Fachhochschule Dortmund
University of Applied Sciences and Arts
Embedded Systems Engineering

Title of your thesis

Master Thesis

Fachhochschule Dortmund
University of Applied Sciences and Arts

Author: Sheikh Muhammad Adib Bin Sh Abu Bakar
Matriculation Number: 7219310
Supervisor: Prof. Dr. rer. nat. Stefan Henkler
Co-Supervisor: You Co-Supervisor(s)
Date: June 10, 2025

Abstract

Your Abstract

Declaration

I, Sheikh Muhammad Adib Bin Sh Abu Bakar, hereby confirm, that I have written the Master Thesis at hand independently – in case of a group work: my respectively designated part of the work -, that I have not used any sources or materials other than those stated, and that I have highlighted any citations properly. .
Dortmund, June 10, 2025

Sheikh Muhammad Adib Bin Sh Abu Bakar

Contents

List of Figures

List of Tables

1

Motivation

1 General motivation

- QUESTION (scope): "How to develop a self-adaptive system for distributed CPS?"
- Modern systems often operate in unpredictable and changing environments
- Rule-Based Adaptation :
  - Limited Flexibility
  - exhaustive offline tuning (Time-Consuming)
- Online DSE can dynamically allocate and reallocate hardware resources (for distributed CPS):
  - Makes decisions based on the current system state and goals
  - self-managing systems in uncertain or evolving environments: optimise application and hardware
  - Optimise latency, throughput, or energy efficiency
  - Adapt to changing workloads or resource availability (scheduling):
    - Task mapping (which task runs where)

    - Voltage/frequency scaling (DVFS)
    - deadline satisfaction

- QUESTION (focus): " How can online Design Space Exploration (DSE) be utilised to optimise hardware utilisation and software execution to enable self-adaptive computing systems? "
- QUESTION (focus): " How to design a lightweight model but accurate to reduce complexity? "

2 Research gap/Problem definition

- lack of standard self-adaptive definition
- modelling complex dCPS to have accurate models for DSE
- weather forecast dependent configuration for dCPS

3 Contribution

- leverage machine learning to improve models over time?
- weather dependent UAVs fleet configuration (scheduling)

4 Paper Flow

2

Related Work

3

Preliminaries

1 System

2 Self-adaptive System

In the literature, there is no standard definition for self-adaptive CPS. []\[1\] discussed in detail about self-adaptive CPS definition while ignoring the broader view of CPS.

Complex and self-adaptable systems need to be autonomous, system- and environment-aware, and self-controlled []\[2\]

[1] https://mediatum.ub.tum.de/doc/1692104/08vhwc8hsj1oacb0165por366.petrovska-ana.pdf

[2] A Review of the Principles of Designing Smart Cyber-Physical Systems for Run-Time Adaptation

4

Approach

1 Methodology

DSE-M []

This section discusses the method used to find a feasible solution for a self-adaptive distributed cyber-physical system. To structure the support system in realising a self-adaptive system, a model-based system engineering (MBSE) approach is used with a strong emphasis on using SysML (Systems Modelling Language) as the primary modelling tool and practised with the guidance and standards promoted by INCOSE.

The approach is to start with a comprehension of the self-adaptive system as explained in chapter 3. To enable self-adaptive capability, many approaches have been proposed in the literature. One of the most popular approaches is to adapt the concept of MAPE-K (Model, Analysis, Plan, Execute and Knowledge) from [**??**] as visualised in figure **??**. This approach could be seen as an extension to a closed-loop control system

[1] Design Space Exploration in Robotics

[2] `https://ieeexplore.ieee.org/document/7194653`

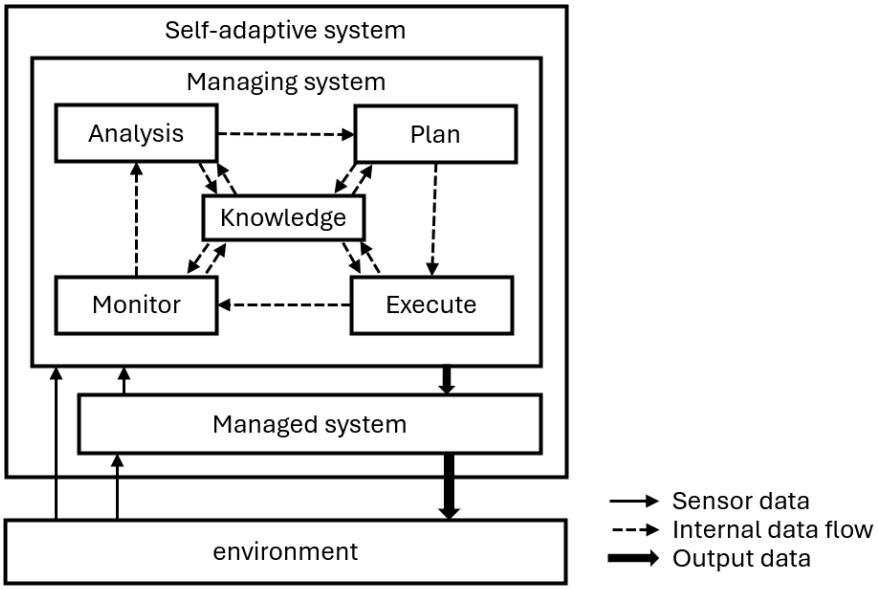

Figure 4.1: MAPE-K concept

< ... The explanation of the MAPE-K ... >

The influence of the managing system, or known as the self-adaptive component in this paper to the managed system and its dependency on the environment is visualised in Figure **??**

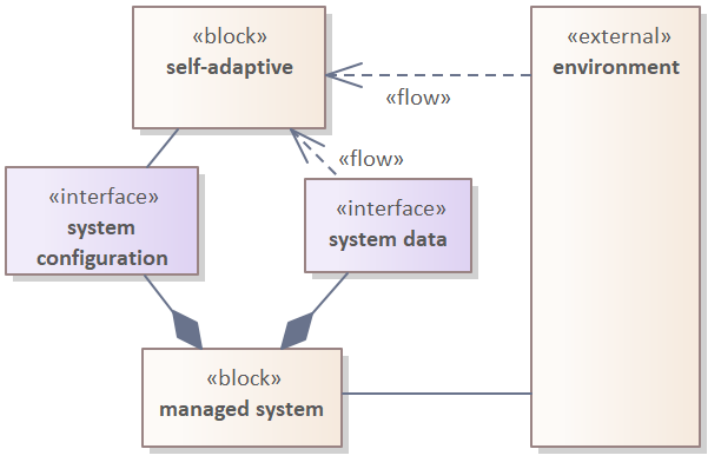

Figure 4.2: Self-adaptive as a component in a larger system view

The detail of the implementation of the MAPE-K concept within the self-

adaptive component is depicted in Figure 4.3

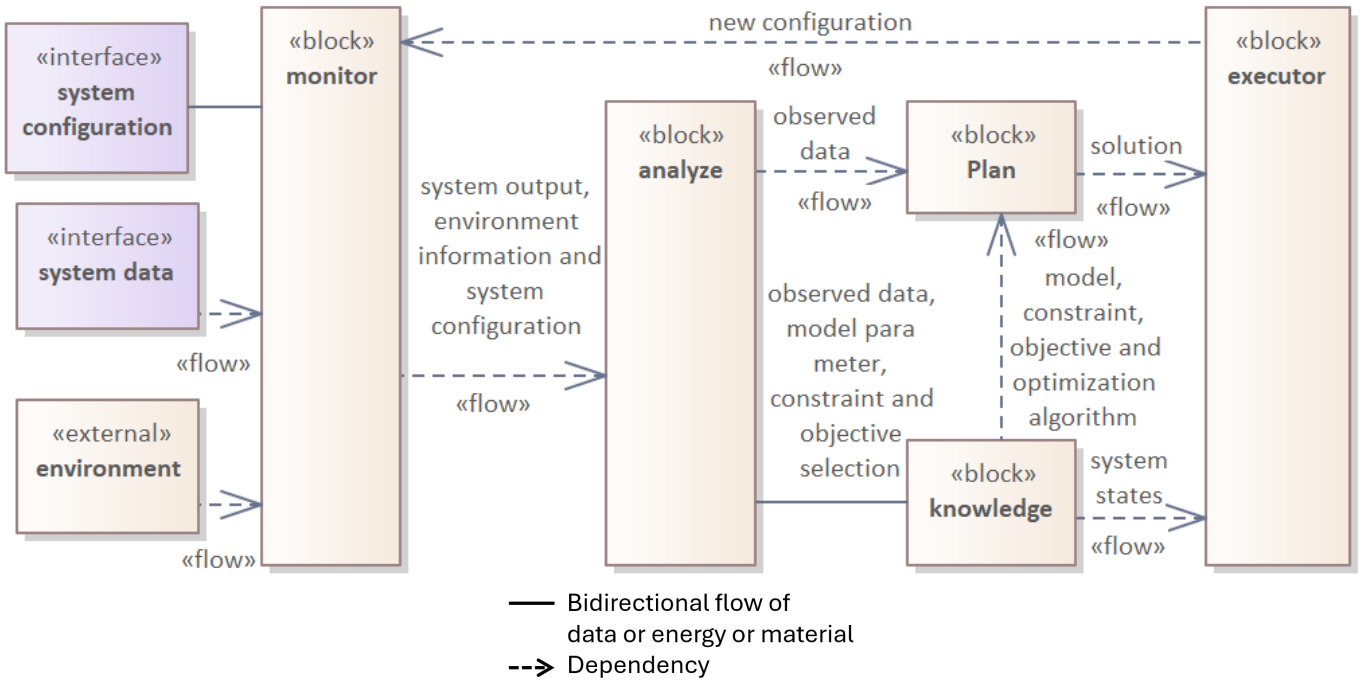

Figure 4.3: MAPE-k adaptation within self-adaptive block

< ... The explanation of the diagram 4.3 ... >

The monitor is responsible for reading and updating both the managed system configuration, application and hardware configuration. The observed data of the managed system by the monitor is used to select the appropriate objective and improve the model used for design space exploration via a machine learning algorithm, so that the model becomes better over time. Figure 4.4 shows that these processes are executed within the analysis block.

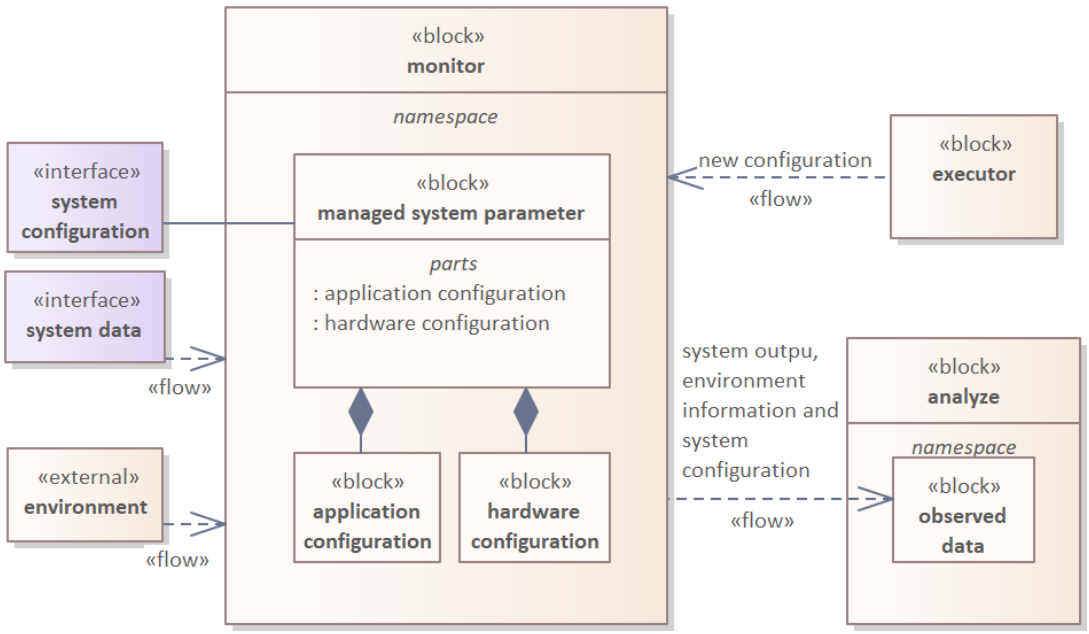

Figure 4.4: Monitor input and output

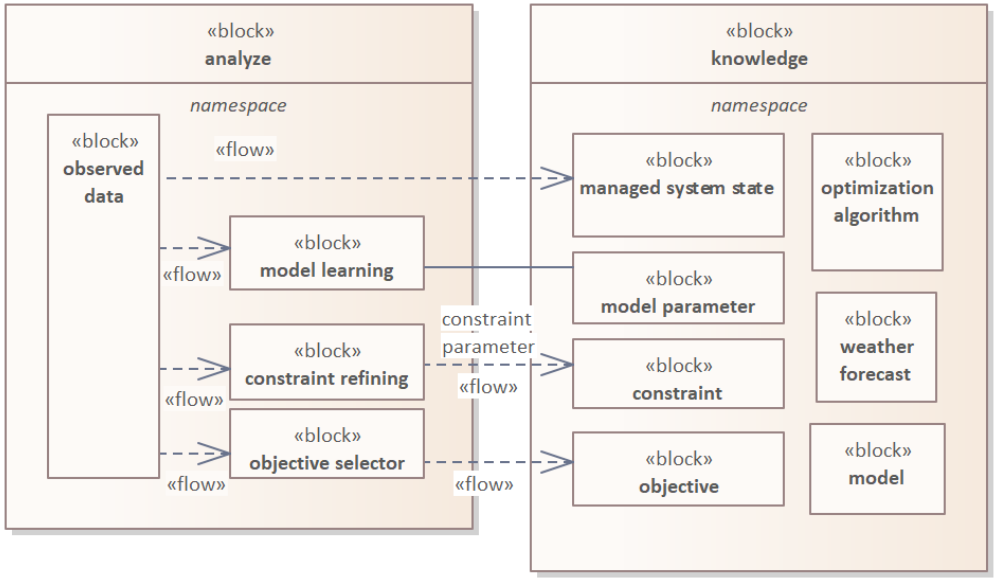

Figure 4.5: Knowledge and analysis block in self-adaptive block

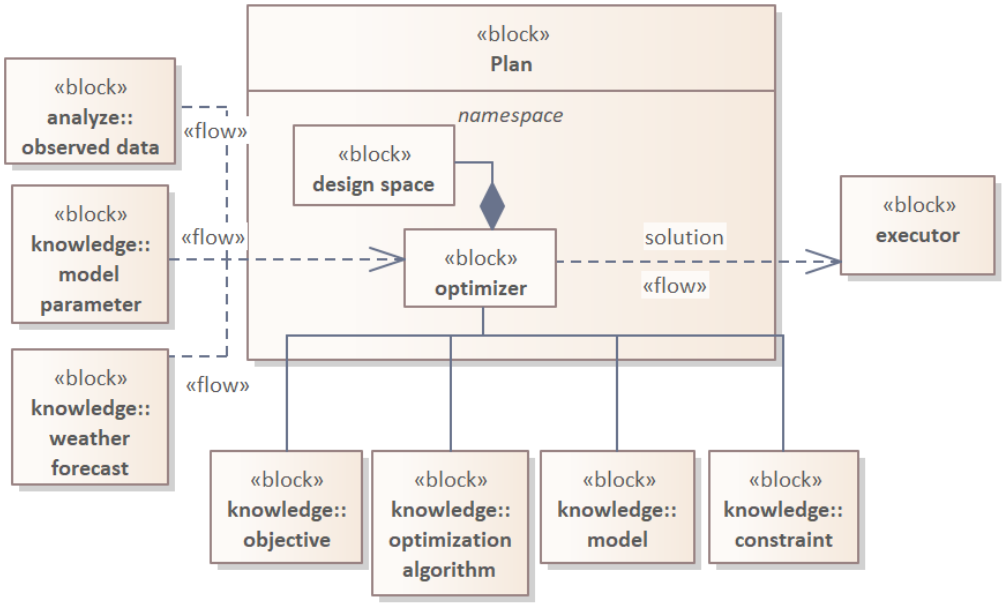

Figure 4.6: Plan block in self-adaptive block

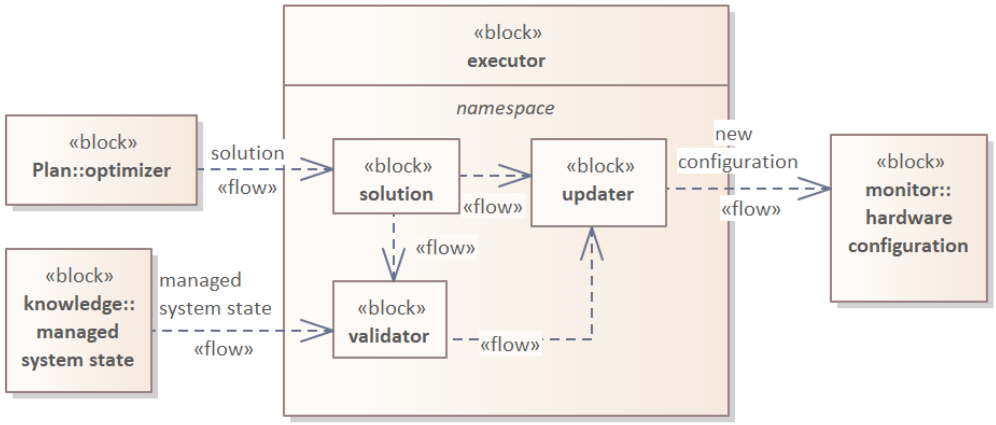

Figure 4.7: Executor block in self-adaptive block

The behaviour of the self-adaptive block is depicted in Figure 4.8

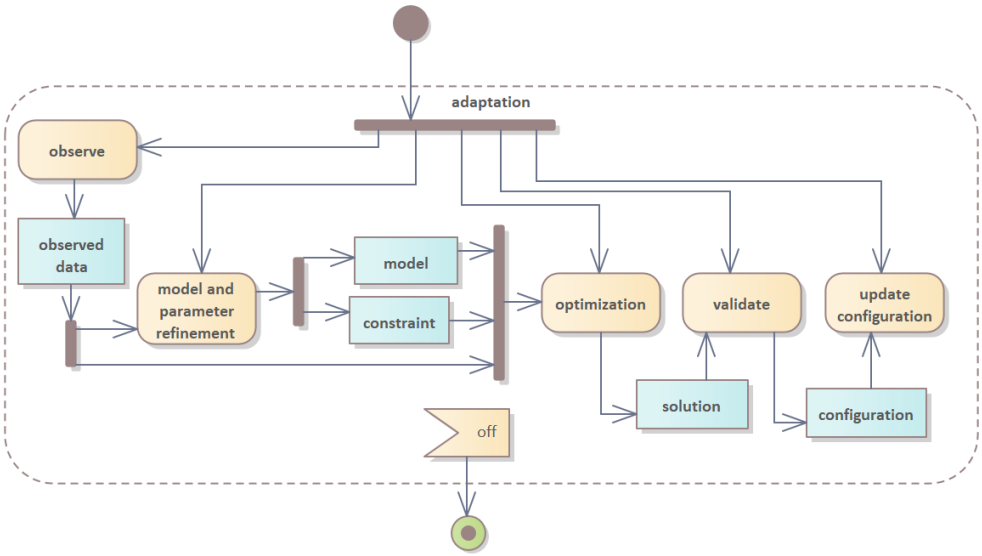

Figure 4.8: The behaviour of the self-adaptive block

In the next section, the case study will be explained, which will be used as an example of a managed system and to show how the whole system will work together.

2 Case study

The theme of this case study is a smart farming system, which encompasses multiple interconnected subsystems. Given the complexity of such a system, this paper focuses on a select few subsystems that are most relevant to the requirements outlined in Figure 4.9. These requirements are specifically tailored to address the key interests and objectives of this study.

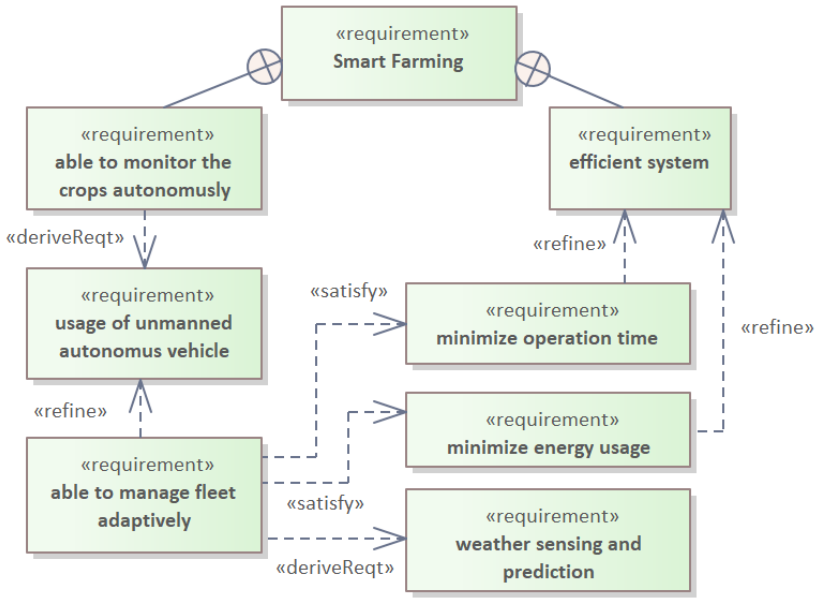

Figure 4.9: Smart farming requirements

From the defined requirements diagram, it can be seen that the system has multiple functionalities, whereas its configuration is adaptively changing depending on the weather. The main functional requirements can be simplified as shown in Figure 4.10 to achieve certain goals

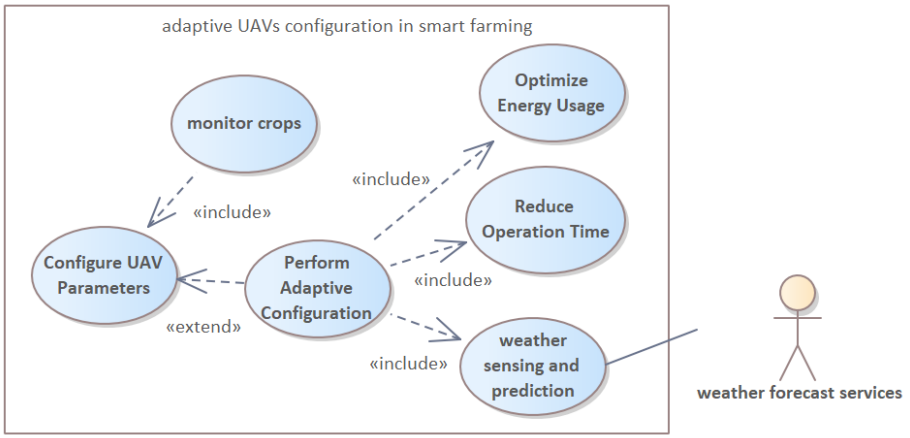

Figure 4.10: Adaptive UAVs configuration in smart farming use case

Figure 4.11 shows the logical system architecture of the fleet management system

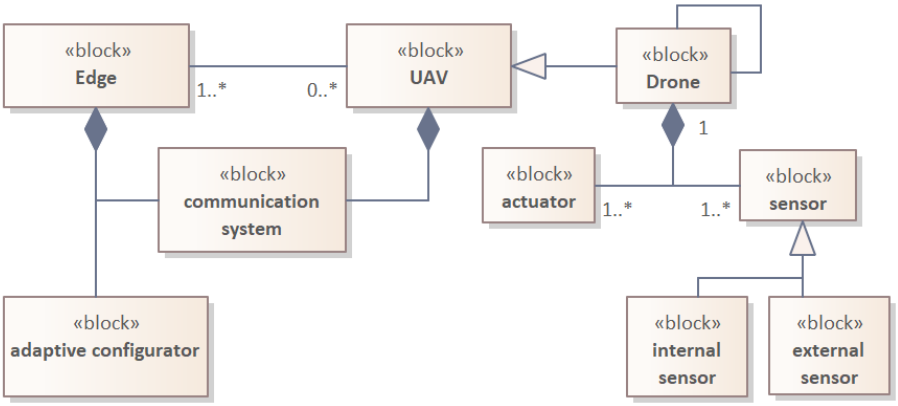

Figure 4.11: Managed system logical architecture: fleet managing system

The behaviour of the system is visualised in Figure 4.12

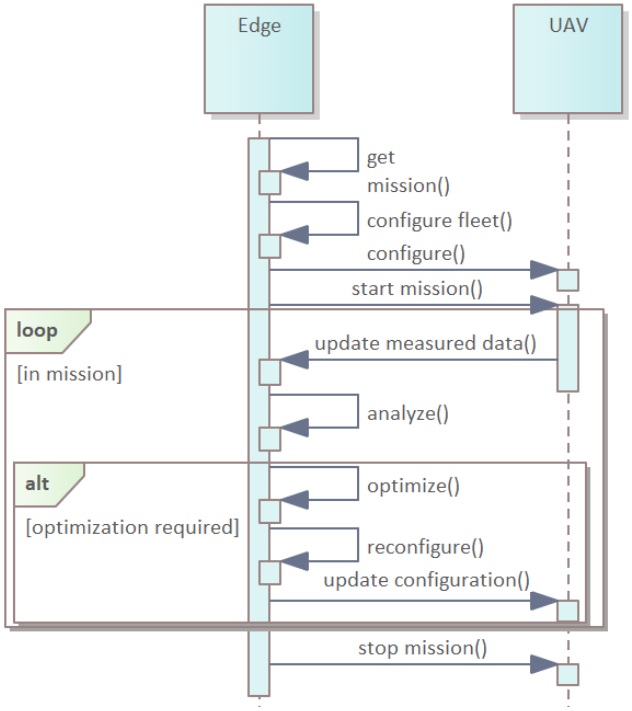

Figure 4.12: The system behaviour as a whole on the abstract level

3 Parameter and Objective

From Figure 4.9 and Figure 4.10, the objective of the implementation of self-adaptivity can be listed as follows:

1. Minimisation of Energy consumption
2. Minimisation of operation time

The observed parameters are controlled parameters like speed and environmental information like wind speed. This parameter will be discussed in detail in the upcoming section and chapter.

Figure 4.13 shows the behaviour of the observer block where weather data is needed to configure the managed system appropriately

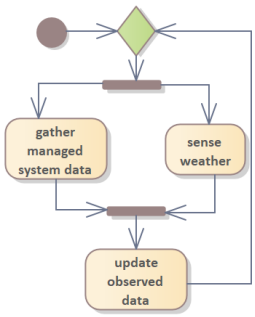

Figure 4.13: Behaviour of observer block

This observation is used by the self-adaptive component to optimise the controlled parameter to achieve those objectives

4 Models

This section describes the model used to construct the design space, which will later serve as the basis for optimizing controllable parameters. The modeling approach is inspired by the method presented in []³, where the system is decomposed into a set of functional blocks, each representing a component of the overall model.

In this framework, the system F is represented as a collection of individual functions:

$$F = \{f_1, f_2, ..., f_n\} \tag{4.1}$$

[3]DESSERT: Design Space Exploration Tool based on Power and Energy at System-Level

Each function f_i represents a distinct operation or behavior of the system. These functions collectively define the operational model and are subject to evaluation based on specific performance metrics.

The detailed explanation of how this functional decomposition is applied follows in the next subsection.

Moreover, the evaluation of these functions can be refined over time using machine learning techniques. This adaptive capability enables not only improved estimation and prediction accuracy for each function but also enhances the modeling of system constraints. As a result, the design space can be effectively pruned, leading to reduced computational costs.

The next subsection will focus specifically on the evaluation of the functions with respect to energy consumption, forming the foundation for modeling the system's total energy usage.

4.1 Energy model

In this paper, the computed energy consumption is an estimate of the average energy consumption. Based on the definition, the computed total energy can be formulated as follows:

$$E = \sum_{i}^{N_d} P \times T \tag{4.2}$$

where N_d is the total number of used drones. To estimate the energy consumed by a UAV, the factors that influence the energy consumption through required functionality are analysedand the result is shown in Figure 4.14

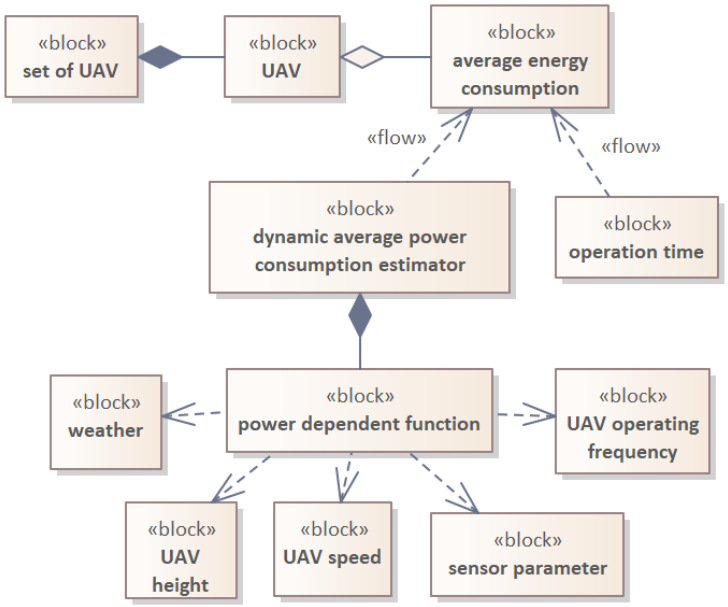

Figure 4.14: Energy model for the set of drones

The functionality is grouped as power-dependent functions. Five main factors influence the power consumption, which are weather conditions, the height, speed and operating frequency of the UAV and the parameters of the sensor attached to the UAV.

It can be seen from the diagram that the model can be split into two main models, which are the power model and the operation time model. Those models will be explained in the upcoming subsection

4.2 Power model

The drone system could be split into 5 main functionalities as shown in Figure 4.15 which are, basic ,computation, communication, sensor and actuator power consumption. To reduce the complexity, only one sensor is consider which is camera for image sensing and other basic sensing functionality like IMU sensor are considered as apart of actuator power consumption or specifically maneuvering function.

The computation power consumption is non zero if and only if there is another task that is not a fundamental task for sensing and actuating that need to be done by the drone. The computation resource by such the task is frequency dependent. Using a frequency as a parameter allow higher abstraction level to measure energy consumed by a processor.[][4]

[4] https://github.com/tgmattso/OpenMP_intro_tutorial/blob/master/omp_hands_on.pdf

A single processor can be seen as a capacitor with capacitance C with supplied voltage V. With those value, the work done can be calculated as follow:

$$W = C \cdot V^2 \tag{4.3}$$

The power can be derived with the operation frequency as in equation 4.4.

$$P = W \cdot f \tag{4.4}$$

In the considered scenario in this paper, the computation power consumption value will always be 0 as there is other task than maneuvering and image sensing.

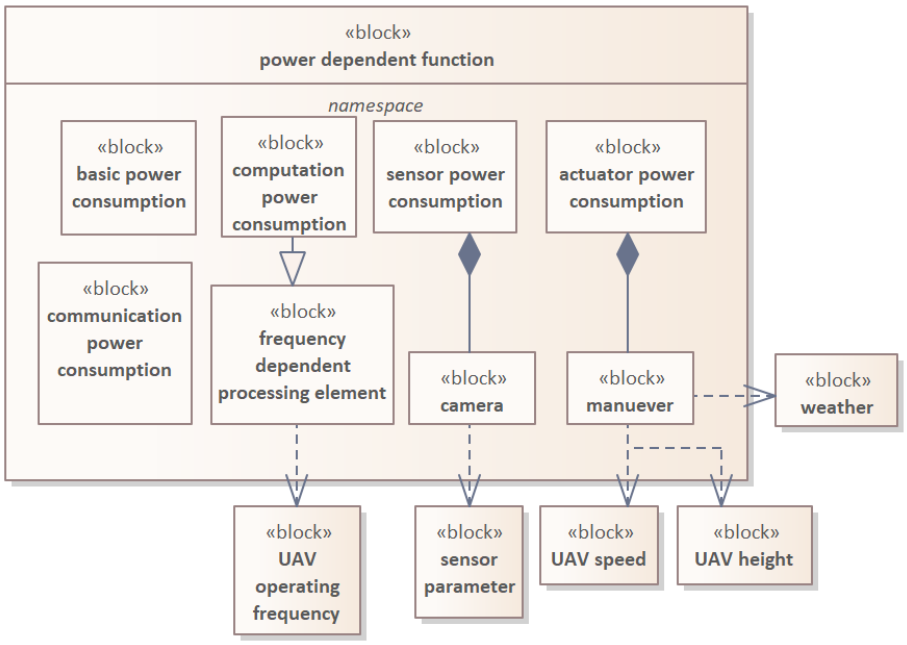

Figure 4.15: Power dependent functions

Based on Figure 4.14 and Figure 4.15, the total power consumption by a set of drones can be define as in the equation **??**.

$$P_{total} = \sum_{i=1}^{N_{drone}} P_{basic,i} + P_{comp,i} + P_{sensor,i} + P_{actuator,i} \tag{4.5}$$

The $P_{basic,i}$ is the power of basic functionalities that cannot be avoided whenever the the drone is started or in other word power consumed in idle state. Meaning that, it is always true that $P_{basic} > 0$ and can be treat as a constant as in eqiation 4.6. The $P_{com,i}$ is the power consumed by a computing resource as explained before, where in this paper the value will always be 0.

The $P_{sensor,i}$ in this paper focused on the power consumed for image sensing including the power required to process the images. The $P_{actuator,i}$ is the power required by the maneuvering functionality.

$$P_{basic,i} = x_i \quad | \, x_i \in \mathbb{R} \tag{4.6}$$

In the considered scenario, the design of the drone is not the main focus, but the adaptive fleet configuration and its dependency. In general, focusing on improving a single drone hardware design configuration is not within the scope of this paper. For that reasons, the model used for every single functionality is more linear rather than complex model. This approach will alow the optimization process later much more less complex.

The $P_{sensor,i}$ or $P_{camera,i}$ depend only on the set FPS and number of pixel used by the sensor. This linear approach based on the analysis of the power consumed by an image sensing device with varying resolution and FPS done by []⁵, where the power consumed is linearly increasing with the resolution when the resolution is above 320×240. This can be disribe using equation 4.7 and 4.8. The description of used parameter can be found in Table 4.1.

$$P_{sensor,i} = P_{camera,i} = \sigma_{t(i)} \cdot n_{pixel,i} + \omega_{t(i)} \cdot f_{fps,i} + \epsilon_{t(i)} \tag{4.7}$$

$$t(i) \in T, \text{ where } |T| = N_{\text{type of drones}} \tag{4.8}$$

Symbol	Description
$m_{d,i}$	Drone's mass
$m_{p,i}$	Load mass
h, i	Drone's height
$h_{ref,i}$	Height reference
$v_{a,i}$	Resultant speed
g	Gravitational acceleration
$\alpha_{t(i)}$	Drone's drag coefficient factor
$\delta_{t(i)}$	Propulsion efficiency of the drone
Γ_i	Drone's thrust
v_{wind}	Wind speed
θ_{wind}	Angle of wind attack
v_i	Speed of the drone
$\eta_{t(i)}$	Power transfer efficiency
$\beta_{t(i)}$	Constant related to altitude's effect on air density

Table 4.1: List of parameters and their descriptions

The $P_{actuator,i}$ depends on the altitude of drone, wind speed, angle of wind attack, speed of drone, the mass of drone and the mass carry by the drone as

[5]Intelligent Systems and Applications-Impact of Image Sensor Output Data on Power Consumption of the Image Processing System

describe in the equation 4.9. The description of used parameter can be found in Table 4.1.

$$P_{actuator,i} = P_{manoeuver,i} = \frac{\Gamma_i(v_{wind} \cdot sin(\theta_{wind}) + v_i) + \Gamma_{h_0,i}(v_{wind} \cdot cos(\theta_{wind}))}{\eta_{t(i)}} \tag{4.9}$$

[]6

$$\Gamma_i = \frac{g(m_{d,i} + m_{p,i}) \times v_{a,i} + \alpha_{t(i)} v_{a,i}^3 + \beta_{t(i)} \left(\frac{h,i}{h_{ref,i}}\right)(m_{d,i} + m_{p,i})}{\delta_{t(i)}} \tag{4.10}$$

[]7

$$\Gamma_{h_0,i} = \frac{g(m_{d,i} + m_{p,i}) \times v_{a,i} + \alpha v_{a,i}^3}{\delta_{t(i)}} \tag{4.11}$$

Detailed drone (manoeuvre) energy consumption is complex as explained in []8 drone hover simple power estimation explained in []9

4.3 Operation time model

The operation time depends on the desire resolution of sensor an covered area at a time by an actuator. Resolution of a sensor is define by coverage area per sensor pixel. The smaller the covered area per pixel, the higher the resolution is. Meaning that, the lower the position of the sensor, the higher the resolution is as illustrated in Figure 4.16. Varying the high of the drone, influence the covered area by an actuator, for example a spray, whereas the lower the drone the smaller covered area at a time causing longer time needed to cover the whole field with the consideration of constant speed. This can also be visualize using Figure 4.16.

[6] https://link.springer.com/article/10.1007/s11227-025-07105-0

[7] comparative study on energy consumption

[8] https://www.researchgate.net/publication/363896841_Quadrotor_Model_for_Energy_Consumption_Analysis/figures?lo=1&utm_source=google&utm_medium=organic

[9] https://news.quadpartpicker.com/how-to-estimate-and-calculate-drone-flight-characteristics/

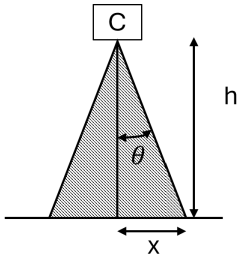

Figure 4.16: Relation between drone height, h and coverage, x

The total operating time can be define as:

$$T_i = \lambda_i(\frac{8tan^3(\frac{\theta_{\text{camera},i}}{2})}{\text{total area}_i}) \cdot \frac{h_i^3}{v_i} \tag{4.12}$$

5 Model online calibration

The model explained in pervious section, section 4.4 contains parameter coefficient with uncertainties which are δ, β, η and α.

To have a better estimation of overall power consumption and estimation of operation time, the value of those parameters should be improved over time. This is where the model learning block in Figure 4.5 plays a big role. This block implements a machine learning approach where the input data is the observed data as shown in Figure 4.8. Figure 4.17 shows the detailed activity of model and parameter refinement. The main focused action in this section is the *train model* action.

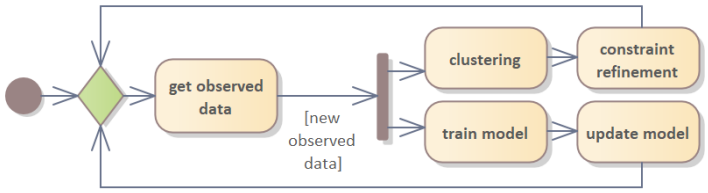

Figure 4.17: Detail activity of model and parameter refinement

From the equation 4.1, a system can be seen as a collection of functions, and each function contains several parameters depending on properties of a system being evaluated, as in equation 4.7 and 4.9. Based on those equations, the definition of a system can be further refined as follows:

$$F = \{f_1(X_1, \theta_1), f_2(X_2, \theta_2), ..., f_n(X_n, \theta_n)\} \tag{4.13}$$

where, X_i is the feature vector and θ_i is the parameter coefficient. The use of machine learning is to have better values of those coefficients that minimise the uncertainties. This implementation will be further explained in the next subsection.

5.1 Machine learning model

Using the equation 4.13, the model of power consumption can be refined as in equation 4.14. As a result, the power consumption by a sensor function and actuator function can be seen as in equations 4.15 and 4.16. The operation time model is also refined as in equation 4.17. The $t(i)$ is used instead of i because the same type of drone should have approximately the same coefficient. Meaning that fewer parameters will be trained with the same amount of data. The the upcoming subsection, each function will be described in detail.

$$P_{total,i} = \sum_{i=0}^{N_{\text{function}}} f_i(X_i, \theta_{t(i)}) + \epsilon_{t(i)} \tag{4.14}$$

$$P_{actuator,i} = P_{\text{maneuver}}(X_{m,i}, \theta_{m,t(i)}) + \epsilon_{m,t(i)} \tag{4.15}$$

$$P_{sensor,i} = P_{\text{camera}}(X_{c,i}, \theta_{c,t(i)}) + \epsilon_{c,t(i)} \tag{4.16}$$

$$T_i = f(X_{T,i}, \theta_{T,t(i)}) + \epsilon_{T,t(i)} \tag{4.17}$$

Actuator Power Consumption Learning Model

The P_{actuator} can be expanded as shown in equation 4.18. The equation can be further simplified to have better distinction between the feature parameter and coefficient parameter resulting the equation 4.19. The definition of variable used in the equation can be found in table 4.2

$$\begin{aligned} P_{\text{actuator},i} = & \\ & \frac{1}{\eta_{t(i)}} \left[\left(\frac{g(m_{d,i} + m_{p,i}) \cdot v_{a,i} + \alpha_{t(i)} v_{a,i}^3 + \beta_{t(i)} \left(\frac{h_i}{h_{\text{ref},i}} \right) (m_{d,i} + m_{p,i})}{\delta_{t(i)}} \right) (v_{\text{wind}} \cdot \sin(\theta_{\text{wind}}) + v_i) \right. \\ & \left. + \left(\frac{g(m_{d,i} + m_{p,i}) \cdot v_{a,i} + \alpha v_{a,i}^3}{\delta_{t(i)}} \right) (v_{\text{wind}} \cdot \cos(\theta_{\text{wind}})) \right] \end{aligned} \tag{4.18}$$

$$\begin{aligned} P_{\text{actuator},i} = & \theta_{1,t(i)} (M_i \cdot A_i) + \theta_{2,t(i)} \left(A_i^3 \right) + \theta_{3,t(i)} (H_i \cdot M_i) \\ & + \theta_{4,t(i)} (M_i \cdot A_i \cdot (v_i + v_{\text{wind}} \cdot \sin(\theta_{\text{wind}}))) \\ & + \theta_{5,t(i)} \left((M_i \cdot A_i + A_i^3) \cdot (v_{\text{wind}} \cdot \cos(\theta_{\text{wind}})) \right) + \epsilon_{t(i)} \end{aligned}$$

Symbol	Description
$M_i = m_{d,i} + m_{p,i}$	Total mass (drone + payload)
$A_i = v_{a,i}$	Airspeed of drone i
$H_i = \dfrac{h_i}{h_{\text{ref},i}}$	Normalized altitude (relative to reference)
v_i	Ground-relative velocity
v_{wind}	Wind speed
θ_{wind}	Wind direction (angle)
$\theta_{1,t(i)}, \ldots, \theta_{5,t(i)}$	Regression model coefficients to be learned
$\epsilon_{t(i)}$	Residual error

Table 4.2: Variable definitions for the power consumption regression model.

Sensor Power Consumption Learning Model

From the power consumption by sensor function described in equation 4.7, the future parameter and coefficient parameter can be distinct as shown in equation 4.19.

$$P_{\text{sensor},i} = P_{\text{camera},i} = \theta_{6,t(i)} \cdot n_{\text{pixel},i} + \theta_{7,t(i)} \cdot f_{\text{fps},i} + \epsilon_{t(i)} \tag{4.19}$$

5.2 Learning

The learning objective is to minimize the loss as shown in equation 4.20 and 4.21. In this approach, mean square error is used.

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(P_{\text{actuator},i} - \sum_{j=1}^{5} \theta_{j,t(i)} x_{j,i} \right)^2 \tag{4.20}$$

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(P_{\text{sensor},i} - \sum_{j=6}^{7} \theta_{j,t(i)} x_{j,i} \right)^2 \tag{4.21}$$

In this paper, the learning algorithm used is linear regression.

6 Online Design Space Exploration

6.1 Design Space

The model described in section 4.4 is used to build the design space. This design space is dynamic as the used model is dynamic, making it adaptive to the environment.

Objective

In this paper, the main objective for the considered scenario is to minimise energy consumption as described below.

$$min\{E_{total} = \sum_{i}^{N_{drone}} P_{total,i} \times T_i\} \tag{4.22}$$

Continuously constructing and exploring the design space is expensive. This is where the reasoning is required before running the optimisation by the plan block, as shown in Figure 4.18.

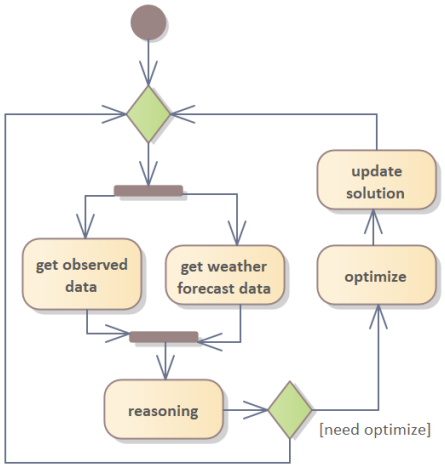

Figure 4.18: Behaviour of optimisation block

The constraint used to construct and prune the design space will be explained in detail in section 4.7.

6.2 Exploration algorithm

To efficiently explore the dynamic design space, a hybrid approach that combines mathematical optimisation with heuristic reasoning is employed. The core of the exploration algorithm is powered by Gurobi, a state-of-the-art solver for linear and mixed-integer programming problems. Gurobi is chosen for its performance, scalability, and ability to handle complex constraints inherent in energy-aware system design.

<... more explanation on exploration based on findings ...>

Gurobi is integrated into the exploration framework to solve the optimisation problem defined in equation 4.22. The solver evaluates multiple configurations within the design space, each representing a potential system state with asso-

ciated energy consumption and execution time. The optimisation process aims to identify the configuration that minimises the total energy consumption while satisfying system constraints such as deadlines, resource availability, and task dependencies.

complexity...

7 Constraint

The design space exploration described in section 4.6 is subjected to the constraint described in the equation 4.23, 4.24, 4.25, 4.26, 4.27, 4.28 and 4.29.

$$\text{resolution}_{max,x} < \text{resolution}_{x,i} = \frac{2 \times tan(\frac{\theta_{\text{camera},i}}{2}) \times h}{\text{pix}_{x,i}} < \text{resolution}_{min,x} \tag{4.23}$$

$$\text{resolution}_{max,y} < \text{resolution}_{y,i} = \frac{2 \times tan(\frac{\theta_{\text{camera},i}}{2}) \times h}{\text{pix}_{y,i}} < \text{resolution}_{min,y} \tag{4.24}$$

$$\frac{\text{resolution}_{y,i}}{\text{resolution}_{x,i}} = \frac{\text{pix}_{y,i}}{\text{pix}_{x,i}} \tag{4.25}$$

$$f_{fps,i} > \frac{v_i}{2 \times tan(\frac{\theta_{\text{camera},i}}{2})} \tag{4.26}$$

$$\text{maximum energy per drone}, P_{t(i)} \times T_{t(i)} < C_{\text{battery},i} \tag{4.27}$$

$$\text{total area} = \sum_{i}^{N_{drones}} \text{total\_area}_i \tag{4.28}$$

$$\text{number of drone}, N_{drone} < \text{number of available drone}, N_{available} \tag{4.29}$$

The constraint is used to guide the construction of the design space. The larger the design space, the more resources and time are needed to find a specific solution. For those reasons, many researchers try to find a way to reduce the design space, including a pruning method.

In this paper, the pruning is done through constraints. That means some of the constraints are dynamically changing. This is done via clustering as mentioned in Figure 4.17.

<... more explanation for clustering ...>

5

Implementation

In this chapter, the details of implementing the self-adaptive system are described in detail. The implementation process involves refining the previous approach into an executable implementation for the evaluation and validation of the approach. First, the refinement of the logical architecture described in section 5.1.

1 System Structure

For the implementation, the logical architecture in Figure 4.2 and 4.3 are first refine into physical architecture as shown in Figure 5.1

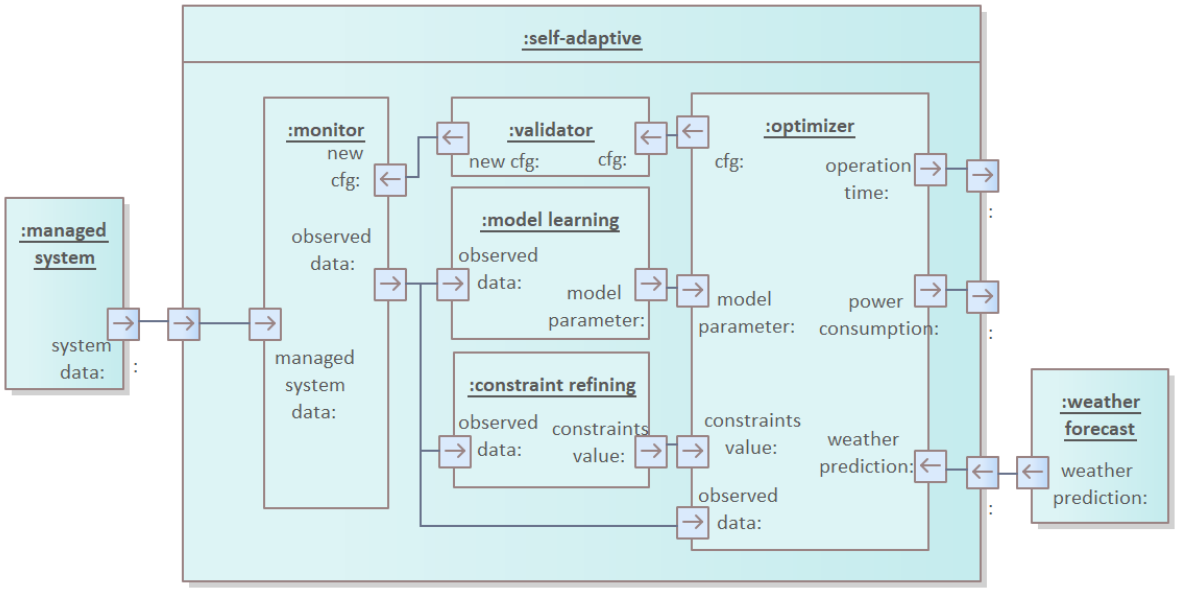

Figure 5.1: Components of self-adaptive block

To reduce the complexity, the managed system consistsof only two main functions which are sensor and actuator as depicted in Figure 5.2

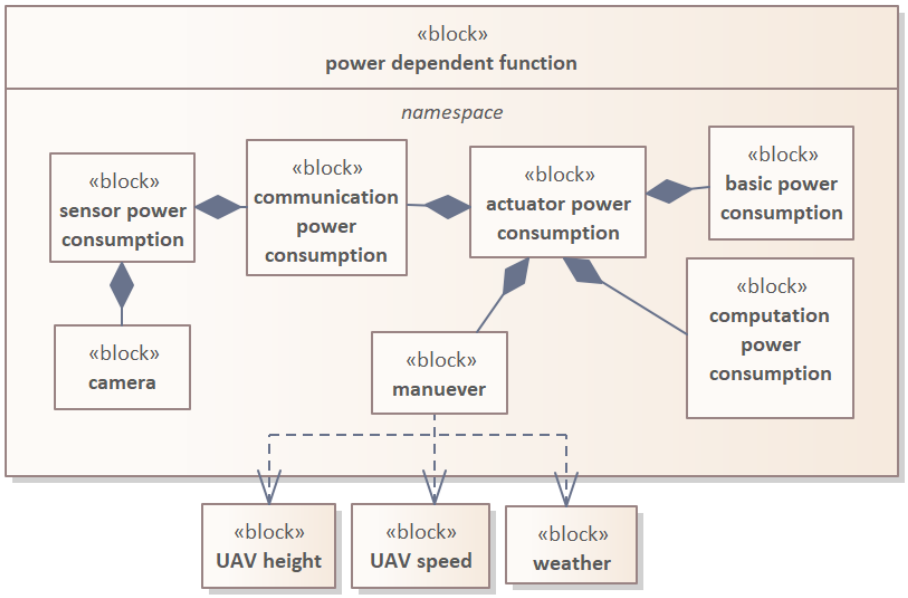

Figure 5.2: Components of self-adaptive block

2 Simulation Framework

- SystemC
- Gurobi
- Dataset

3 Dataset

A dataset is required to train and validate the model during the simulation of a self-adaptive system.

This paper focuses on the approach and not specifically on the case study scenario that is used to explain the approach. The dataset used to train and validate the ML model during the simulation is gathered from the literature.

The gathered dataset is not fully accurate as they are combined from different datasets of different drones, but sufficient to simulate the approach used in this paper.

The dataset used for basic power consumption and manoeuvre power consumption, which includes wind angle and speed parameters, is used from []1.

The sensor focused on for the simulation is a camera whose resolution and FPS can be varied. The dataset is from []2

[dataset]3

4 Learning processes implementation

The learning process is done by the *model learning* component in Figure 5.1. This component train two model described in equation 4.19 and equation 4.19.

The training objective for both models is as described in equation 4.21 and equation 4.20. The algorithm used is a regression algorithm, as it is easier to implement.

The library used for the algorithm is MLpack.

5 Exploration

[1] `https://kilthub.cmu.edu/articles/dataset/Data_Collected_with_Package_Delivery_Quadcopter_Drone/12683453`

[2] Intelligent Systems and Applications-Impact of Image Sensor Output Data on Power Consumption of the Image Processing System

[3] `https://kilthub.cmu.edu/articles/dataset/Data_Collected_with_Package_Delivery_Quadcopter_Drone/12683453`

6

Evaluation

1 benefits of dynamic frequency

3 task, same arival time $a_i = 0 \mid i = \{0, 1, 2\}$

Task	**Energy Consumption**	
$\tau_0(1, 14)$, $\tau_1(2, 4)$, $\tau_2(3, 12)$	Total energy (on a single PE)	~ 18
	Total energy (on 2 homogeneous PE with dynamic frequency scaling and a recovery slot for fault handling	~ 14

Table 6.1: energy consumption

2 Limits for energy minimisation

Each task/processing element has a minimum frequency. below it ... high energy ¡graph¿

3 Computation complexity

4 Simulation result

¡graph f, utilization, PE , ...¿

7

Summary and Outlook

Bibliography

A

Appendix