
WP2 Foundations
WP3- Consumer engagement methodology definition & activity coordination
WP4 - Prosumer/ Building-level modelling & optimisation tools
WP5 - Energy Community & end-user tool suite
WP6- ACCEPT solution integration, pre-validation, roll-out
WP7- ACCEPT Demonstration Activities
WP8 - Impact Assessment & Business modelling planning
WP9 - Outreach activities for impact enhancement
WP10 - Cooperation & Collaboration Activities

Figure 1. ACCEPT Work Package interdependency Graph

2. ACCEPT System Architecture

2.1. Methodological Framework

The initial step towards defining ACCEPT's system architecture and providing the required specifications for the development of the necessary components has been the analysis of T2.1 results. The scope of T2.1 has been the extraction of the end-user requirements, as well as the creation of the business and use cases of the project. With the user requirements and Use Cases (UC) elicitation being the scope of T2.1, upon which the system functionalities and architecture are then constructed, it is worth mentioning that within D2.1, each UC was described not only via a textual and a Unified Markup Language (UML) UC diagram, but also through a sequence diagram detailing the interactions between the different components. As such, the inclusion of those diagram in this deliverable was deemed redundant. Therefore, the two documents must be seen as complementary to each other in their content and purpose.

Upon finalisation of the UCs' definition and submission of D2.1, the work pertaining T2.3 was initiated. Here, the first step was the provision of a robust and concrete methodology, based on which the technical partners could derive the necessary information for the subsequent development tasks, considering the requirements derived by the aforementioned analysis. The steps of the adopted methodology included the followings:

1. Analysis of UCs for extraction of functional requirements and validation of responsible components/ contributing partners;
2. Introduction to necessary concepts and tools for the creation of the UML component diagrams. The online free suite diagrams.net was adopted for the generation of the diagrams. Hypertech performed a short tutorial call with other partners to introduce the tool;
3. Generation of the component characterization template, which incorporated all fields identified as necessary for the accumulation of the required information; the template was created by Hypertech and was distributed to all involved partners, along with instructions for properly filling in the required information;
4. Creation of the high-level system architecture by Hypertech, which was distributed to all partners; along with the characterisation templates;
5. Collection of filled in templates, consolidation, and evaluation of the content;
6. Second iteration of the components' characterisation, in order to align interdependencies between components and verify that all required functionalities are considered; and finally
7. Refinement of the high-level system architecture and creation of the SGAM mapping.

In the following sections, we present the results of this process, starting from the high-level architecture, and then moving on to the individual software components. What is reported here will be used as reference for the development of the first prototype ACCEPT system. After its evaluation, any lessons learnt will be integrated into the second version of ACCEPT's architecture (D2.4, due M18).

2.2. High-Level System Architecture

ACCEPT's high-level system architecture can be seen in Figure 1. It was derived from the initial description of the ACCEPT solution and its components in the DoW, further refined and updated to cover any new requirements and reflect the current status of work performed within WP2. Arrows in the diagram indicate functional or data dependencies from the source packages to the pointed packages.

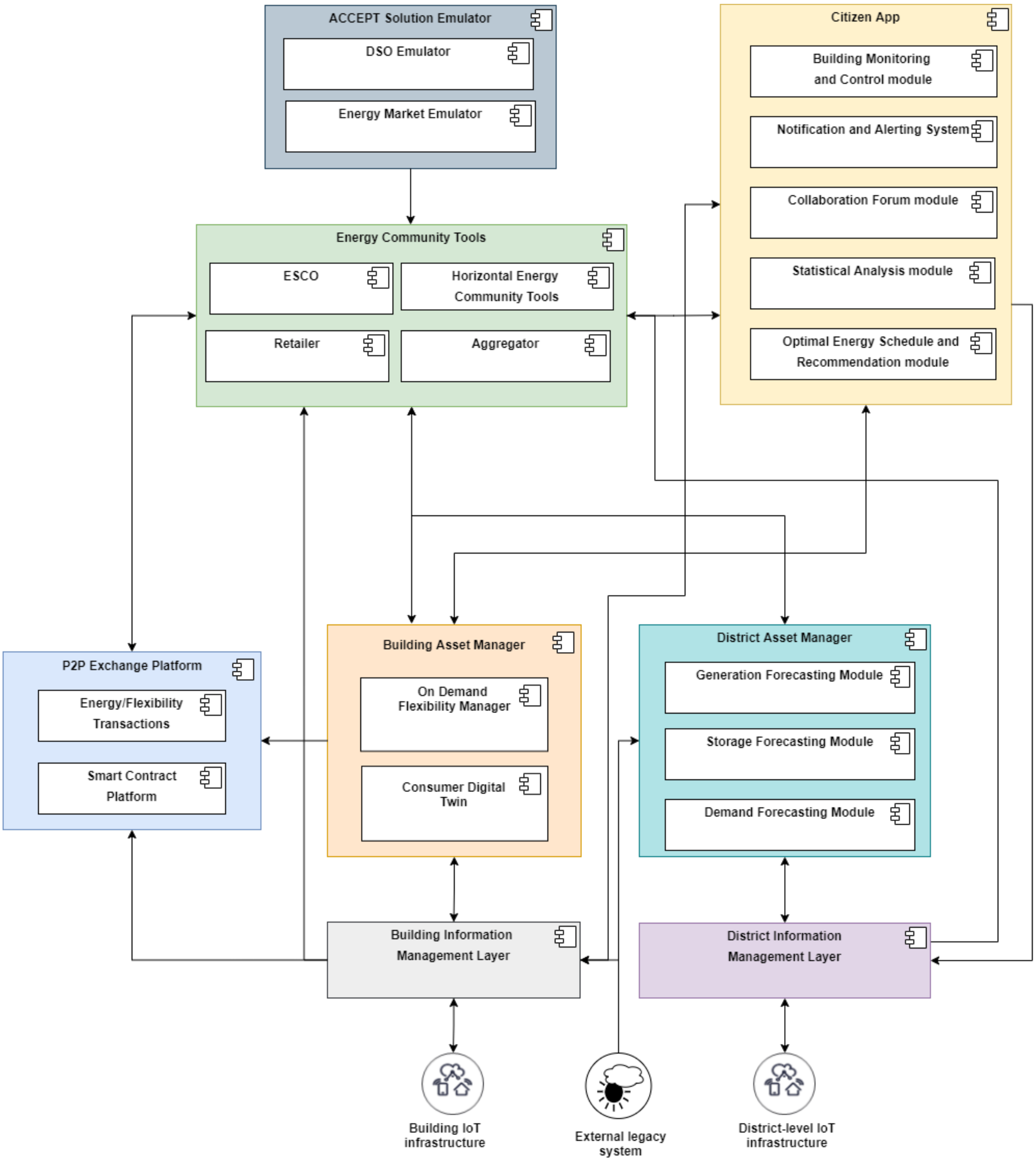

ACCEPT Solution Emulator
DSO Emulator
Energy Market Emulator
Citizen App
Building Monitoring and Control module
Notification and Alerting System
Collaboration Forum module
Statistical Analysis module
Optimal Energy Schedule and Recommendation module
Energy Community Tools
ESCO
Horizontal Energy Community Tools
Retailer
Aggregator
P2P Exchange Platform
Energy/Flexibility Transactions
Smart Contract Platform
Building Asset Manager
On Demand Flexibility Manager
Consumer Digital Twin
District Asset Manager
Generation Forecasting Module
Storage Forecasting Module
Demand Forecasting Module
Building Information Management Layer
District Information Management Layer
Building IoT infrastructure
External legacy system
District-level IoT infrastructure

Figure 2. ACCEPT high-level system architecture

The main composite components of the ACCEPT system have remained largely unchanged, both in terms of naming (slight adjustments have been made for consistency and accuracy reasons) and in terms of general functionalities, from their descriptions in the DoW. Below we shortly restate the main objectives of each such part of the system. For extended information on all the components, the reader is directed to the next chapter.

- **Building Information Management Layer (BIML):** It is responsible for capturing live information streams from a host of sensors, meters and actuators in the building, pre-process the data, perform disaggregation, prepare semantically-enhanced information for the following components and facilitate the downstream communication of control commands. One of the main roles of this component is ensuring data security.
- **District Information Management Layer (DIML):** It is responsible for capturing live information streams from available district-level assets owned by the communities taking part in the project, pre-process data, perform disaggregation, prepare semantically-enhanced information for other ACCEPT solution components and facilitate downstream communication of control actions.
- **District Asset Management (DAM):** It includes the functionalities for monitoring, control and management of district-level assets (e.g., photovoltaic (PV) panels, Electric Vehicle (EV) charging, storage, heat pumps) to enable their integration in the Virtual Power Plant that encapsulates all energy assets under the direct control of the energy community (or within the portfolio managed by the market actor) and whose flexibility can be leveraged for the business objectives of the community.
- **Building Asset Manager (BAM):** This composite component wraps all components related to management and analysis of building-derived data. It includes the On-Demand Flexibility Management (ODFM), which is responsible for consumer/building-level optimizations, the Consumer Digital Twin, which provides mathematical models of the building and the citizens-residents' preferences, as well as the Dynamic SRI Performance Rating, which is responsible for quantifying and updating the Smart Readiness Indicator (SRI).
- **Community-Level P2P Exchange Platform (P2P-EP):** It is a blockchain-based and smart-contract-enabled platform to facilitate the exchange of energy or flexibility between the community members for individual consumer or community level optimization.
- **Citizen Application (C-App):** They offer desirable functionalities to citizens through energy and non-energy services (e.g., convenience, security, ambient assisted living features) based on the infrastructure and background services necessary for the provision of demand response services from buildings in order to enhance the value proposition of the ACCEPT.
- **Energy Community Tools (ECTs):** They are a collection of tools that enable the energy community to collectively manage the assets at hand as either of three market roles: retailer, Energy Service Company (ESCo) or aggregator – or any combination of these. These tools facilitate Virtual Power Plant management for consolidation and optimal use of available flexibility in markets or retailer operations as well as energy self-consumption optimization at the community level.

2.3. SGAM mapping

2.3.1. SGAM Introduction

As stated in T2.3, one of the goals is to present a mapping of the system architecture on the SGAM model, so as to promote reusability and also enable comparisons with the system architectures proposed by other BRIDGE projects. SGAM was introduced by the Smart Grid Coordination Group "as a methodology for describing smart grid use cases and services with an architectural approach. This methodology allows a neutral representation of the involved technologies highlighting their interoperability supported by standards and, consequently, enabling standards gap analysis"[1]. SGAM's first level of abstraction consists of five interoperable layers:

- The **business layer** represents the business view on the information exchange
- The **function layer** describes functions and services including their relationships from an architectural viewpoint
- The **information layer** describes the information that is being used and exchanged between functions, services and components.

[1] Estebsari, Abouzar & Barbierato, Luca & Bahmanyar, Alireza & Bottaccioli, Lorenzo & Macii, Enrico & Patti, Edoardo. (2019). A SGAM-Based Test Platform to Develop a Scheme for Wide Area Measurement-Free Monitoring of Smart Grids under High PV Penetration. Energies. 12. 1417. 10.3390/en12081417.

- The **communication layer** describes protocols and mechanisms for the interoperable exchange of information between components
- The **component layer** shows the physical distribution of participating components in the smart grid context.

It has to be stressed that SGAM offers further classification properties within each layer, effectively providing a three-dimensional architectural model. For the stated purposes though, it was deemed sufficient to perform only the first level mapping of ACCEPT's system architecture.

2.3.2. Mapping System Architecture to the SGAM model

Figure 3 and Figure 4 restate the ACCEPT system architecture, but this time under the perspective of the Communication and Information layers of SGAM, respectively. Each component is placed over the relevant area of the SGAM plane. To improve readability of the figures, sub-components of the ACCEPT solution have been removed. Furthermore, high level information about connections between the composite components have been provided. Details about each connection are provided in Section 3.

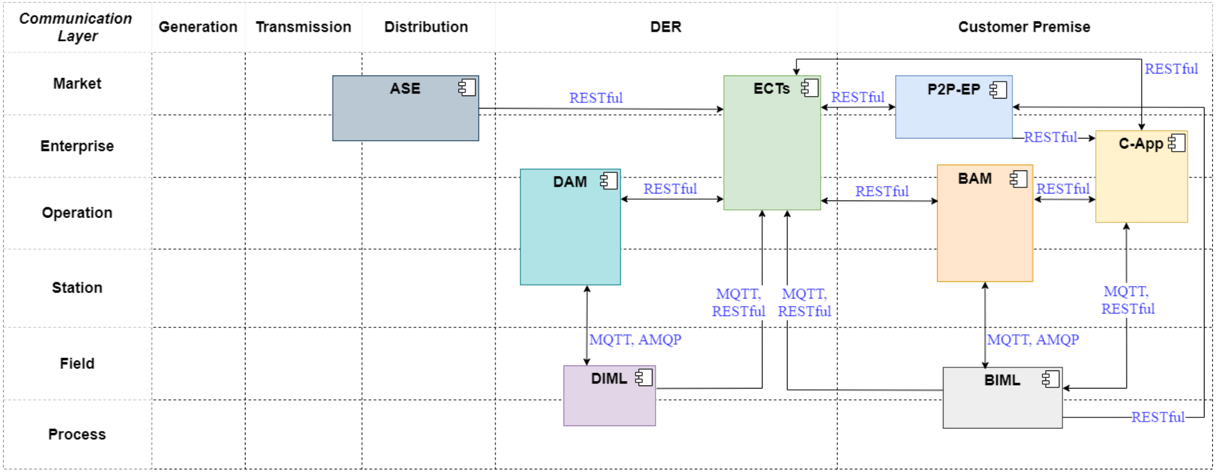


Figure 3. ACCEPT system architecture - SGAM Communication Layer mapping

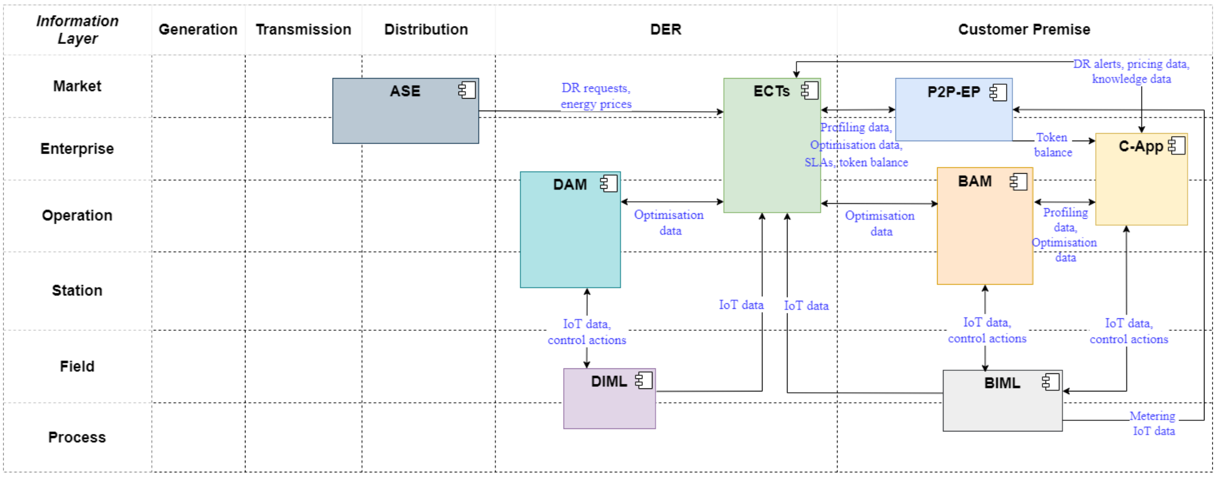

Information Layer
Generation
Transmission
Distribution
DER
Customer Premise
Market
Enterprise
Operation
Station
Field
Process
ASE
ECTs
P2P-EP
C-App
DAM
BAM
DIML
BIML
DR alerts, pricing data, knowledge data
DR requests, energy prices
Profiling data, Optimisation data, SLAs, token balance
Token balance
Optimisation data
Optimisation data
Profiling data, Optimisation data
IoT data
IoT data
IoT data, control actions
IoT data, control actions
IoT data, control actions
Metering IoT data

Figure 4. ACCEPT system architecture - SGAM Information Layer mapping

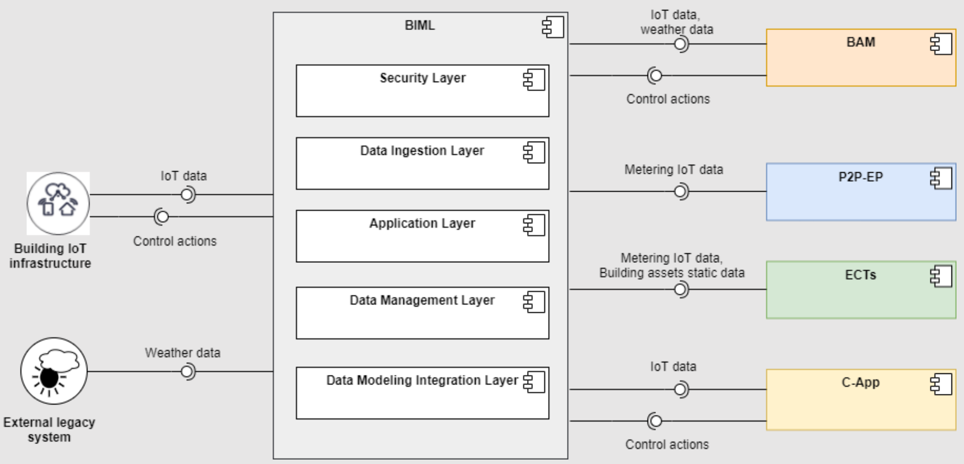
These figures provide a solid basis for the ACCEPT-to-SGAM mapping process. Since this work reflects the activities that have been accomplished for the first version of the ACCEPT system architecture, assumptions and restrictions have been considered. Hence, refinements and extensions of the ACCEPT-to-SGAM mapping for all layers are anticipated in the final version of the system architecture.

3. ACCEPT Components' Specifications and Requirements

In this chapter we present the filled in templates of all identified components comprising the ACCEPT software system. The original template, along with guidelines/clarifications on what exactly information should be filled in in each field can be found in Annex I – Component's Specifications and Requirements Template. Each section in this chapter is named after one of the six composite components of the ACCEPT system. Within, subsections, if any, correspond to the main constituent components respectively.

3.1. Building Information Management Layer – BIML

General Information	
Component name	Building Information Management Layer (BIML)
Component Description	BIML is the component responsible for interfacing with the real world, by collecting metering, sensing and monitoring data from the physical assets of the building and processing it for further use from other components. It is a combination of a software and a hardware system. The hardware part consists of an IoT gateway that enables bidirectional interoperable communication between building's sensors, meters and actuators and the rest of the ACCEPT system components. The software stack allows for the continuous processing of the live data streams by a smart ingestion engine. The BIML follows a multilayer architecture, consisting of five main layers: • The Security Layer is responsible for preserving data security and enforcing information access control to other components. • In the Data Ingestion Layer, data from IoT infrastructures is ingested into a queue of messages to be processed before being permanently stored in the BIML data repository. Data ingestions are delivered as messages through message-brokers to ensure stability, delivery consistency, fault-tolerance transmission, asynchronous communication between services, increasing reliability and system performance. • In the Application Layer data stored in the BIML data repository or directly received from the message brokers are divided into small batches and sent for processing; applications for real-time data-processing (e.g., cleansing, normalising) are included in this layer. • The Data Management Layer provides a common storage cloud infrastructure, accommodating all the building level IoT and EV data. • The Data Modelling Integration Layer hosts the IoT and EV static and stream data mapping to the BIML internal data models.
Interfaces with End-Users	None
Relevant UCs	UC1, UC2, UC3, UC4, UC8, UC9, UC11, UC13
Lead Developing Partner	QUE
Programming Language(s)	Java, Scala
Deployment Requirements	Physical components require manual installation in premises. For the Wireless Sensor Network (WSN) design and installation, audits of pilot sites are required. The IoT data ingestion and processing will be deployed as a cloud service.
Specifications	
Functional Specifications	1. Establish connection with meters, sensors and actuators
	2. Receive device status and data streams
	3. Enable dispatching of control commands
	4. Protect sensible and private data

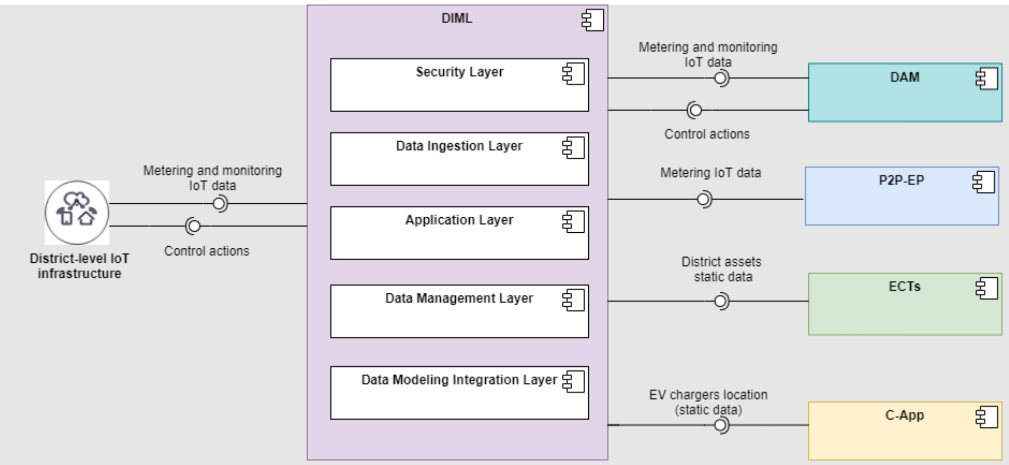
	5. Cleanse incoming stream of data			
	6. Perform data Normalization and/or Aggregation			
	7. Perform non-intrusive load monitoring			
	8. Store data locally			
Functional Dependencies	**Function**	**Responsible Component**		
	Generate Control Actions for activation of flexibility	Building Demand Manager		
Non-functional Specifications	1. Scalability – integration of a large number of devices			
	2. Robustness and error reporting			
	3. Interoperability to enable communication with a variety of sensing, metering and actuating devices.			
Implementation View				
Component Diagram	 BIML Security Layer Data Ingestion Layer Application Layer Data Management Layer Data Modeling Integration Layer IoT data Control actions Building IoT infrastructure Weather data External legacy system IoT data, weather data Control actions BAM Metering IoT data P2P-EP Metering IoT data, Building assets static data ECTs IoT data Control actions C-App			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	IoT data[2]	Building IoT infrastructure	JSON	AMQP
	Weather data	External legacy system	JSON	RESTful
	Control actions	Building Asset Manager	JSON	AMQP
	Control actions	Citizen App	JSON	AMQP
	Data	To	**Payload Format**	**Communication protocol**
Output	IoT data	Building Asset Manager	JSON	RESTful
	Weather data	Building Asset Manager	JSON	RESTful
	Metering IoT data	P2P Exchange Platform	JSON	RESTful
	Metering IoT data	Energy Community Tools	JSON	RESTful
	Building asset static data[3]	Energy Community Tools	JSON	RESTful
	IoT data	Citizen App	JSON	RESTful
	Control actions	Building IoT infrastructure	JSON	RESTful

[2] IoT data include metering, sensing and monitoring data from downstream (low-level) IoT infrastructure. More specifically, metering data refer to data gathering by metering devices (such as smart clamps and smart meters), sensing data refer to ambient condition data gathered through sensors, and monitoring data refer to data obtained from the actuators/controllers of the IoT infrastructure.

[3] Such data refer mainly to the characteristics (attributes) of building-level assets.

3.2. District Information Management Layer – DIML

General Information		
Component name	District Information Management Layer (DIML)	
Component Description	Similar to the BIML for building-level assets, the DIML enables bidirectional communication with district-level DER elements, incl. generation, storage and demand district-level assets. The DIML 's software consists of five main layers, identical to the layers of BIML for the district-level assets monitoring, metering and sensing data management (see Section 3.1): • The Security layer; • the Data Ingestion layer; • the Application layer; • the Data Management layer; and • the Data Modelling Integration layer.	
Interfaces with End-users	Citizens (through citizen app)	
Relevant UCs	UC5, UC6, UC8, UC9, UC10, UC12, UC13	
Lead Developing Partner	CIRCE	
Programming Language(s)	C++, Python	
Deployment Requirements	Physical component(s) that require(s) deployment at the demo site (the district-level IoT infrastructure).	
Specifications		
Functional Specifications	1. Establish connection with meters, sensors and actuators	
	2. Receive device status and data streams	
	3. Enable dispatching of control commands	
	4. Protect sensible and private data	
	5. Cleanse incoming stream of data	
	6. Perform data Normalization and/or Aggregation	
Functional Dependencies	**Function**	**Responsible Component**
	Generate Control Actions for activation of flexibility from district-level DERs	District Asset Manager (DAM)
Non-functional Specifications	Secure communications	
	Scalability – integration of a large number of devices	
	Robustness and error reporting	
	Interoperability to enable communication with a variety of sensing, metering and actuating devices.	
Implementation View		
Component Diagram		

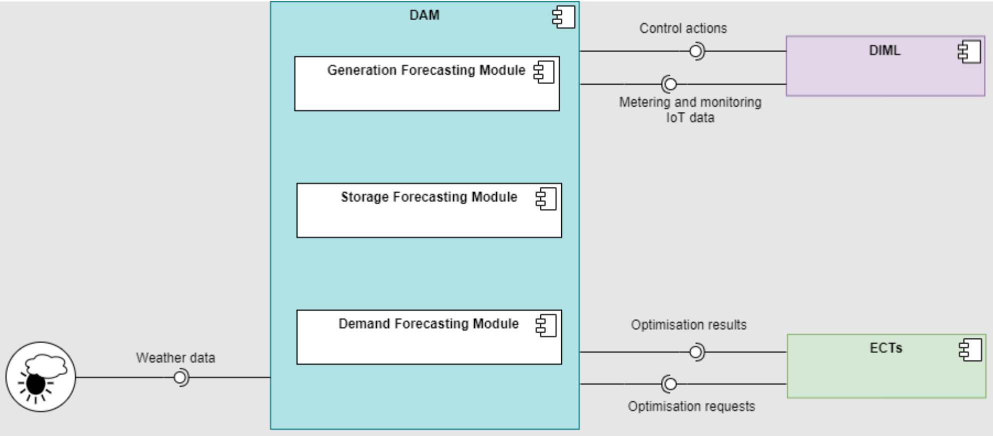
	 DIML Security Layer Data Ingestion Layer Application Layer Data Management Layer Data Modeling Integration Layer District-level IoT infrastructure Metering and monitoring IoT data Control actions Metering and monitoring IoT data Control actions DAM Metering IoT data P2P-EP District assets static data ECTs EV chargers location (static data) C-App			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	Monitoring and metering IoT data	Available district-level IoT infrastructure	TBD	-
	Control actions for district-level DERs	District Asset Manager	JSON	MQTT
	Data	To		**Communication protocol**
Output	Control actions	Available district-level IoT infrastructure	TBD	TBD
	Metering and monitoring IoT data	District Asset Manager	JSON	MQTT
	Metering IoT data	P2P Exchange Platform	JSON	MQTT
	District assets static data	Energy Community Tools	JSON	MQTT
	EV chargers location	Citizen App	JSON	MQTT

3.3. District Asset Manager – DAM

General Information	
Component name	District Asset Manager (DAM)
Component Description	The DAM retrieves information on available district-level assets from the DIML and generates forecasts of their operation, which are then passed to the energy community tools for consideration under various community-wide optimisation scenarios (such as community-wide self-consumption). The DAM is also responsible for receiving optimisation requests from the energy community tools and translating them into control actions for the DIML. The DAM comprises three main sub-components, namely the Generation Forecasting Module, the Storage Forecasting Module and the Demand Forecasting Module, responsible for the generation of forecasts for generation, storage and demand district-level assets respectively.
Interfaces with End-users	None
Relevant UCs	UC5, UC8, UC9, UC10, UC12, UC13

Lead Developing Partner	CIRCE
Programming Language(s)	Ada, Go, C++, Python
Deployment Requirements	Deployment in the cloud. The installation of the DIML in the relevant pilot sites.
Specifications	
Functional Specifications	1. Forecast district-level generation 2. Forecast district-level demand (demand from district-level loads/assets, such as district-level heat pumps) 3. Forecast district-level storage capacity and usage based on information on load and generation, as well as dynamic pricing 4. Create demand flexibility forecasts (upwards/downwards) for storage and demand assets 5. Generate control actions for district-level assets 6. Retrieve weather data

	Function	**Responsible Component**
Functional Dependencies	Monitor district-level devices	District Information Management Layer
	Apply control timeseries	District Information Management Layer
	Optimize community self-consumption	Energy Community Tools
	Apply portfolio DR request	Energy Community Tools

Non-functional Specifications	Secure communications
Implementation View	
Component Diagram	

DAM
Generation Forecasting Module
Storage Forecasting Module
Demand Forecasting Module
Control actions
Metering and monitoring IoT data
DIML
Optimisation results
Optimisation requests
ECTs
Weather data

Interfaces/Data Requirements

	Data	**From**	**Payload Format**	**Communication protocol**
Input	Metering and monitoring IoT data	District Information Management Layer	JSON	MQTT
	Optimisation requests	Energy Community Tools	JSON	RESTful
	Weather data	External legacy system	JSON	RESTful
	Data	To	**Payload Format**	**Communication protocol**
Output	Control actions	District Information Management Layer	JSON	MQTT

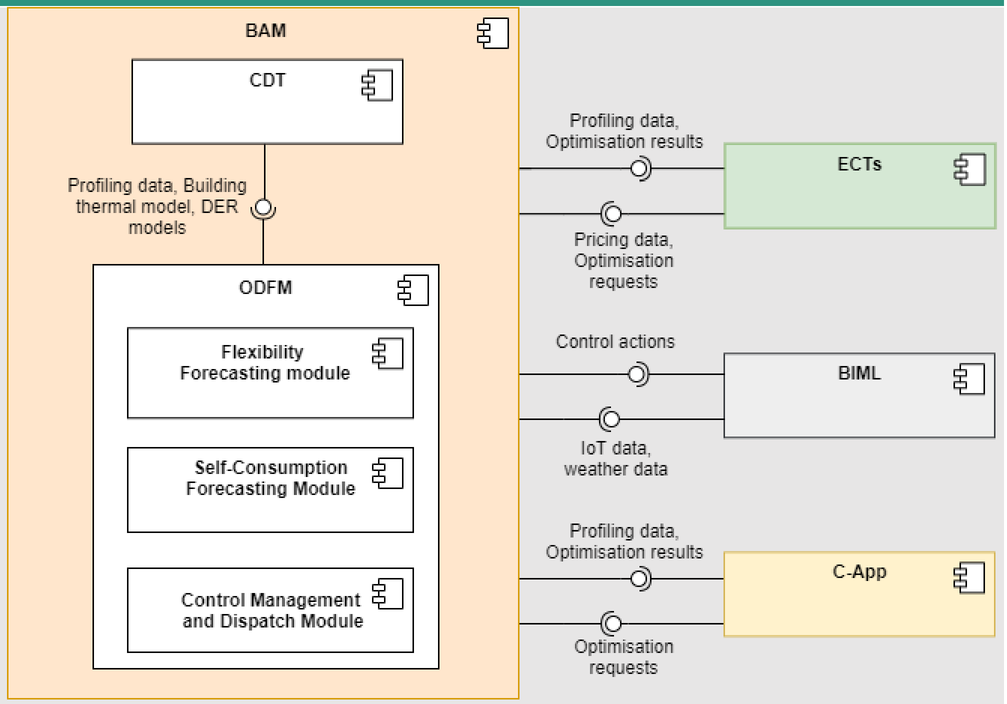
	Optimisation results	Energy Community Tools	JSON	RESTful

3.4. Building Asset Manager – BAM

The Building Asset Manager (BAM) wraps the various components that are deployed at building level, in particular the On Demand Flexibility Manager (ODFM) and the Consumer Digital Twin (CDT). As such, it suffices to characterize its constituent parts.

3.4.1. On Demand Flexibility Manager – ODFM

General Information		
Component name	On Demand Flexibility Manager (ODFM)	
Component Description	The ODFM constitutes the optimisation and control dispatch engine within the Building Asset Manager and comprises the Flexibility Forecasting Module, the Self-consumption Forecasting Module and the Control Management and Dispatch Module. In summary, the component will provide the following functionalities: • Self-consumption scenarios optimisation, aiming to schedule the use of resources in such a way that as much self-generated energy is consumed during the day; • Energy/cost explicit Demand Response (DR) scenarios optimisation, where the baseline and alternative possible timeseries of electricity consumption (which are then bundled as offered flexibility) are calculated by solving optimization problems with different objective functions and comfort constraints' formulations; • Control dispatch, for breaking down a DR request and computing the necessary control actions for the building assets. A local repository within the ODFM will be collecting and storing the data and control demands from the BIML, which will then be utilised for performing optimisations.	
Interfaces with End-users	None	
Relevant UCs	UC2, UC3, UC7, UC8, UC10, UC11, UC13	
Lead Developing Partner	Hypertech	
Programming Language(s)	Python, Java	
Deployment Requirements	Deployed as a cloud service. Requires installation of the BIML at the relevant pilot sites and adequate historical IoT data for the Consumer Digital Twin (CDT) models training.	
Specifications		
Functional Specifications	1. Build baseline optimisation system	
	2. Build extended comfort optimisation	
	3. Forecast environmental conditions	
	4. Minimize energy bought from the grid	
	5. Minimize cost of purchased energy from the grid	
	6. Maximize consumption of self-generated energy	
	7. Compile Flexibility timeseries	
	8. Translate devices' consumption timeseries to control timeseries	
Functional Dependencies	**Function**	**Responsible Component**
	Model building residents' occupancy	Consumer Digital Twin
	Model building residents' comfort	Consumer Digital Twin

	Model building thermal spaces	Consumer Digital Twin
	Model building Heating, Ventilation and Air-Conditioning (HVAC) systems	Consumer Digital Twin
	Model building Domestic Hot Water (DHW) systems	Consumer Digital Twin
	Model building generation (renewables), EVs and Stationary Battery systems	Consumer Digital Twin
	Retrieve raw data	Building Information Management Layer
	Monitor devices	Building Information Management Layer
	Apply control timeseries	Building Information Management Layer
	Optimize community self-consumption	Energy Community Tools
	Apply portfolio DR request	Energy Community Tools
Non-functional Specifications	1. Raw data and control commands securely transferred to/from BIML and stored	
	2. One ODFM active for each pilot building	
	3. Self-consumption optimization and/or Flexibility forecasting provided either on demand or on a scheduled basis	
Implementation View		
Component Diagram	 BAM CDT Profiling data, Building thermal model, DER models ODFM Flexibility Forecasting module Self-Consumption Forecasting Module Control Management and Dispatch Module Profiling data, Optimisation results ECTs Pricing data, Optimisation requests Control actions BIML IoT data, weather data Profiling data, Optimisation results C-App Optimisation requests	

Interfaces/Data Requirements

	Data	**From**	**Payload Format**	**Communication protocol**
Input	Profiling data[4]	Consumer Digital Twin	JSON	RESTful
	Fitted building thermal model	Consumer Digital Twin	JSON	RESTful

[4] Profiling data include occupancy profiles, comfort profiles, citizen lifestyle and activity patterns, and EV profiles (e.g., mobility habits, such as driving and charging patterns).

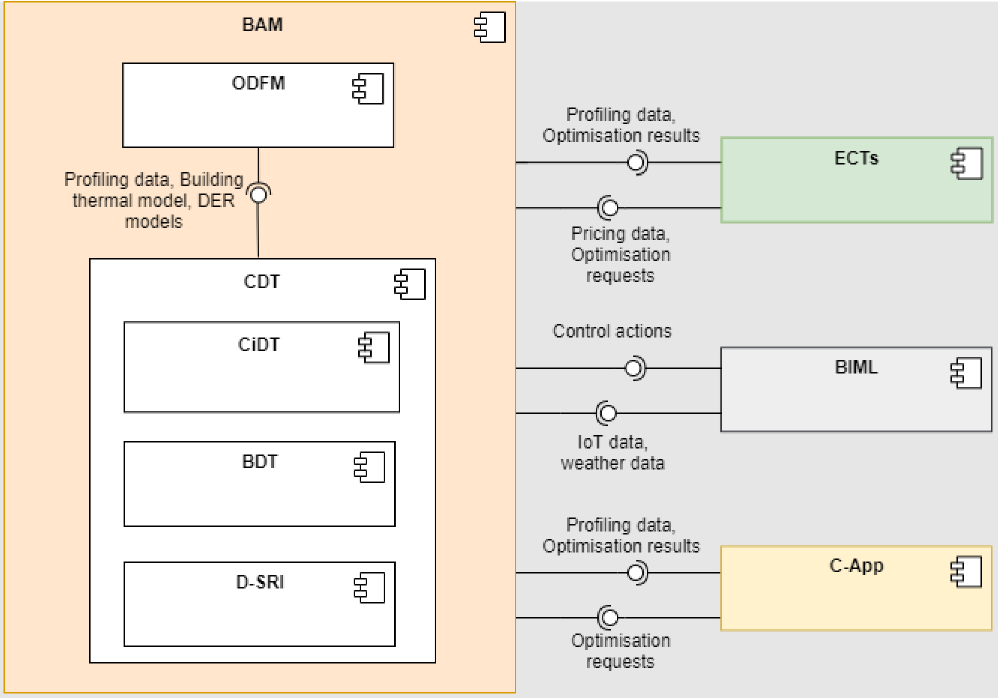
	Fitted DER models[5]	Consumer Digital Twin	JSON	RESTful
	Optimisation requests	Energy Community Tools	JSON	RESTful
	IoT data	Building Information Management Layer	JSON	RESTful
	Weather data	Building Information Management Layer	JSON	RESTful
	Optimisation requests	Citizen Application	JSON	RESTful
	Retail price forecast	Energy Community Tools	JSON	RESTful
	Data	**To**	**Payload Format**	**Communication protocol**
Output	Profiling data	Energy Community Tools	JSON	RESTful or AMQP
	Optimisation results	Energy Community Tools	JSON	RESTful or AMQP
	Profiling data	Citizen App	JSON	RESTful or AMQP
	Optimisation results	Citizen App	JSON	RESTful or AMQP
	Control actions	Building Information Management Layer	JSON	RESTful or AMQP

3.4.2. Consumer Digital Twin – CDT

General Information	
Component name	Consumer Digital Twin (CDT)
Component Description	The Consumer Digital Twin (CDT) component is responsible for exposing the main modelling functionalities to the rest of the BAM components. It consists of three main subcomponents, namely the Citizen Digital Twin (CiDT), the Building Digital Twin (BDT) and the Dynamic Smart Readiness Indicator (D-SRI). The CiDT has as its main objective to mathematically capture the behavioural profiles of the building residents and provide an energy-related characterization of their habits. In essence, the component will perform the following: • Learn and forecast the occupancy patterns of the residents, • Identify the thermal comfort preferences of the end-users, • Estimate the demand patterns for usage of Domestic Hot Water (DHW), as well as for EV charging, if any is present, • Capture the price elasticity of demand for each household when variable tariff schemes are applied. The BDT sits alongside the CiDT in order to provide the thermal modelling of the building envelope (thermal resistance and capacitance), as well as the characterization of any energy resources that are located within the building, both in terms of consuming loads (HVAC) as well as energy storage and generation systems. The models integrated in this system will be utilized for flexibility forecasting as well as translating DR requests to control dispatches. The D-SRI component is the third sub-component of CDT. Its purpose is to calculate in a dynamic fashion (time and resident dependent) an SRI-based Smart Readiness Indicator. To achieve this, the component will take advantage of the modelling capabilities of the other

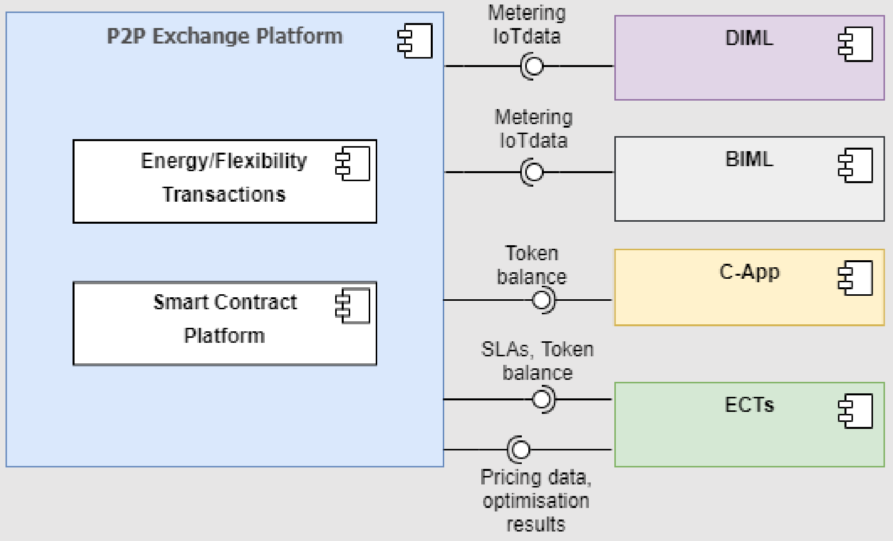
[5] DER models may include HVAC, DHW, EV, storage and PV models (depending on building-level device availability at pilot sites).

	BAM components in order to periodically update appropriately selected/relevant metrics based on the streams of data at hand.	
Interfaces with End-users	None	
Relevant UCs	UC2, UC3, UC4, UC6, UC8, UC10, UC13	
Lead Developing Partner	Hypertech	
Programming Language(s)	Python, Java	
Deployment Requirements	Deployed as a cloud service	
Specifications		
Functional Specifications	**CiDT:**	
	1. Model building residents' occupancy	
	2. Model building residents' comfort	
	3. Model building residents' demand patterns	
	4. Fit occupancy model	
	5. Fit comfort model	
	6. Retrieve environmental conditions and retail prices	
	7. Model user EV usage patterns	
	8. Fit elasticity regression model	
	9. Estimate consumer elasticity	
	BDT:	
	1. Model building thermal spaces	
	2. Model building HVAC systems	
	3. Model building DHW systems	
	4. Model building generation (renewables) and Stationary Battery systems	
	5. Fit thermal space model	
	6. Fit HVAC model	
	7. Fit DHW model	
	8. Fit EV model	
	9. Fit generation system model	
	10. Retrieve environmental conditions	
	D-SRI:	
	1. Calculate static SRI performance	
	2. Calculate dynamic building SRI performance	
	3. Send SRI score to Citizen App / UI	
Functional Dependencies	**Function**	**Responsible Component**
	Retrieve raw data	Building Information Management Layer
	Retrieve citizen profiles	Consumer Digital Twin
	Retrieve metering/sensing data	Building Information Management Layer
	Visualize results	Citizen App
Non-functional Specifications	1. Secure handling of personal data	
	2. Automated periodic updating of models	
	3. One instance deployed per pilot building	
	4. Automated scheduling for periodic updates of the SRI metrics	

	5. Uninterrupted communication with other components of the CDT and the BIML for model and data retrieval.			
	6. One instance deployed per pilot building			
Implementation View				
Component Diagram	 BAM ODFM Profiling data, Building thermal model, DER models CDT CiDT BDT D-SRI Profiling data, Optimisation results ECTs Pricing data, Optimisation requests Control actions BIML IoT data, weather data Profiling data, Optimisation results C-App Optimisation requests			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	IoT data	Building Information Management Layer	JSON	RESTful
	Weather data	Building Information Management Layer	JSON	RESTful
	Pricing data	Energy Community Tools	JSON	RESTful
	Optimisation requests	Energy Community Tools	JSON	RESTful
	Optimisation requests	Citizen App	JSON	RESTful
	Data	To	**Payload Format**	**Communication protocol**
Output	Profiling data	On Demand Flexibility Manager	JSON	RESTful or AMQP
	Fitted building thermal model	On Demand Flexibility Manager	JSON	RESTful or AMQP
	Fitted DER models	On Demand Flexibility Manager	JSON	RESTful or AMQP
	Profiling data	Energy Community Tools	JSON	RESTful or AMQP
	Optimisation results	Energy Community Tools	JSON	RESTful or AMQP
	Control actions	Building Information Management Layer	JSON	RESTful or AMQP
	Profiling data	Citizen App	JSON	RESTful or AMQP
	Optimisation results	Citizen App	JSON	RESTful or AMQP

3.5. Community-level P2P Exchange Platform

General Information		
Component name	Community-level P2P exchange platform (P2P-EP)	
Component Description	The Community-level P2P exchange platform (P2P-EP) comprises two main components, the Energy/Flexibility Transactions (EFT) component and the Smart Contract Platform (SCP). The EFT is a blockchain based and smart contract enabled platform that primarily will facilitate the exchange of energy and flexibility among community members in a trustworthy and transparent manner. On a second level the Platform will enable the execution of Smart Contracts to allow the delivery of energy and non-energy services to the community members. The SCP is a tool with a local database that allows: • the creation of Smart Contracts based on predefined Service Level Agreements; • the Instantiation of Smart Contracts; and • the monitoring of Service Level Agreements related KPIs.	
Interfaces with End-Users	None	
Relevant UCs	UC7	
Lead Developing Partner	QUE	
Programming Language(s)	Go (with Java Plugins), RESTFul, Java, Rust	
Deployment Requirements	Linux based operational environment with min 1.5GHz, 64-bit quad-core processor, 2GB SDRAM memory and broad connectivity.	
Specifications		
Functional Specifications	**EFT:**	
	1. Store Transaction Data	
	2. Check Balance of the user	
	3. Receive Smart Contract Results	
	4. Anonymize user	
	5. Receive Market related inputs	
	SCP:	
	1. Provide predefined SLAs and KPI description	
	2. Provide SLAs to authorized Service Providers	
	3. Create SLA – user bundles	
	4. Save Complete SLAs	
	5. Monitor SLA related KPIs	
	6. Instantiate Smart Contracts	
	7. Update Smart Contracts	
Functional Dependencies	**Function**	**Responsible Component**
	Receive Market Inputs	Energy Community Tools
	Receive Energy Production/Consumption data	Building Information Management Layer
	Receive SLA – user bundles	Energy Community Tools
	Update smart contracts	Building Asset Manager
Non-functional Specifications	1. Scalability	
	2. Stability	

	3. Security			
Implementation View				
Component Diagram	 P2P Exchange Platform Energy/Flexibility Transactions Smart Contract Platform Metering IoTdata DIML Metering IoTdata BIML Token balance C-App SLAs, Token balance ECTs Pricing data, optimisation results			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	Metering IoT data	District Information Management Layer	JSON	MQTT
	Metering IoT data	Building Information Management Layer	JSON	RESTful
	Pricing data	Energy Community Tools	JSON	RESTful
	Optimisation results	Energy Community Tools	JSON	RESTful
	Data	**To**	**Payload Format**	**Communication protocol**
Output	Token Balance	Citizen App	JSON	RESTful
	Token Balance	Energy Community Tools	JSON	RESTful
	SLAs	Energy Community Tools	JSON	RESTful

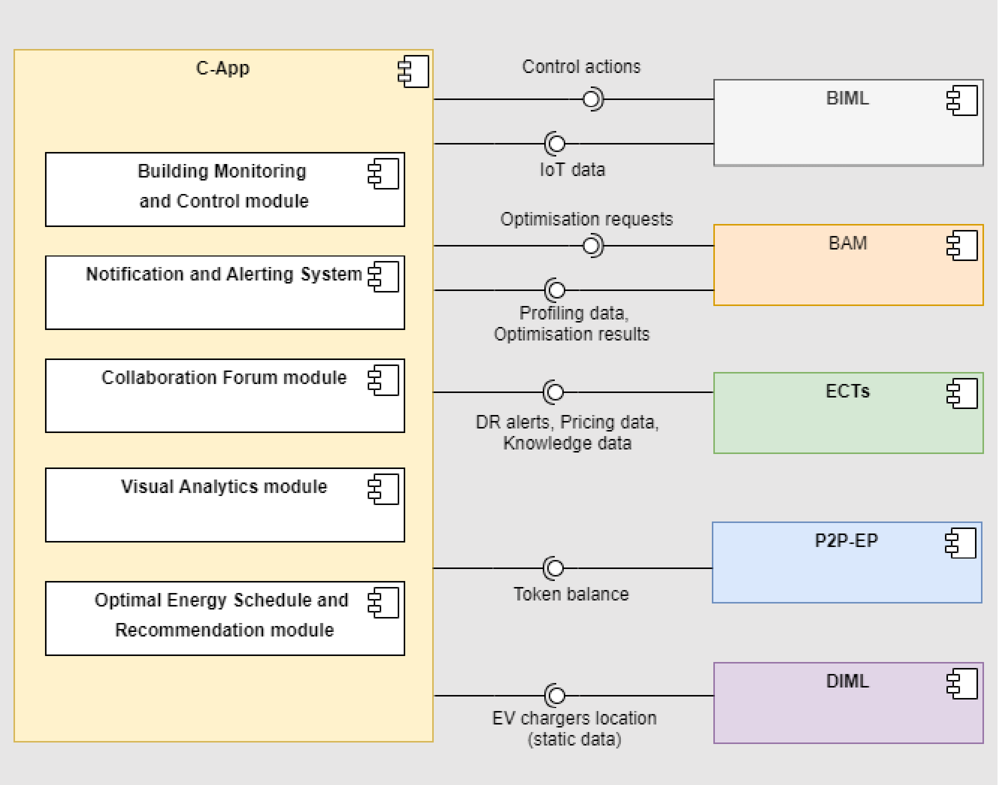
3.6. Citizen Application

The Citizen Application (C-App) includes both analytical and interfacing functionalities directed towards the building end-users. The C-App and its constituent modules are detailed below.

General Information	
Component name	Citizen Application (C-App)
Component Description	C-App is a mobile app composed of four main added-value modules that aim to facilitate citizens engagement and cover energy, smart living, wellbeing, smart living, collaboration,

	mobility, to name but a few, domains, as they are briefly described below: 1. The Building Monitoring and Control module is responsible for processing all necessary information regarding the building facility and the citizen profiling. It is a module that provides a building smart-living experience to the citizen allowing him/her to monitor and control different building assets remotely. 2. The Notification and Alerting system aims to populate and send notifications to the end-users about the operational status of the building, the EV and the energy community, detected anomalies/faults and other topics of interest. 3. The Collaboration Forum module provides a collaboration platform with several engagement tools and services. 4. Visual Analytics module is in charge of processing building IoT data and applying analytical reasoning techniques to enable (a) fast interpretation of past and present situations, (b) identification possible alternative futures and their warning and (c) decision making support. Anomaly detection mechanisms on energy consumption, cost will highlight any outlier cases and provide insight on how to avoid them in the future. 5. The Optimal Energy Schedule and Recommendation module aims to provide optimal recommendations to the end user regarding their energy consumption, energy efficiency, energy cost, EV charging and DR participation.
Interfaces with End-Users	Yes / Consumer - Citizen
Relevant UCs	UC1, UC2, UC3, UC6, UC7, UC8, UC9, UC11, UC14
Lead Developing Partner	CERTH
Programming Language(s)	Python, Angular 9
Deployment Requirements	The application will be deployed on a mobile device; the backend modules will run on an ubuntu server.
Specifications	
Functional Specifications	**Building Monitoring and Control module:** 1. Display IoT data (sensing, metering, and monitoring)
	2. Display EV charging/discharging data
	3. Display user Profiling data (e.g., Comfort bounds, Occupancy, EV charging/discharging)
	4. Enable manual editing of user preferences (e.g., Desired Comfort bounds, desired SoC bounds)
	5. Enable the building's systems remote control through the app
	Notification and Alerting system:
	1. Notify end user about the approximate energy costs
	2. Notify end user about optimal energy recommendations
	3. Notify end user about detected anomalies
	4. Notify end user about DR events/requests
	5. Notify end user about relevant p2p transactions
	6. Notify end user about energy community news/updates
	7. Notify end user about security alerts
	8. Notify end user about estimated time of charging completion
	Visual Analytics module: 1. Apply analytical reasoning techniques
	2. Translate data into a visible form that highlights important features
	3. Display visual analytics data
	4. Display billing data
	Collaboration Forum module:
	1. Allow the user to participate in individual topics of a collaboration forum
	2. Provide a Gamification platform that creates daily/weekly objectives for the user to ensure his/her engagement to the collaboration forum
	3. Incorporate a Reward system with ACCEPT points that could be translated to energy sales

	4. Display a ranking table with the top engaged users according to the ACCEPT points 5. Display a Q&A table that contains general information about the functionality of this platform 6. Display and monitor P2P transactions of the end user	
	Optimal Energy Schedule and Recommendation module:	
	1. Engage the user for potential savings	
	2. Provide insights about the energy cost per appliance / activity	
	3. Provide forecasted estimated costs and recommendations	
	4. Display recommendations of optimized solutions for energy efficiency considering their comfort level.	
	5. Display recommendations of optimal daily/weekly EV charging schedules.	
	6. Display profit/penalties metrics in case that the user follows/deviates from the recommended schedule.	
Functional Dependencies	**Function**	**Responsible Component**
	Retrieve building near real-time and historical IoT and EVs data and send control actions	Building Information Management Layer
	Send optimisation requests and retrieve the results	Building Asset Manager (ODFM)
	Retrieve User Profiling data (e.g., Comfort bounds, Occupancy, EV charging/discharging)	Building Asset Manager (CiDT)
	Retrieve Knowledge, Pricing and DR related data	Energy Community Tools (Retailer, Aggregator)
	Retrieve Token balance data	P2P Exchange Platform
	Retrieve EV chargers' location data	District Information Management Layer
Non-functional Specifications	1. Availability-Component is continuous running	
	2. Performance-Service simultaneous users with a good response time	
	3. Scalability-Possibility to expand the system and avoid adversely affected performance	
	4. Use-ability-Ease of use and user-friendly interface	
	5. Responsiveness-Respond to a user's input or an external event	
	6. Security-Respect and protection of user's privacy and security	
	7. Extensibility- Possibility for new functional requirements addition	
Implementation View		

Component Diagram	 C-App Building Monitoring and Control module Notification and Alerting System Collaboration Forum module Visual Analytics module Optimal Energy Schedule and Recommendation module Control actions IoT data Optimisation requests Profiling data, Optimisation results DR alerts, Pricing data, Knowledge data Token balance EV chargers location (static data) BIML BAM ECTs P2P-EP DIML			
Interfaces/Data Requirements				
Input	**Data**	**From**	**Payload Format**	**Communication protocol**
	IoT data	BIML	JSON	RESTful
	Profiling data	BAM	JSON	RESTful
	Optimisation results	BAM	JSON	RESTful
	DR alerts	ECTs	JSON	RESTful
	Pricing data	ECTs	JSON	RESTful
	Knowledge data	ECTs	JSON	RESTful
	Token balance	P2P-EP	JSON	RESTful
	EV chargers' location (static data)	DIML	JSON	MQTT
Output	**Data**	To	**Payload Format**	**Communication protocol**
	Control actions	BIML	JSON	AMQP
	Optimisation requests	BAM	JSON	RESTful

3.7. Energy Community Tools – ECTs

The Energy Community Tools (ECTs) include all components that interface or support the business cases of the market actors and/or the energy communities.

The ECTs, apart from all the back-end components presented herein, will also be supported by User Interfaces (UIs) and Business Intelligence (BI) Suite, which will provide valuable, data-driven insights on the ACCEPT solution performance in through optimal visualisation aids for the community. In summary, the aforementioned suite will support the following:

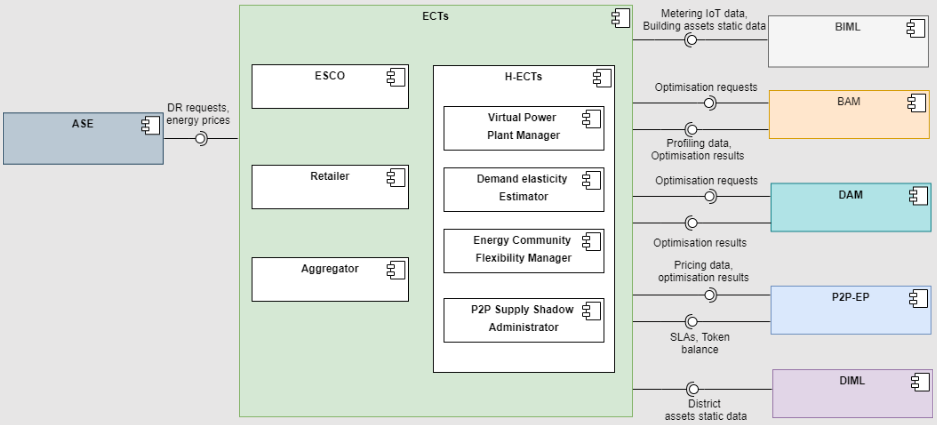
- Provide visualisations, graphs and KPIs, supporting the user (in this case, the energy community as an aggregator, retailer, ESCo) in data-driven decision making
- Support the user in extracting insights and discovering crucial points in speeding up the process and simultaneously improving the performance.
- Support the end users in initiating some of the UCs by exploiting their inputs. For example, the end users can request the optimization of day-ahead retail energy prices etc.

The main components comprising the Energy Community toolset are described below.

3.7.1. Horizontal Energy Community Tools – H-ECTs

General Information	
Component name	Horizontal Energy Community Tools (H-ECTs)
Component Description	The H-ECTs support all market role (i.e., aggregator, retailer and ESCo) specific functionalities of the Energy Community Tools. The H-ECTs consist of four main components: • The Virtual Power Plant (VPP) Manager is responsible for deciding on the optimal clustering of available assets of the community-level portfolio (VPP formulation and configuration) based on dynamic information received from the aggregator, retailer or ESCo tools. This dynamic information will refer to the optimisation context and scenarios required by each actor at a given time to cover community and/or grid needs. • The Demand Elasticity Estimator is responsible for estimating the optimal pricing signal at which prosumers within a community are most likely to respond to and provide flexibility services. • The Energy Community Flexibility Manager is the core community-level optimisation engine responsible for receiving optimisation requests from the aggregator, retailer and ESCo tools and translating them into certain flexibility /DR requests for specific prosumers and assets within the appropriately configured VPP. The responsibility of the specific component includes also the dispatch of such flexibility/DR requests to all relevant prosumers and assets within the community portfolio. • The P2P Supply Shadow Administrator is a tool that will stimulate community optimization through price signals. Taking into account implicit flexibility requests, electricity demand and supply within the community and the overall self-optimization strategy of the community, it will provide price signals to the prosumers in order to steer consumption. The H-ECTs are also responsible for collecting all asset (both at building- and district-level) information (static and dynamic), as well as all flexibility forecasts generated by other ACCEPT solution components.
Interfaces with End-users	None
Relevant UCs	UC4, UC5, UC6, UC7, UC8, UC9, UC10, UC11, UC12, UC13

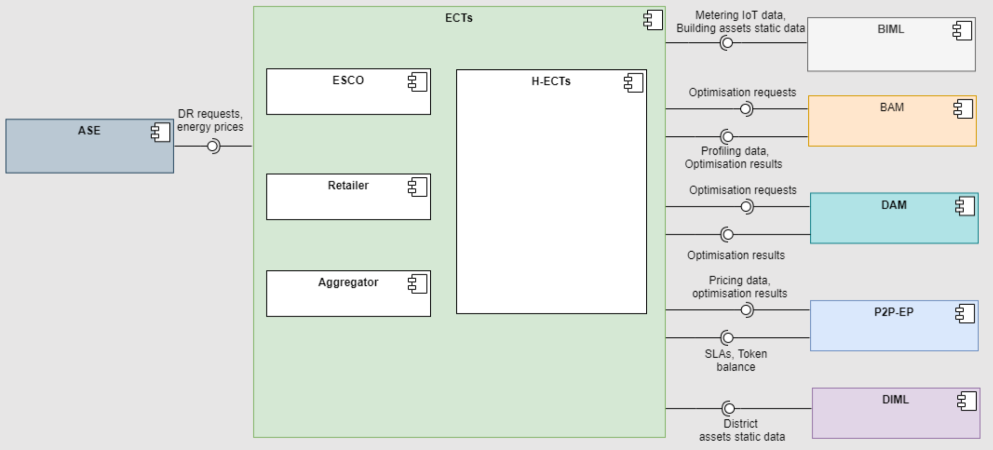
Lead Developing Partner	Hypertech	
Programming Language(s)	Python, Java	
Deployment Requirements	Deployed as a cloud service	
Specifications		
Functional Specifications	**Virtual Power Plant Manager:**	
	1. Receive clustering criteria from the vertical energy community tools (i.e., the aggregator, retailer and ESCo tools)	
	2. Create optimal VPPs for specific optimisation scenario based on the relevant VPP formulation criteria	
	3. Send information on created optimal VPP to the Energy Community Flexibility Manager	
	Demand Elasticity Estimator:	
	1. Create consumers' elasticity regression models	
	2. Estimate portfolio elasticity	
	3. Forecast ideal demand curve at portfolio level and at prosumer level	
	4. Calculate optimal energy retail prices	
	5. Retrieve consumer's tariff data	
	6. Retrieve wholesale market prices	
	7. Monitor performance	
	Energy Community Flexibility Manager:	
	1. Receive flexibility/DR request	
	2. Retrieve optimal VPP information	
	3. Retrieve building-level optimisation results	
	4. Retrieve district-level optimisation results	
	5. Decide on DR requests per available prosumer and/or district-level asset	
	6. Dispatch DR requests	
	P2P Supply Shadow Administrator:	
	1. Deliver energy/ flexibility pricing data	
	Generic functionalities:	
	1. Receive static building-level and district-level assets data	
	2. Receive dynamic building-level and district-level assets data	
	3. Receive building-level and district-level optimisation results (e.g., flexibility forecasts)	
Functional Dependencies	**Function**	**Responsible Component**
	Collect building-level asset specifications	Building Information Management Layer
	Collect district-level asset specifications	District Information Management Layer
	Provision of Total Energy Surplus/Deficit	Energy Community Tools
	Collect wholesale energy price	ACCEPT Solution Emulator
	Collect DR requests	ACCEPT Solution Emulator
	Collection of prosumer flexibility forecasts	Building Information Management Layer
	Collection of district asset flexibility forecasts	District Information Management Layer
Non-functional Specifications	1. Requires a scalable relational database	
	2. Suitable views for filtering assets	
	3. Handling of concurrent requests/ responses	

	4. Stability 5. Security			
Implementation View				
Component Diagram	 ECTs ESCO Retailer Aggregator H-ECTs Virtual Power Plant Manager Demand elasticity Estimator Energy Community Flexibility Manager P2P Supply Shadow Administrator ASE DR requests, energy prices Metering IoT data, Building assets static data BIML Optimisation requests BAM Profiling data, Optimisation results Optimisation requests DAM Optimisation results Pricing data, optimisation results P2P-EP SLAs, Token balance DIML District assets static data			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	DR requests	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Energy prices	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Metering IoT data	Building Information Management Layer	JSON	RESTful
	Building assets static data	Building Information Management Layer	JSON	RESTful
	Profiling data	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation results	District Asset Manager	JSON	RESTful
	SLAs	P2P Exchange Platform	JSON	RESTful
	Token balance	P2P Exchange Platform	JSON	RESTful
	District assets static data	District Information Management Layer	JSON	MQTT
	Data	To	**Payload Format**	**Communication protocol**
Output	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	District Asset Manager	JSON	RESTful
	Pricing data	P2P Exchange Platform	JSON	RESTful

3.7.2. ESCo Tools

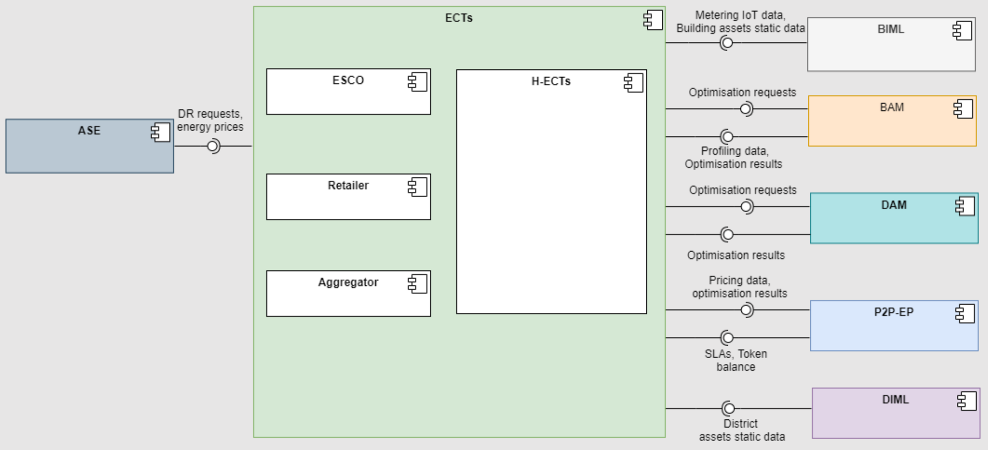
General Information	
Component name	ESCo Tools (ETs)
Component Description	The ETs suite consists of all functional level components addressing the needs of an ESCo (or an Energy Community acting as one). In summary, the tools will implement the following:

	• A portfolio-wide self-consumption (or self-balancing) optimization framework, • communication with P2P supply chain to offer flexibility to third parties, • continuous monitoring of the Energy Performance Contracting KPIs for all portfolio assets. • Amenity-as-a-Service offerings based on Service Level Agreements (SLAs).	
Interfaces with End-users	None	
Relevant UCs	UC5, UC10, UC12, UC13	
Lead Developing Partner	Hypertech	
Programming Language(s)	Python, Java	
Deployment Requirements	Deployed as a cloud service	
Specifications		
Functional Specifications	1. Receive DR request from the ACCEPT solution emulator	
	2. Translate DR request into optimisation scenario	
	3. Optimize community self-consumption	
	4. Optimize community self-balancing	
	5. Monitor building energy performance	
	6. Receive SLAs	
	7. Create compound service offerings for end customers	
	8. Translate optimisation scenario into clustering criteria for VPP manager	
	9. Translate optimisation scenario into optimisation constraints for Energy Community Flexibility Manager	
Functional Dependencies	**Function**	**Responsible Component**
	Flexibility Forecast	Building Asset Manager
	Flexibility Forecast	District Asset Manager
	Retrieve building asset static data	Building Information Management Layer
	Retrieve district asset static data	District Information Management Layer
	Monitor Performance	Building Asset Manager
	Receive DR request	ACCEPT Solution Emulator
Non-functional Specifications	1. Deployed as a cloud service	
	2. One instance per associated stakeholder	
Implementation View		

Component Diagram	 ECTs ESCO Retailer Aggregator H-ECTs ASE DR requests, energy prices Metering IoT data, Building assets static data BIML Optimisation requests BAM Profiling data, Optimisation results Optimisation requests DAM Optimisation results Pricing data, optimisation results P2P-EP SLAs, Token balance DIML District assets static data			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	DR requests	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Energy prices	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Metering IoT data	Building Information Management Layer	JSON	RESTful
	Building assets static data	Building Information Management Layer	JSON	RESTful
	Profiling data	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation results	District Asset Manager	JSON	RESTful
	SLAs	P2P Exchange Platform	JSON	RESTful
	Token balance	P2P Exchange Platform	JSON	RESTful
	District assets static data	District Information Management Layer	JSON	RESTful
	Data	**To**	**Payload Format**	**Communication protocol**
Output	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	District Asset Manager	JSON	RESTful
	Pricing data	P2P Exchange Platform	JSON	RESTful

3.7.3. Aggregator Tools

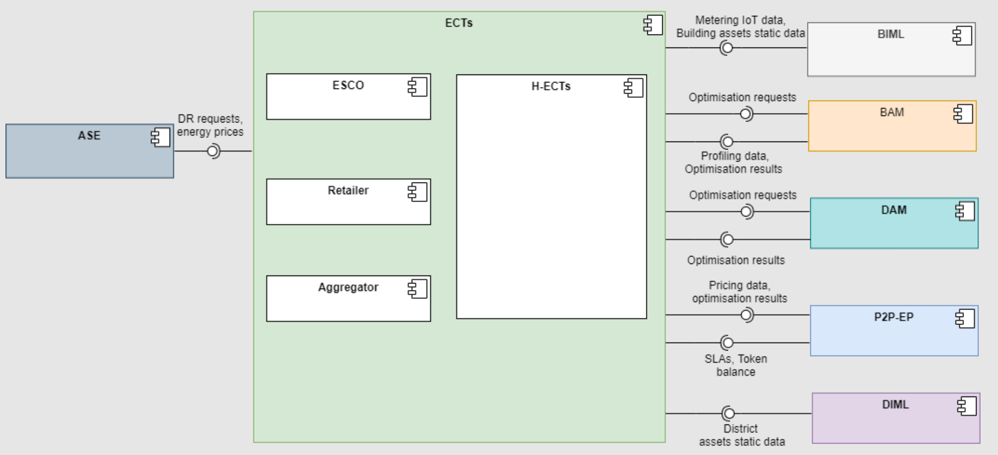
General Information		
Component name	Aggregator Tools (ATs)	
Component Description	The ATs suite of tools includes all the computational engines that support the implementation of the explicit DR scenarios in the ACCEPT project, from the side of the aggregator.	
Interfaces with End-users	None	
Relevant UCs	UC6, UC7, UC8, UC10, UC13	
Lead Developing Partner	Hypertech	
Programming Language(s)	Python, Java	
Deployment Requirements	Deployed as a cloud service	
Specifications		
Functional Specifications	1. Receive DR request from the ACCEPT solution emulator	
	2. Translate DR request into optimisation scenario	
	3. Translate optimisation scenario into clustering criteria for VPP manager	
	4. Translate optimisation scenario into optimisation constraints for Energy Community Flexibility Manager	
	5. Retrieve building flexibility timeseries	
	6. Accumulate flexibility	
	7. Disaggregate requested consumption modification to buildings, EVs, district assets	
	8. Disaggregate requested consumption to different buildings	
Functional Dependencies	**Function**	**Responsible Component**
	Forecast Flexibility	Flexibility Forecast Collection
	Retrieve assets	Community Asset Portfolio Registry
	Monitor event	Portfolio Monitoring and Control Dispatch
Non-functional Specifications	1. Deployed as a cloud service	
	2. One instance per associated stakeholder	
Implementation View		

Component Diagram				
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	DR requests	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Energy prices	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Metering IoT data	Building Information Management Layer	JSON	RESTful
	Building assets static data	Building Information Management Layer	JSON	RESTful
	Profiling data	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation results	District Asset Manager	JSON	RESTful
	SLAs	P2P Exchange Platform	JSON	RESTful
	Token balance	P2P Exchange Platform	JSON	RESTful
	District assets static data	District Information Management Layer	JSON	RESTful
	Data	**To**	**Payload Format**	**Communication protocol**
Output	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	District Asset Manager	JSON	RESTful
	Pricing data	P2P Exchange Platform	JSON	RESTful

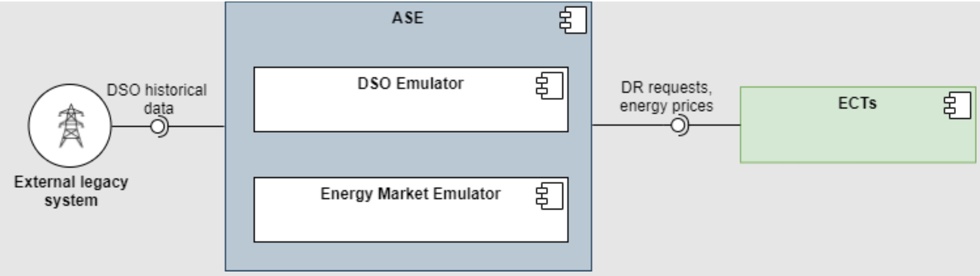
ECTs
ESCO
Retailer
Aggregator
H-ECTs
ASE
DR requests, energy prices
Metering IoT data, Building assets static data
BIML
Optimisation requests
BAM
Profiling data, Optimisation results
Optimisation requests
DAM
Optimisation results
Pricing data, optimisation results
P2P-EP
SLAs, Token balance
DIML
District assets static data

3.7.4. Retailer Tools

General Information		
Component name	Retailer Tools (RTs)	
Component Description	The RTs suite supports the Energy retailer role of the community services, by providing portfolio-wide management for dynamic pricing scenarios, as well as billing information for end customers.	
Interfaces with End-users	None	
Relevant UCs	UC4, UC9, UC11	
Lead Developing Partner	Witside	
Programming Language(s)	Python, Java	
Deployment Requirements	Deployed as a cloud service	
Specifications		
Functional Specifications	1. Receive DR request from the ACCEPT solution emulator	
	2. Translate DR request into optimisation scenario	
	3. Translate optimisation scenario into clustering criteria for VPP manager	
	4. Translate optimisation scenario into optimisation constraints for Energy Community Flexibility Manager	
	5. Estimate portfolio elasticity	
	6. Forecast portfolio demand curve	
	7. Calculate optimal energy retail prices	
	8. Retrieve consumer's tariff data	
	9. Retrieve wholesale market prices	
	10. Monitor prosumer energy consumption	
	11. Generate billing information	
Functional Dependencies	**Function**	**Responsible Component**
	Estimate consumer elasticity	Demand Elasticity Estimator
	Energy consumption measurements for prosumers	Building Information Management Layer
Non-functional Specifications	1. Deployed as a cloud service	
	2. One instance per associated stakeholder	
Implementation View		

Component Diagram	 ASE DR requests, energy prices ECTs ESCO Retailer Aggregator H-ECTs Metering IoT data, Building assets static data BIML Optimisation requests BAM Profiling data, Optimisation results Optimisation requests DAM Optimisation results Pricing data, optimisation results P2P-EP SLAs, Token balance DIML District assets static data			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	DR requests	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Energy prices	ACCEPT Solution Emulator	JSON	AMQT, MQTT or RESTful
	Metering IoT data	Building Information Management Layer	JSON	RESTful
	Building assets static data	Building Information Management Layer	JSON	RESTful
	Profiling data	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation results	District Asset Manager	JSON	RESTful
	SLAs	P2P Exchange Platform	JSON	RESTful
	Token balance	P2P Exchange Platform	JSON	RESTful
	District assets static data	District Information Management Layer	JSON	RESTful
	Data	**To**	**Payload Format**	**Communication protocol**
Output	Optimisation requests	Building Asset Manager	JSON	RESTful or AMQP
	Optimisation requests	District Asset Manager	JSON	RESTful
	Pricing data	P2P Exchange Platform	JSON	RESTful

3.8. ACCEPT solution emulator – ASE (CIRCE)

General Information		
Component name	ACCEPT solution emulator (ASE)	
Component Description	The ASE is the component responsible for emulating the role of external to the energy community energy stakeholders, such as the DSO and the energy market operator. The ASE consists of two sub-components: • The DSO emulator, which is responsible for dynamically assessing the grid state (through appropriate simulations/power flow analysis) and issuing demand response /flexibility requests to the energy communities, i.e., triggering the ACCEPT digital toolset. • The Energy Market emulator, which is responsible for estimating/forecasting and communicating wholesale energy prices to the respective actors within ACCEPT, namely the energy community as an aggregator, retailer and ESCo.	
Interfaces with End-users	Energy community	
Relevant UCs	UC4, UC5, UC8, UC9, UC10, UC11	
Lead Developing Partner	CIRCE	
Programming Language(s)	Java, Python	
Deployment Requirements	Cloud Based. Availability of DSO historical data which can be used for simulation of grid state.	
Specifications		
Functional Specifications	1. Retrieve DSO historical data 2. Estimate electricity grid state 3. Estimate flexibility needs based on electricity grid state 4. Create compound DR request based on estimated flexibility needs for grid constraint alleviation 5. Forecast energy wholesale prices	
Functional Dependencies	**Function**	**Responsible Component**
	Retrieve DSO historical data	External legacy system
Non-functional Specifications		
Implementation View		
Component Diagram	 ASE DSO Emulator Energy Market Emulator DSO historical data External legacy system DR requests, energy prices ECTs	
Interfaces/Data Requirements		

	Data	**From**	**Payload Format**	**Communication protocol**
Input	DSO historical data	External legacy system	JSON	RESTful
	Data	To	**Payload Format**	**Communication protocol**
Output	DR requests	Energy Community Tools (Aggregator Tools, Retailer Tools and ESCo Tools)	JSON	AMQT, MQTT or RESTful
	Energy prices	Energy Community Tools	JSON	AMQT, MQTT or RESTful

4. Conclusions

In the previous pages we detailed the software architecture of ACCEPT’s first prototype system, as well as the specifications of its components. All technical partners associated with developing part of the solution, were asked to fill in a characterization table for each component, providing the functional and non-functional specifications, highlighting interdependencies with other components, and declaring the input/output data requirements.

Additionally, detailed component diagrams were created, and finally, the connection with D2.1 was performed via a mapping of components to Use Cases that required relevant functionalities. The collected material was integrated and two rounds of updates followed, in order to consolidate and bring in line the various inputs. During this process, it was ensured that data and functional specifications between the components were harmonised, and that functionalities offered satisfy the end-user requirements and use cases defined in D2.1. The latter will be further explored in the final version of the project architecture, after engaging with the pilot partners and end users for understanding their requirements. The deliverable also contains a mapping of the high-level architecture to the SGAM model, in order to assist to future integration and development activities.

This is the first version of ACCEPT's system architecture. It will drive the development of the first iteration of the system and its constituent components, up to Month 16 of the project. Following that, a pre-validation stage will evaluate the system prototype and any refinements/changes/updates will be reflected on the Deliverable 2.4, which will present the final version of the architecture.

Annex I – Component's Specifications and Requirements Template

Table 1. Component's Specifications and Requirements Template

General Information				
Component name	*Component name as shown in the system architecture (if modified, changes should be propagated there)*			
Component Description	*Short textual description of the Component (objective, functionalities, etc.)*			
Interfaces with End-users	*End-users interacting with the component if any*			
Relevant UCs	*UCs in which the component will need to be used (provide the ID of each such UC)*			
Lead Developing Partner	*Partner developing the component and its sub-components*			
Programming Language(s)	*Which programming languages will be used for the development*			
Deployment Requirements	*Software and hardware requirements for deploying the solution to the pilot sites*			
Specifications				
Functional Specifications	*The functions that the component needs to offer (Taken out of the shared excel sheet)*			
Functional Dependencies	**Function**		**Responsible Component**	
	Functions that should be provided by other components in order to implement all functionalities listed above		*Component responsible for implementing the respective functionality*	
Non-functional Specifications	*Specifications related to security, performance, reliability, data granularity, scheduling, etc.*			
Implementation View				
Component Diagram	*UML component diagram showing the sub-components of the component and interfaces to other components (Do not skip the second part, it is important)*			
Interfaces/Data Requirements				
	Data	**From**	**Payload Format**	**Communication protocol**
Input	*Input data required by the component to implement a functionality*	*Component that provides the data*	*Format of exchanged information (e.g., JSON)*	*Communication protocol (e.g., HTTP RESTful, AMQP)*
	Data	To	**Payload Format**	**Communication protocol**
Output	*Output data generated by the component as a result of a function*	*Component that requests the data*	*Format of exchanged information (e.g., JSON)*	*Communication protocol (e.g., HTTP RESTful, AMQP)*