Intelligent Assistants for Flexibility Management (Grant Agreement No 957670)

D3.7 Initial Automated flexibility management module

Date: 2021-06-30

Version 1.0

Published by the iFLEX Consortium

Dissemination Level: PU - Public

Co-funded by the European Union's Horizon 2020 Framework Programme for Research and Innovation under Grant Agreement No 957670

Document control page

Document file: D3_7_Initial_Automated_flexibility_management_module.docx
Document version: 1.0
Document owner: **VTT**

Work package: WP3 Artificial Intelligence for forecasting and automated flexibility management
Deliverable type: DEM - Demonstrator, pilot, prototype

Document status: ☒ Approved by the document owner for internal review
☒ Approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Jussi Kiljander (VTT)	2021-02-15	Initial ToC
0.2	Roope Sarala (VTT)	2021-05-25	Initial draft of sections 4 and 5
0.3	Roope Sarala (VTT)	2021-06-16	Updates on section 4 and 5
0.4	Roope Sarala (VTT)	2021-06-17	Updates in sections 2 to 5
0.5	Jussi Kiljander (VTT)	2021-06-18	Updated 3.1, Appendix
0.9	Roope Sarala (VTT)	2021-06-21	Document finalized for internal review
1.0	Roope Sarala (VTT)	2021-06-24	Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Ilias Lamprinos	2021-06-23	Suggestions for content enhancement were made throughout the text.
Christos Krasopoulos	2021-06-21	Accepted with minor modifications and comments

Index:

1 Executive summary

This deliverable presents the first iteration of the Automated Flexibility Management (AFM) module, a part of the iFLEX Framework for phase one. AFM module is responsible for evaluating flexibilities to market/aggregator module and optimizing energy management of the building. The goal for this deliverable is to provide initial specifications of the AFM module, interfaces with other modules and insights into different optimal control approaches.

The work here presents a detailed look on the architecture of the AFM module and how it interfaces with adjacent software modules along with initial implementation specification for a pilot site. In addition, this deliverable discusses the different optimal control techniques in the context of Artificial Neural Networks (ANN), also presenting a hierarchical, three-layered control strategy, starting from the building automation system up to the AFM module.

Internally, the AFM module is split into three software modules: *Planner, Optimizer* and *Resource*. *Planner* is responsible for producing flexibility estimates, and optimized energy consumption plan. It does this by utilizing *Optimizer* module that hosts optimization algorithms for all AFM needs. Finally, *Resource* module provides an interface with the models in Digital Twin repository and additionally handles second-level control of the resources. In most cases, there are multiple *Resource* instantiations within the AFM, each for a different resource of the building.

2 Introduction

2.1 Purpose, context and scope

The purpose of this deliverable is to document the first phase results of the task *3.4 - Automated decision-making and energy optimization*. The goal of the task is to develop a component capable of automated control and planning of energy systems, utilizing digital twins of the energy systems and consumers. This is achieved by using model-based deep reinforcement learning combined with model predictive control approach. Moreover, the control strategy involves multiple layers of control, including rule-based control to ensure safe and robust control.

In phase one, the focus is to present an initial specifications for the AFM module, interfaces with other modules together with a reliable and well-documented initial codebase with the necessary implementation for piloting and further development. This includes initial implementations of flexibility evaluation, control and also interface specifications with other software modules. Evaluating flexibility and control is done via heuristic approach whereas the energy consumption optimization is not considered at this phase.

2.2 Content and structure

This deliverable is structured as follows:

- Section 3 provides an overview, mapping the contents of the deliverable to the use cases and to the iFLEX architecture.
- Section 4 introduces the main methods and approaches applied in the work.
- Section 5 details the implementation of the Automated Flexibility Management module, outlining the architecture, interfacing and real world instantiation of such system.
- Section 6 concludes the deliverable.

3 Overview

3.1 Relation to use cases

The use cases of the iFLEX project are documented in D2.1 - Use cases and requirements. Based on the use cases and system level requirements, component specific requirements have been defined for different functional components of the iFLEX Framework. Four high level requirements have been specified for the Automated Flexibility Management (AFM) component documented in this deliverable. The requirements are related to all three of the high level requirements. The following list introduces the AFM specific requirements and maps them to the Primary Use Cases (PUC):

- FN-AFM-01 - Provide baseline forecasts, related to PUC-8
- FN-AFM-02 - Flexibility potential, related to PUC-8
- FN-AFM-03 - Activate offered flexibility, related to PUC.9
- FN-AFM-04 - Optimize flexibility based on prices (implicit demand response), related to PUC-9

The requirements in iFLEX are managed via Jira tool that provides methods for creating, prioritising, secluding and monitoring requirements. Figure 1 illustrates how the requirements are managed in Jira. The full list of current requirements are presented in the Appendix.

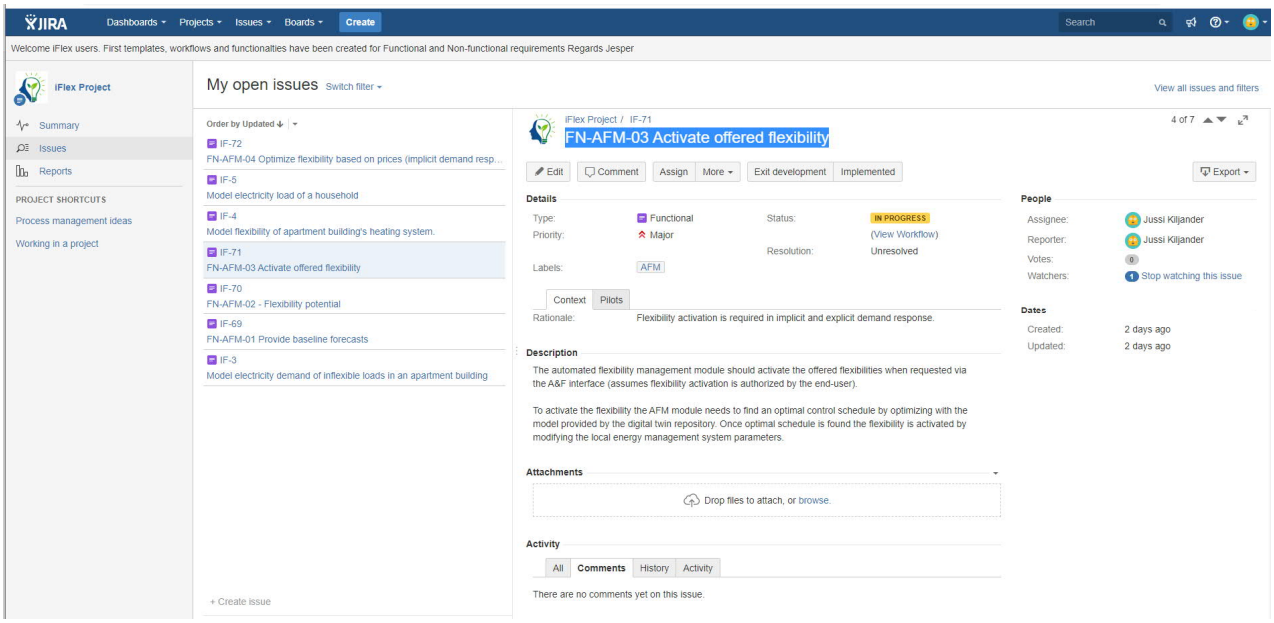

Figure 1: AFM requirements captured in the project's Jira tool.

3.2 Relation to the functional architecture of the iFLEX Framework

Automated Flexibility Management module is at the centre of the iFLEX assistant framework, as is depicted in Figure 2. The main responsibilities of the module are:

- Forecasting on status and energy consumption of the building.
- Evaluation potential flexibilities on both electricity and district heat vectors.
- Optimization building demand-response with respect to energy price, CO2 emissions etc.
- Production of control signals to BEMS/HEMS according to the activated flexibility or optimized load plan.

Forecasting is done by utilizing Digital Twin repository, which contains, as the name implies, all models related to the system. These models provide forecasts on energy loads, flexibility and response of flexible assets with respect to various control inputs. The models are documented in detail in *D3.1 - Initial Hybrid Modelling Module.* Communication with BEMS/HEMS is done via Resource Abstraction Interface (RAI), which is documented in *D4.1 - Initial Resource Abstraction Interface*. This interface also provides access to external data sources, such as weather and CO2 emissions data. Additionally, the AFM module communicates with the end-user (through end-user interface, see *D3.4 - Initial Natural User Interfaces*) for receiving user-defined comfort and/or consumption preferences as well as request approval on changes to baseline consumption. Finally, activation of flexibilities is received from Aggregator and market interface, documented in *D4.4 - Initial Market and Aggregation Interface Module*.

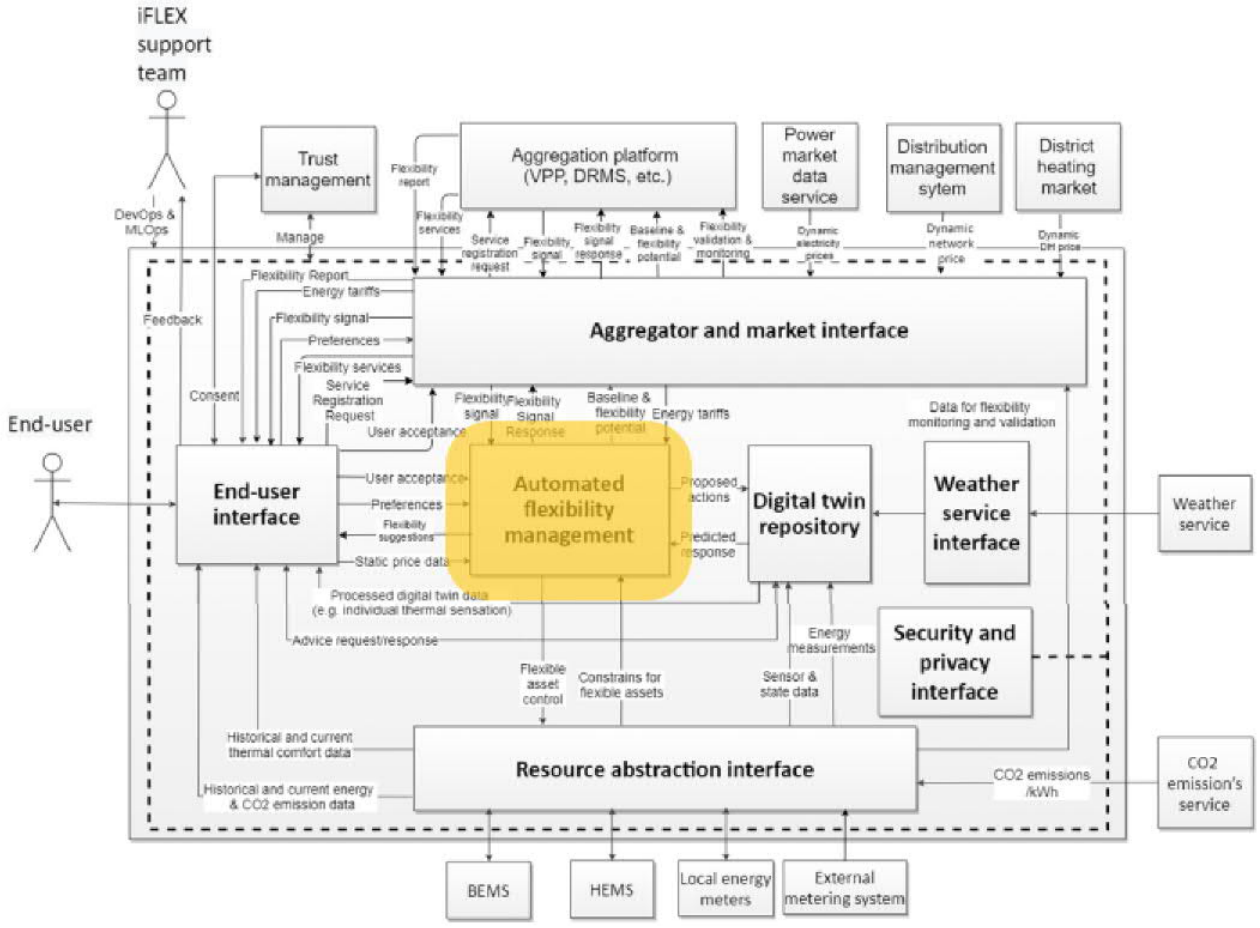

Figure 2. Functional view of the iFLEX assistant with Automated Flexibility Management module highlighted.

3.3 First phase focus

First phase focus of AFM module implementation is to provide a simple but functional software that is able to provide baseline consumption forecasts, evaluate flexibilities and produce control signals in order to follow previously created load plans. In phase 1, control signals are optimized using brute force method, where every possible outcome is evaluated and the most suitable among them is selected. In addition, no implicit demand response optimization is done and the target is to follow baseline consumption load (forecasted in advance) as accurately as possible without sacrificing user comfort. Flexibilities are evaluated using a heuristic search algorithm as brute force method is infeasible due to large amount of possible system states.

4 Methodology and approach

4.1 Control architecture

In iFLEX framework, the control architecture can be divided into three hierarchical levels, depicted in Figure 3. From highest to lowest (in terms of abstraction), these levels are:

- Building / Apartment level controller.
- Individual resource controller in Automated Flexibility Management module.
- Resource's internal controller, located at the site.

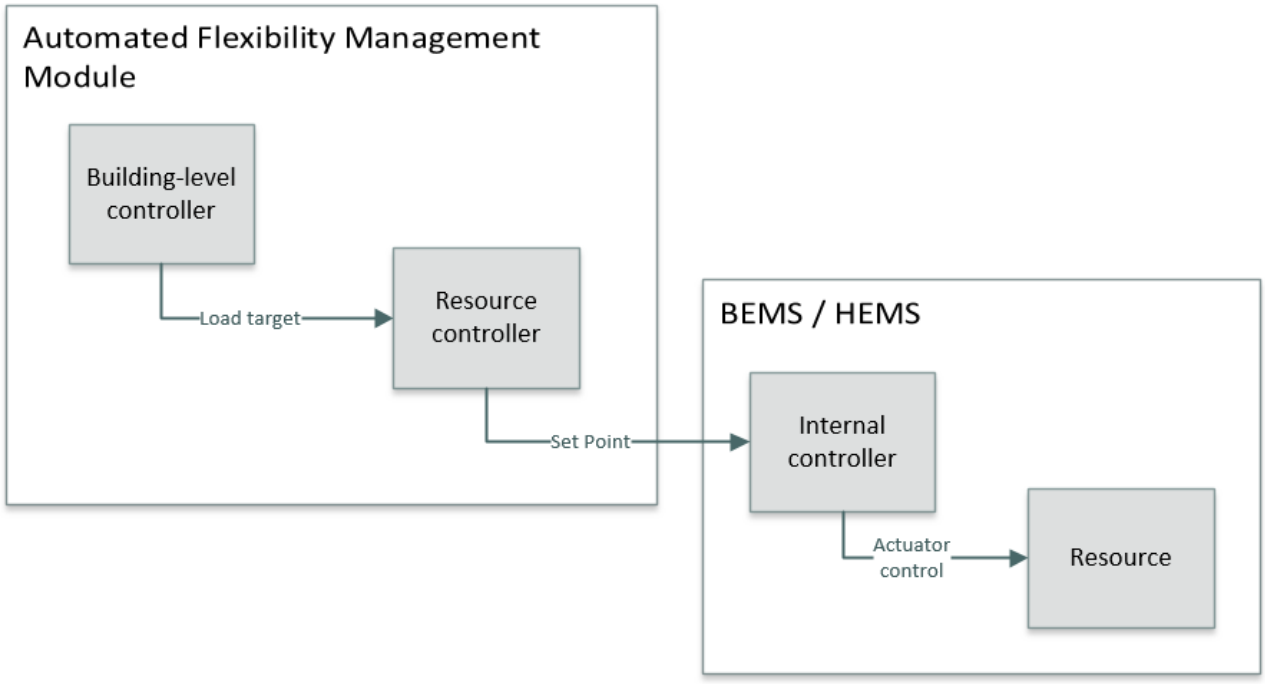

Figure 3. Different control levels in iFLEX framework.

First, and the lowest-level control is done at the resource level. It ensures the normal operation, i.e. operating range, of the system and also translates more high-level control (e.g. temperature set point) signals into actuator control signals. An example of this could be the controller inside a heat pump that turns the pump on and off while maintaining a desired temperature. In addition, it will maintain its parameters in operational range even by overriding incoming control signals if necessary. Typically, a Proportional-Integral-Derivative (PID) controller is used, which uses feedback mechanism to minimize error between desired set point and measured process variable.

On top of that, at this level also operates the existing rule-based control system, developed by human experts, that serves not only as a baseline control plan but also ensures safe and reliable operation of the system in case AI-based control is unsure of its decision or is otherwise compromised.

Next level of control is implemented at iFLEX framework level, inside AFM module controlling only single resource. The objective of this controller is to translate the load trajectory target of the optimized plan into control set points sent to the RAI. Depending on control frequency and resource dynamics, this can be done either via traditional reactive control mechanism but typically a more complex predictive control is needed. If the dynamics of the resource are known or modelled, like in our case (in Digital Twin repository), we can use them to simulate possible scenarios and choose the optimal one. The optimization problem presented here can be labelled as a load following problem, which in standard optimization problem form can be presented as

$$\min_{a_1,\dots,a_T} \sum_{t=!}^{N} \left(E_t - \widehat{E_t}\right)^2$$
$$s.t.\ \ s_t = f_f(s_{t-1}, a_{t-1})$$
$$s_{min} \leq s_T \leq s_{max}$$
$$\widehat{E_t} \in s_t,$$

where E_t is the energy in the load plan, $\widehat{E_t}$ is the energy consumption predicted by the model f_f, and s_t is the state of the system including the energy consumption and user comfort. This can be solved in many ways, for example if f_f is a continuous function, convex optimization algorithms such as quadratic programming can be used (Bianchini, Casini, Vicino, & Zarrilli, 2016)(West, Ward, & Wall, 2014). However, if the control horizon is short and control space limited, a brute force (guess and check) method will suffice.

4.2 Control algorithms for energy optimization

One more layer of control is needed for global optimization of the system. To achieve this, two approaches can be recognized, namely *model-free* and *model-based* optimal control. In model-free control, as the name implies, there is no dynamics model to simulate states from. A popular technique is to use *reinforcement learning* in which the agent learns to control the system through interaction with the environment, bearing similarities with human learning. With the rise of neural networks, the agent used today is typically a neural network, in which case the technique is labelled as *deep reinforcement learning (DRL)*. Among different DRL algorithms, the most popular for DR control is *Q-learning* (Sutton & Barto, 1998), which has been battle-tested in many scenarios (Chen, Norford, Samuelson, & Malkawi, 2018; Patyn, Ruelens, & Deconinck, 2018; Ruelens et al., 2017). In Q-learning, the agent learns a function, labelled as Q-function, that estimates the *value* of an action in a given state (Figure 4). It does this by exploring the system by doing different actions and observing different reward-action pairs and then updating the value function using the following equation iteratively:

$$Q^{new}(s_t, a_t) \leftarrow Q^{old}(s_t, a_t) + \alpha\left(r_t + \gamma \max_a Q(s_{t+1}, a) - Q^{old}(s_t, a_t)\right),$$

where s_t is the state, a_t is the action, α is the learning rate and γ is the future rewards discount factor. In essence, the new value of Q-function is obtained by adding the total net reward scaled by the learning rate to the previous value.

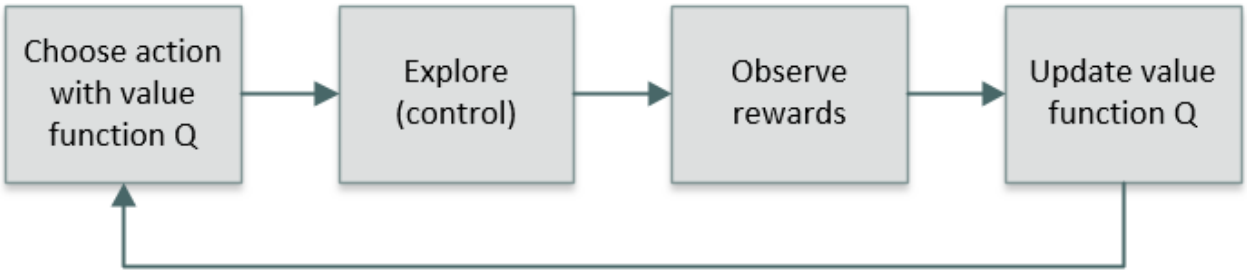

Figure 4. Q-learning algorithm.

In contrast to model-free algorithms, model-based control techniques rely on dynamics model, which is used to simulate different control scenarios. This optimal control problem for DR can be presented in general level as (Kiljander et al., 2021)

$$\max_{a_1,\dots,a_T} \sum_{t=1}^{T} r(s_t, a_t)$$
$$s.t.\ s_t = f_f(s_{t-1}, a_{t-1}) + f_g(s_{t-1,}) + f_d(s_{t-1,})$$
$$s_{min} \leq s_t \leq s_{max},$$

where *r* is the reward function, s_t is the state of the system, and a_t is the action. The s_{max} and s_{min}represent possible constrains such as the minimum and maximum values for indoor temperature. The f_f, f_g and f_d represent Artificial Neural Network (ANN) models for flexible resources, power generation and inflexible

demands, respectively. The dynamics models f_f, f_g, f_d can be any functions, from simple heuristics to deep neural networks. However, the choice of the models also dictate the possible algorithms that can be used to solve the problem. Generally, the optimization algorithms can be divided into *gradient-based* and *gradient-free* methods. Gradient-based methods are usually faster and also can (mathematically) guarantee an optimal solution, but are notoriously difficult to use with neural network models. This is mainly due to the exploding and vanishing gradient problem (Pascanu, Mikolov, & Bengio, 2012). Gradient-free methods, such as genetic algorithms and particle swarm optimization (Kusiak, Tang, & Xu, 2011; Ma & Wang, 2011; Wang & Jin, 2000), do work well with neural networks and are used extensively with them but lag on accuracy and efficiency compared to gradient-based methods. In addition, model-based optimal control with neural networks typically employs a *Nonlinear Model Predictive Control (NMPC)* approach, a form of closed-loop control, where the control algorithm uses feedback from the system when making new decisions, as is shown in Figure 5. In practice, this means that a new optimized plan is done at every time step with the latest information available, compared to doing it only once per period. An upside of this approach is that it leads to more optimal control with a trade-off in plan predictability.

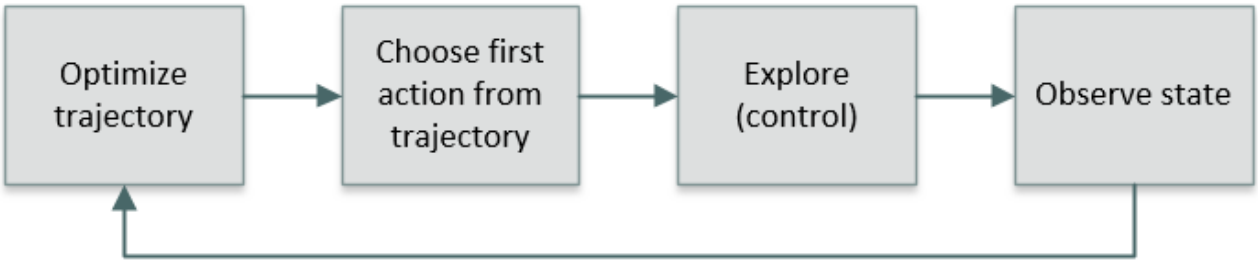

Figure 5. Model predictive control (MPC) method for model-based optimal control.

5 Implementation

5.1 Overview

The AFM module documented here is implemented using Python programming language. Code implementation follows object-oriented programming paradigm and is overall designed with expandability in mind. Interfacing with other modules is done via MQTT protocol, using the Eclipse Paho MQTT Python client library. Data between software modules is transferred in Pandas DataFrame format, serialized to JSON notation. Various run parameters are specified in a separate configuration file, where data polling, flexibility evaluation, planning and control intervals are set as well as paths and MQTT topics are configured.

5.1.1 Software architecture

Internally, AFM is split into *Planner*, *Optimizer*, and *Resource* components, as seen in Figure 6. Each of the three components are implemented as individual modules, though they are all contained in a single package. The roles of the components are briefly described in Table 1.

Table 1. Roles of the components inside automated flexibility management module.

Component	Responsibilities	Data store	Interfacing
Planner	Evaluate flexibility Send control commands to Resource component Send flexibility potential to market interface Handle user communication Handle market communication	Baseline plan Optimized plan User preferences Market constraints Flexibilities	Market End-user interface Optimizer Planner RAI
Optimizer	Optimize energy consumption Optimize path following Calculate maximum flexibility		Planner Resource
Resource	Translate load target into control commands Baseline consumption forecast Send control commands to RAI	Optimized control plan Resource state	Digital Twin repository RAI Optimizer Planner

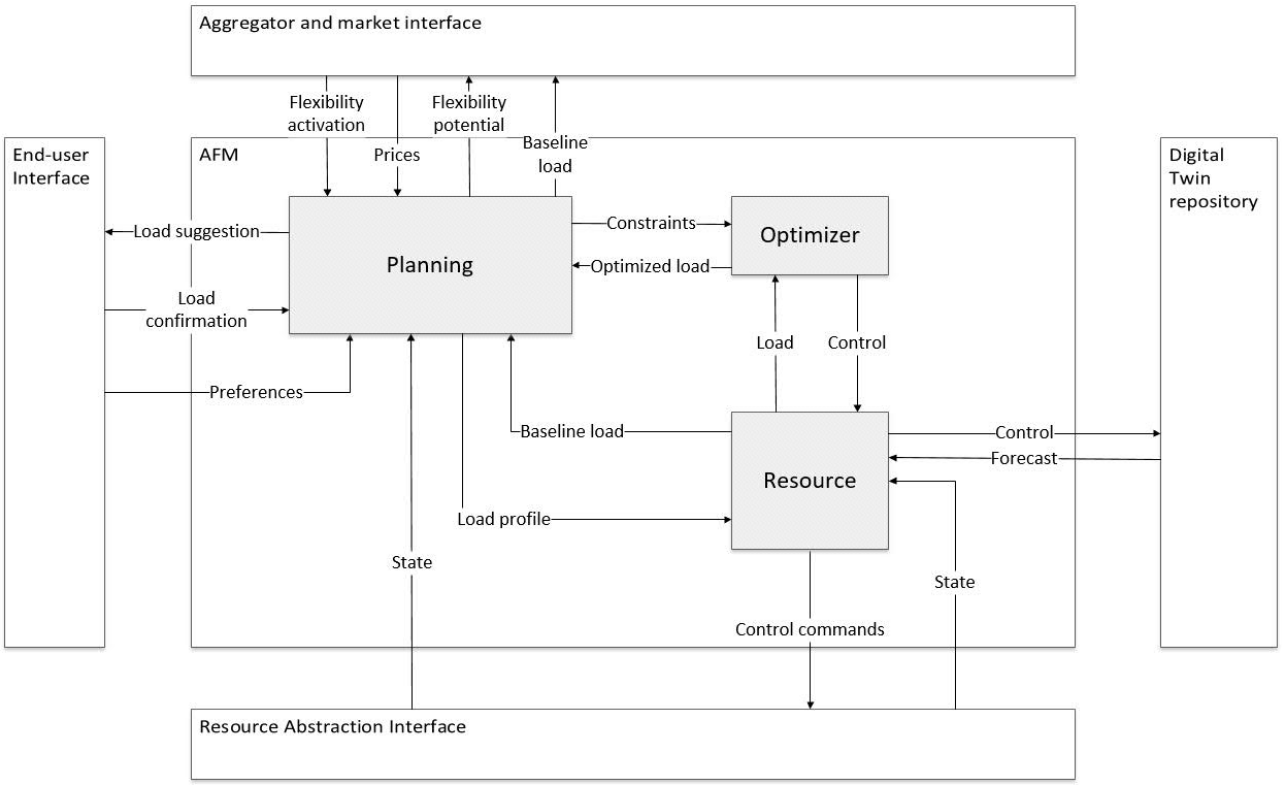

Figure 6. Architecture of the Automated Flexibility Management module.

In Figure 7, another perspective to the architecture is given. It maps the software components to the core functionality of the module. The arrows aim to show what components are the main contributors in each of the functionalities, for example the *Resource* is involved in evaluating flexibilities but only as a messenger between *Optimizer* and Digital Twin repository. Thus, there is no arrow from *Resource* to Flexibility in the figure below.

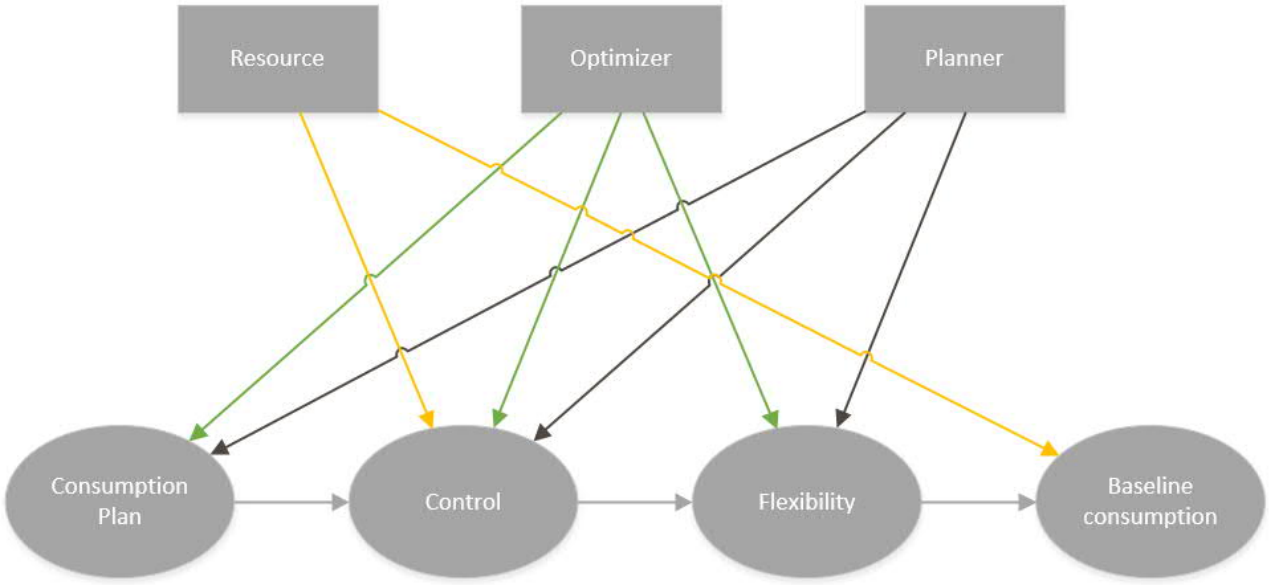

Figure 7. Mapping of the components to core functionality in the automated flexibility management module. The arrows represent the most important components in producing each of the function.

5.1.1.1 Evaluating baseline consumption and flexibility

Flexibility in this context can be described as the potential to deviate from each time step's *baseline consumption.* Baseline consumption is defined as the energy consumption what would ensue if no control is performed. In essence, the site will then be fully controlled by the automation system located on site. The amount of flexibility is the maximum deviation from the baseline consumption in each time step, i.e. more formally

$$-F_i = C_i^{base} - \max(C_i^{ctrl}),$$

where F is the flexibility, C^{base} is the baseline consumption and C^{ctrl} is the consumption obtained through some control inputs. Max function denotes that we could have different consumptions using different controls and i refers to the time step we are evaluating. For simplicity reasons, we only consider one flexibility value in each direction (up or down) in each time step. Having multiple values would be exponentially harder to evaluate and also is (possibly) more confusing for the end user to understand. Even still, calculating the maximum amount of flexibility for each time step is not trivial. If we want to evaluate all possibilities, we would need to calculate $\sum_i^r n^i$ options, where *n* is the number of control options, *r* is the length of the forecast and i is the current time step. In practice this exponential growth implies that for any forecasts longer than a couple of steps it is impossible to use brute force (calculate all possible trajectories), so instead a heuristic or algorithm is used to reduce search space. One option is to only apply control in the step that is being evaluated while leaving the other as is, i.e.

$$C_i^{ctrl}(t) = C^{base}(t \neq i) + C^{ctrl}(t = i),$$

where C_i is the consumption at time step *i*. This approach reduces the number of iterations from $\sum_i^r n^i$ to $r * n$, which is much more feasible to calculate in short time.

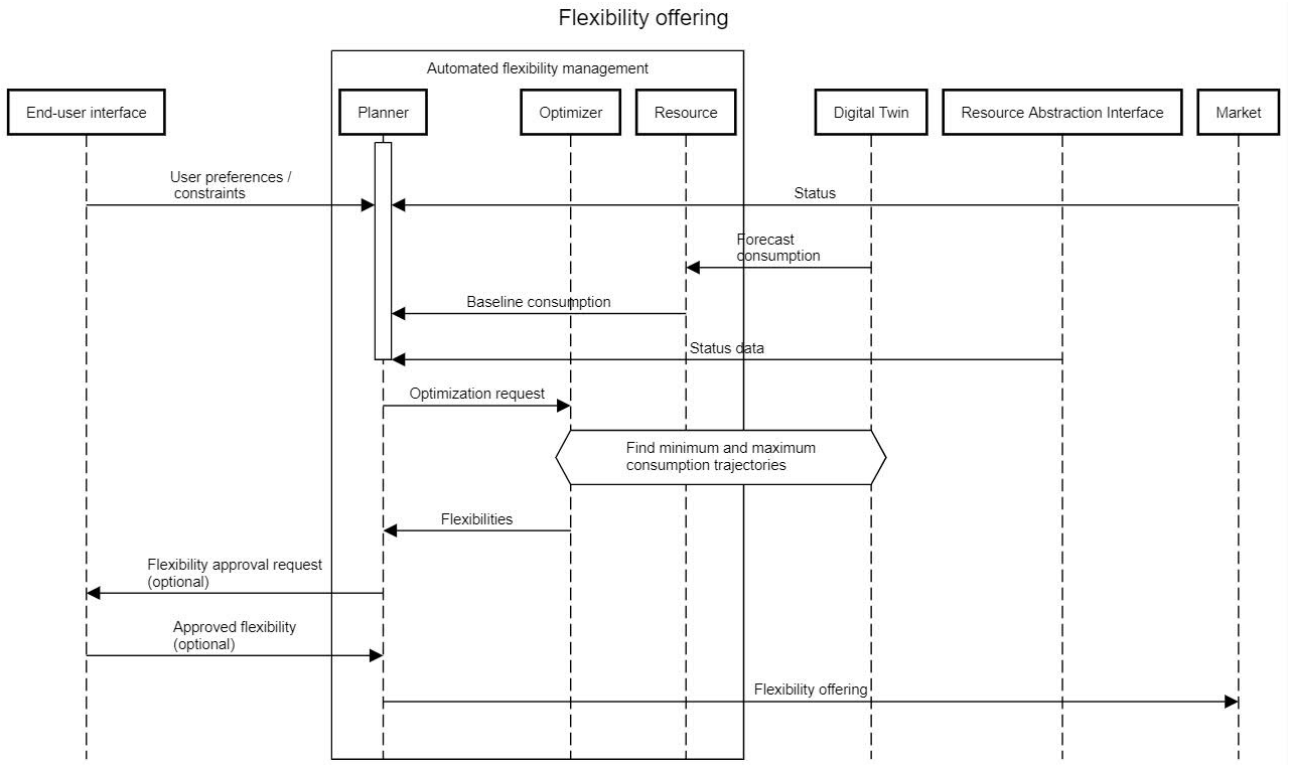

Figure 8. Sequence diagram of the flexibility offering process.

Figure 8 depicts the flexibility evaluation process that begins on planning module by polling the baseline consumption from each resource. This is done using the internal dynamics model of each resource in the Digital Twin repository. Next, the planning module aggregates the data into *baseline plan* and after adding any constrains (from end user, market or other) sends data to the optimizer module. The optimizer sends back now the *optimized consumption plan* which the planning module forwards to the end-user. The end-user then

approves or rejects the plan. This step can be skipped if user has authorized the iFLEX assistant to do so. Finally, flexibilities are then sent to the market module.

Once a flexibility is activated in the market module, the information is sent to the *Planner* in AFM module. After confirming activation, *Planner* triggers first the re-optimization of load plan with now additional flexibility constraint. After that the flexibility offering process has to be reinitiated. The sequence diagram of this process is presented in Figure 9.

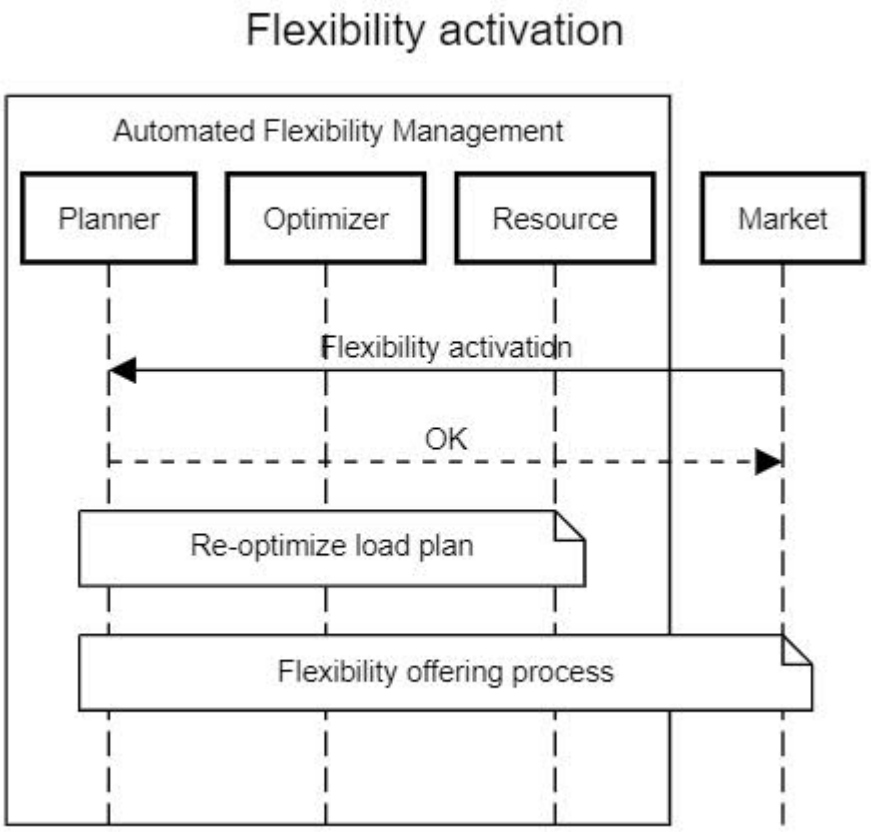

Figure 9. Sequence diagram of flexibility activation process.

5.1.1.2 Plan optimizer

In the first phase of the pilot, the optimized plan is defaulted to the baseline plan created beginning of each period (e.g. a day) and is not updated again during the period. However, as the forecasted plan will not be completely accurate, it will necessarily differ from the actual consumption and thus committing to the plan requires quite a bit of controlling. In addition, there is already value at simply controlling to the (forecasted) baseline plan; in many cases differing from previously planned consumption trajectory incur costs.

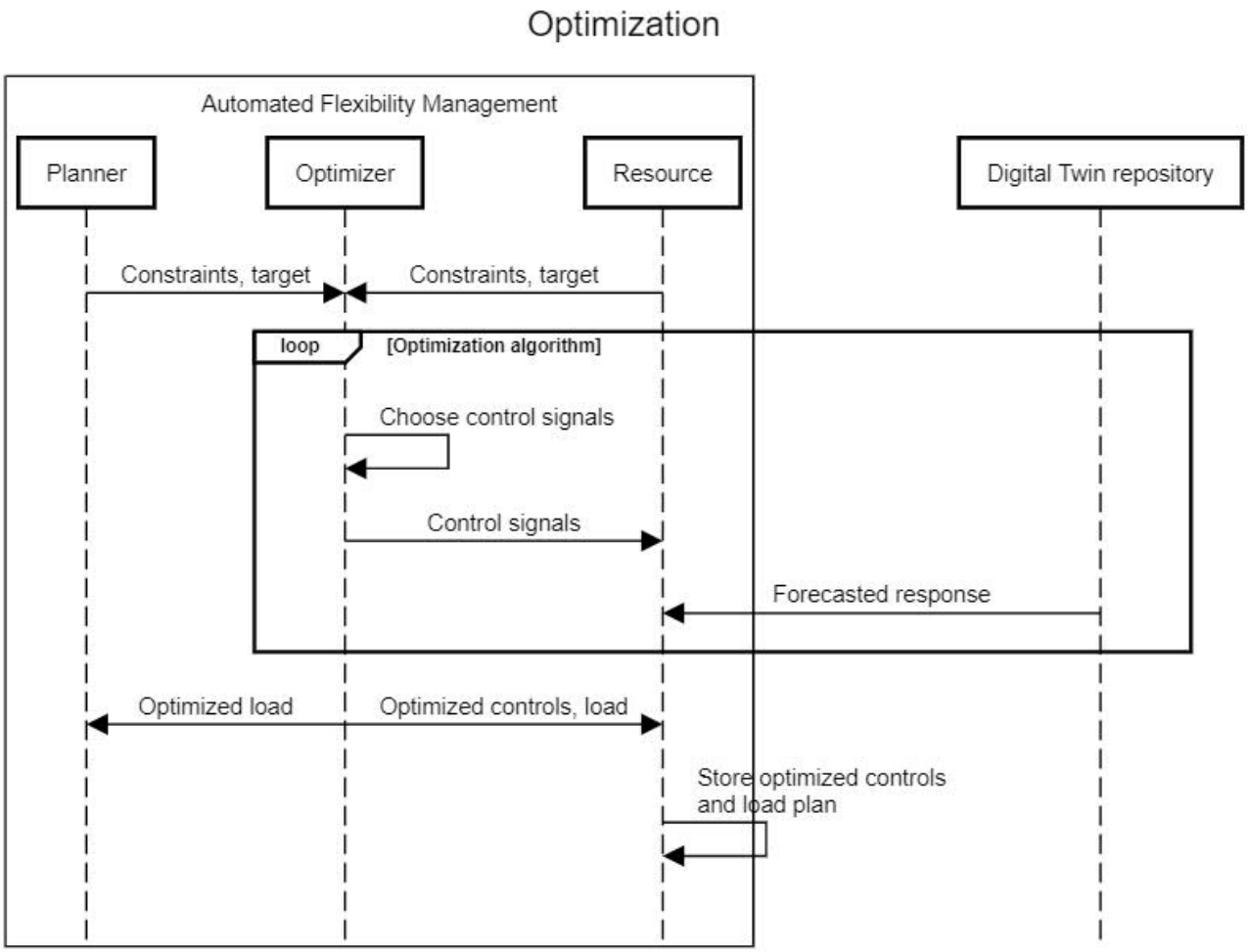

Figure 10. Sequence diagram of the optimization process.

The general sequence of the optimization process is depicted in Figure 10. All optimization processes (energy, flexibility, control) all follow the same template. The core idea is that the optimizer is stateless; it is the responsibility of the *Planner* or *Resource* to format the problem in correct way and ensure constraints are up to date. In later phases of the project, it will be possible to move the *Optimizer* component to a separate machine, for example if more computing power is needed.

5.1.1.3 Control

Control is performed at set intervals and uses MPC-type closed loop control. In phase one, a brute-force search algorithm is used. This means, that at each time step, all possible control options are evaluated by simulating them and then the most suitable is selected for control.

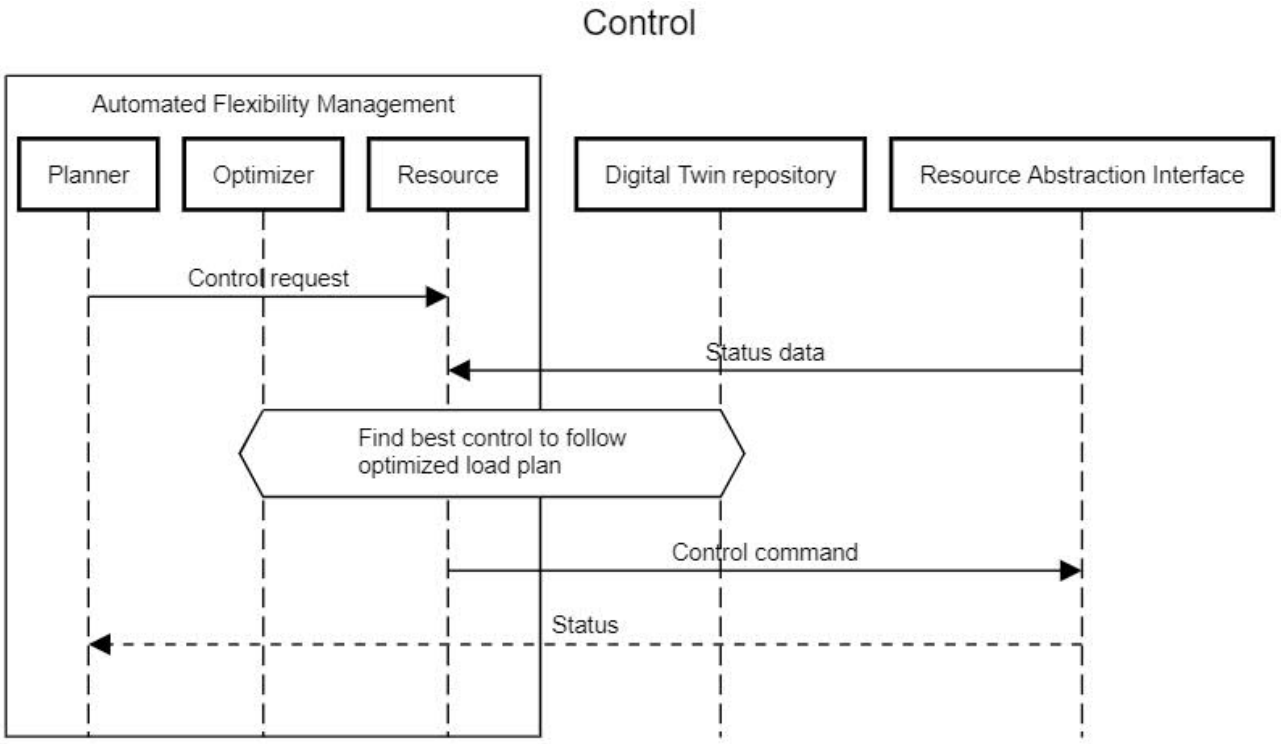

Figure 11. Sequence diagram of control process

The process begins when *Planner* sends load target for next step to the *Resource* (Figure 11). Next, all possible control scenarios (set point values) for the next step are evaluated using models in the Digital Twin repository and algorithms in the *Optimizer* module. The best one is chosen and this information is sent to the RAI module and to the Planner. A message is sent back from the RAI confirming that control was set successfully. Confirmation message is important, as it is quite possible that for some reason the rule-based control system in the BEMS/HEMS does not accept the control request. In that case the rejection, indicated in the status message is followed up retrieval of true control value via RAI.

5.1.2 Interface provided by the Automated Flexibility Management module

The current implementation of the AFM module can be interfaced via MQTT. The interface specification for phase one includes only the necessary communication (i.e. consumption and flexibility information as well as flexibility activation) and will be completed in later phases of the project. The message payloads are serialized with JSON. Table 2 describes to MQTT topics of the AFM interface.

Table 2. MQTT topics for the AFM component interface

Topic	Description	Method	Payload
<assistantId>/<energyVector >/baseline	Topic for receiving notifications of baseline load profiles of the prosumer/consumer.	Subscribe	A
<assistantId >/<energyVector>/ flexibility	Topic for receiving notifications of flexibility potential of the prosumer/consumer.	Subscribe	B
<assistantId>/<energyVector>/activate	Topic for activating offered flexibilities by publishing changes to load plans.	Publish	A

<assistantId>/<energyVector>/load	Topic for receiving load (consumption - production) notifications from the prosumer/consumer.	Subscribe	A

The Payload A consists of a time series of loads (kW). The time format can be either Epoch (UNIX time) or ISO 8601. The loads are represented in kW. When the Payload A is used in baseline and load topics, a positive value corresponds to the demand and negative to the production from the grid. In activate topic, a negative value indicates reduction of the baseline load. An example of the Payload A is represented in Figure 12. This format was selected as it can be easily extended with new parameters. It is also directly supported by Pandas data frame JSON serialization methods.


```
[
        {"time":1518922800000,"load":171.0},
        {"time":1518926400000,"load ":162.0},
        {"time":1518930000000,"load ":161.0},
        {"time":1518933600000,"load ":161.0},
        {"time":1518937200000,"load ":171.0}
]
```


Figure 12. Example of Payload A.

The Payload B is reserved for the flexibility offers. It is a time series that consist of records with three parameters: time, flexibilities, and expiration time. Flexibilities parameter includes a list of flexibilities (up and down). Negative values indicate reduction to the baseline load and vice versa. The expiration time parameter specifies for how long the flexibility offer is valid. Figure 13 presents an example of the Payload B.


```
[
        {"time":1518948000000,"flexibilities":[-101.3648666667],"expiration_time":[1518947940000]},
        {"time":1518951600000,"flexibilities":[-97.098], "expiration_time":[1518947940000]},
        {"time":1518955200000,"flexibilities":[-96.2152],"expiration_time":[15189479400000]},
        {"time":1518958800000,"flexibilities":[-95.2588333333],"expiration_time":[1518947940000]}
]
```


Figure 13. Example of Payload B.

5.2 Instantiation

5.2.1 HVAC control at apartment building level for explicit demand response

The apartment building consists of 90 residential apartments, each monitored for thermal comfort and air quality. All apartments share infrastructure for heating (district heating) and domestic water. In addition, other notable (consumption related) infrastructure includes an elevator and a common sauna. The building is also equipped with a building automation system, including a building energy management system (BEMS).

Controllable flexible assets for phase 1 include space heating, which is primarily provided by the district heating network, an exhaust heat pump for additional heat production and controllable ventilation power. Heating assets can be fully controlled between 0-100%, and is achieved by controlling various control points, such as valves and heat exchangers. Ventilation cannot be completely turned off, thus its controllable range is set between 30-100%. Detailed documentation of these control points are provided in *D4.1 - Initial Resource Abstraction Interface*.

A simplified overview of the energy flows in the building is presented in Figure 14. In short, incoming heating energy is coming from the district heating network and from electricity that is used to heat apartment air. Within the system is an exhaust heat pump that captures energy from exhaust air back into apartment heating. Additionally there is a water boiler in the system.

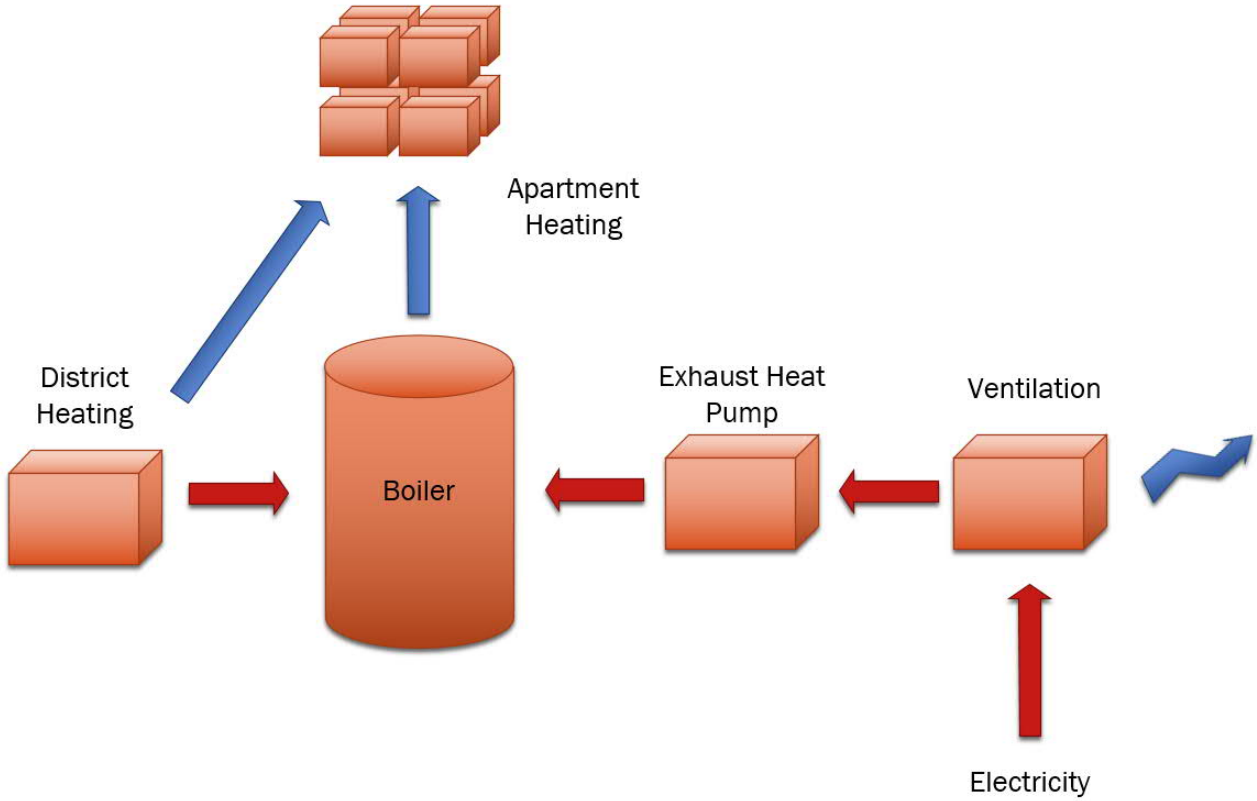

Figure 14. Simplified view of heating energy flows in the pilot building. Red arrow represent energy inflows while blue arrows indicate energy outflows.

A good selection of measurements are also available from the building. As with the control points, the full list of measurements are found in deliverable *D4.1*, but the most important are listed below:

- Building level electricity consumption (at high frequency, <1min)
- District heating consumption (at high frequency, <1min)
- Indoor air quality comprising of temperature, humidity and CO2 measurements
- Local weather data
- Status information on all relevant BEMS system variables (e.g. valves, pumps, fans)

In practice, controlling heating and ventilation is done via actuating many components in the system, such as valves, pumps and heat exchangers simultaneously. To simplify control options, these controls are bundled into different control “modes” which are designed by human experts and stored in the RAI. This limits possible control space (easier to compute) and ensures that each control command is meaningful. Still, it enables many combinations of controls, including but not limiting to examples in Table 3.

Table 3. An example of some of the control modes in the pilot building.

Control mode description	Target effect
Minimize / Maximize all heating	Decrease / Increase consumption
Minimize district heating, boost exhaust heat pump	Use more electricity compared to district heating
Change radiator network set point temperature	Adjust consumption up or down

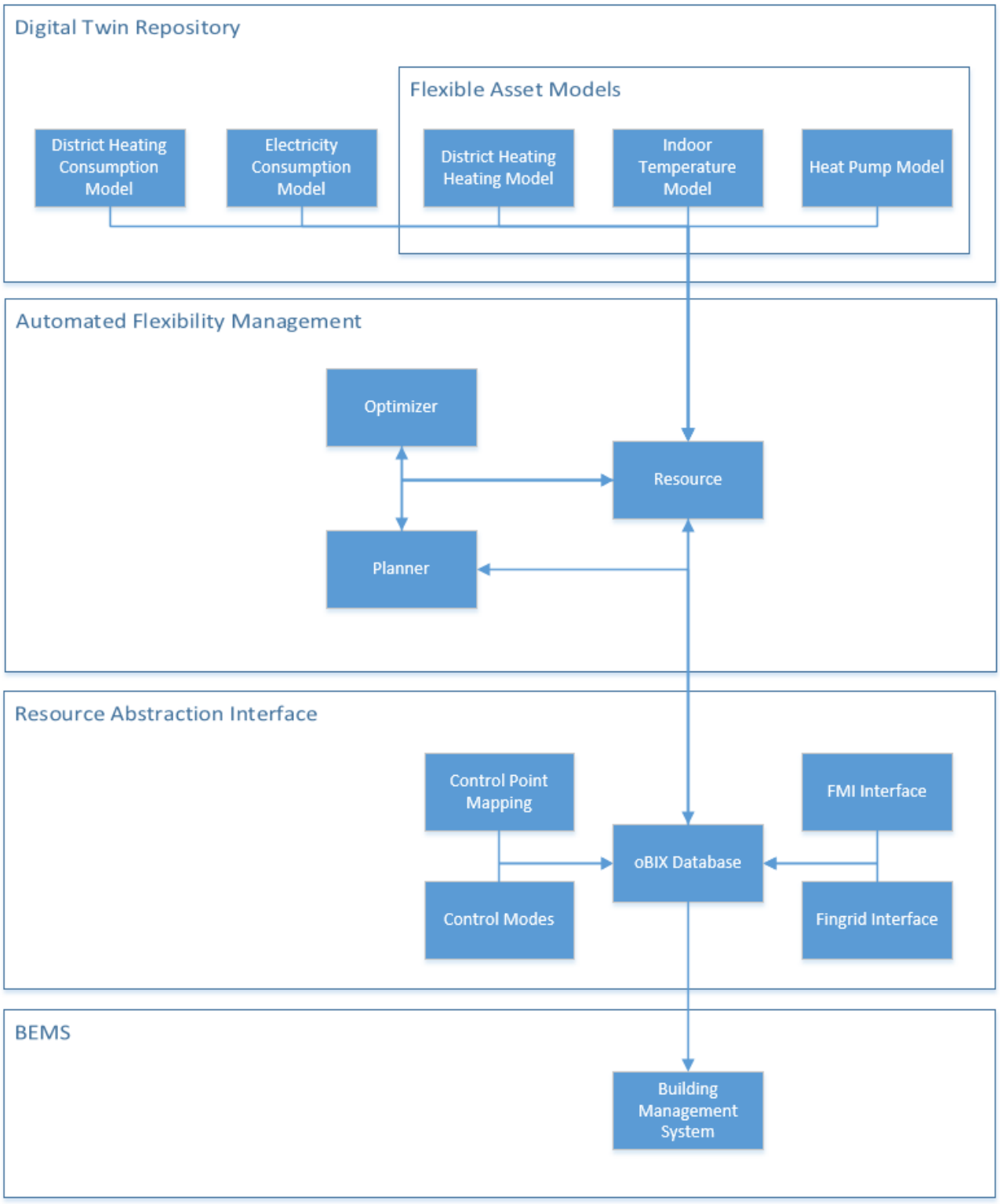

Figure 15. Different models and RAI components included in the first phase pilot building instantiation.

Figure 15 aims to relate the architectural view described in 5.1.1, especially Figure 7, with the modules in the pilot building. The flexible assets include district heating and heat pump components. In Digital Twin repository, district heating is modelled as two separate models; one for modelling the energy consumed and the other for the heating effect of the consumed energy. The reason is that the district heat is also used in heating domestic hot water. Similar division is done for the heat pump, though the electricity consumption model includes inflexible loads as well. An indoor temperature model is used to evaluate the effect of heating to the apartment temperatures. The AFM module is site agnostic and is not changed for different sites, However, it requires a

separate configuration file for site-specific details. The Resource Abstraction Interface is implemented as a oBIX Database with mappings to the BEMS and interfaces to receive weather and CO2 emissions data.

6 Conclusion

This deliverable reports the initial work on the Automated Flexibility Management module of the iFLEX Assistant framework. A software architecture of the module is presented, comprising of *Planner, Optimizer* and *Resource* components with their intra- and interconnected roles visualized in sequence diagrams. The initial implementation done is by design simple, yet suitable for first phase testing and experimentation, with groundwork laid both in design and implementation for future improvements on the software and features. In the next phase optimization features are improved in all aspects, and the code maturity is further increased.

7 List of figures and tables

7.1 Figures

7.2 Tables

8 References

Bianchini, G., Casini, M., Vicino, A., & Zarrilli, D. (2016). Demand-response in building heating systems: A Model Predictive Control approach. *Applied Energy*, *168*, 159–170. https://doi.org/10.1016/j.apenergy.2016.01.088

Chen, Y., Norford, L. K., Samuelson, H. W., & Malkawi, A. (2018). Optimal control of HVAC and window systems for natural ventilation through reinforcement learning. *Energy and Buildings*, *169*, 195–205. https://doi.org/10.1016/j.enbuild.2018.03.051

Kiljander, J., Sarala, R., Rehu, J., Pakkala, D., Pääkkönen, P., Takalo-Mattila, J., & Känsälä, K. (2021). Intelligent Consumer Flexibility Management with Neural Network-Based Planning and Control. *IEEE Access*, *9*, 40755–40767. https://doi.org/10.1109/ACCESS.2021.3060871

Kusiak, A., Tang, F., & Xu, G. (2011). Multi-objective optimization of HVAC system with an evolutionary computation algorithm. *Energy*, *36*(5), 2440–2449. https://doi.org/10.1016/j.energy.2011.01.030

Ma, Z., & Wang, S. (2011). Supervisory and optimal control of central chiller plants using simplified adaptive models and genetic algorithm. *Applied Energy*, *88*(1), 198–211. https://doi.org/10.1016/j.apenergy.2010.07.036

Pascanu, R., Mikolov, T., & Bengio, Y. (2012). On the difficulty of training Recurrent Neural Networks. *30th International Conference on Machine Learning, ICML 2013*, (PART 3), 2347–2355. Retrieved from http://arxiv.org/abs/1211.5063

Patyn, C., Ruelens, F., & Deconinck, G. (2018). Comparing neural architectures for demand response through model-free reinforcement learning for heat pump control. In *2018 IEEE International Energy Conference, ENERGYCON 2018* (pp. 1–6). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ENERGYCON.2018.8398836

Ruelens, F., Claessens, B. J., Vandael, S., De Schutter, B., Babuska, R., & Belmans, R. (2017). Residential Demand Response of Thermostatically Controlled Loads Using Batch Reinforcement Learning. *IEEE Transactions on Smart Grid*, *8*(5), 2149–2159. https://doi.org/10.1109/TSG.2016.2517211

Sutton, R. S., & Barto, A. G. (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*. https://doi.org/10.1109/tnn.1998.712192

Wang, S., & Jin, X. (2000). Model-based optimal control of VAV air-conditioning system using genetic algorithm. *Building and Environment*, *35*(6), 471–487. https://doi.org/10.1016/S0360-1323(99)00032-3

West, S. R., Ward, J. K., & Wall, J. (2014). Trial results from a model predictive control and optimisation system for commercial building HVAC. *Energy and Buildings*, *72*, 271–279. https://doi.org/10.1016/j.enbuild.2013.12.037

9 Appendix: Jira requirements

[IF-72] FN-AFM-04 Optimize flexibility based on prices (implicit demand response) Created: 15/Jun/21 Updated: 15/Jun/21

Status:	Open
Project:	iFlex Project
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Functional	**Priority:**	Major
Reporter:	Jussi Kiljander	**Assignee:**	Jussi Kiljander
Resolution:	Unresolved	**Votes:**	0
Labels:	AFM		

Rationale:	iFLEX Assistant needs to be able to optimize flexible assets schedule and/or setpoints in order to reduce end-user costs.
Pilot Finland:	Phase two
Pilot Greece:	Not applicable
Pilot Slovenia:	Not applicable

Description

The automated flexibility management module should provide mechanisms to optimize flexibility with respect to dynamic tariffs, possible across different energy vectors.

[IF-71] FN-AFM-03 Activate offered flexibility Created: 15/Jun/21 Updated: 15/Jun/21

Status:	In Progress
Project:	iFlex Project
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Functional	**Priority:**	Major
Reporter:	Jussi Kiljander	**Assignee:**	Jussi Kiljander
Resolution:	Unresolved	**Votes:**	0
Labels:	AFM		

Rationale:	Flexibility activation is required in implicit and explicit demand response.
Pilot Finland:	Phase one
Pilot Greece:	Phase two
Pilot Slovenia:	Phase two

Description

The automated flexibility management module should activate the offered flexibilities when requested via the A&F interface (assumes flexibility activation is authorized by the end-user).

To activate the flexibility the AFM module needs to find an optimal control schedule by optimizing with the model provided by the digital twin repository. Once optimal schedule is found the flexibility is activated by modifying the local energy management system parameters.

[IF-70] FN-AFM-02 - Flexibility potential Created: 15/Jun/21 Updated: 15/Jun/21

Status:	In Progress
Project:	iFlex Project
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Functional	**Priority:**	Major
Reporter:	Jussi Kiljander	**Assignee:**	Jussi Kiljander
Resolution:	Unresolved	**Votes:**	0
Labels:	AFM		

Rationale:	Flexibility potential data is needed to provide more deterministic explicit demand response.
Pilot Finland:	Phase one
Pilot Greece:	Phase two
Pilot Slovenia:	Phase two

Description

The automated flexibility management module should provide flexibility potential information to the A&M Interface module. The flexibility potential should be calculated by comparing the minimum and maximum loads to the baseline load profile at different time periods.

The length of the flexibility window and the frequency should be configurable.

[IF-69] FN-AFM-01 Provide baseline forecasts Created: 15/Jun/21 Updated: 15/Jun/21

Status:	In Progress
Project:	iFlex Project
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Functional	**Priority:**	Major
Reporter:	Jussi Kiljander	**Assignee:**	Jussi Kiljander
Resolution:	Unresolved	**Votes:**	0
Labels:	AFM		

Rationale:	Baseline load profile is required to estimate and validate the flexibility at individual prosumer level. Prosumer/consumer specific baseline load profiles can be also used to calculate aggregated load profiles at different levels of the power system.
Pilot Finland:	Phase one
Pilot Greece:	Phase two
Pilot Slovenia:	Phase two

Description

The automated flexibility management module should provide information about the baseline load profile (including consumption and local production) of the prosumer/consumer.