Deliverable D4.8
Virtual Laboratory version 3.0 with Artificial Agent 1.0

Finn Olav Bjørnson (SINTEF Ocean), Aya Saad (SINTEF Ocean)

Version 1.0

WP 4
Deliverable 4.8
Lead Beneficiary: SINTEF Ocean
Call identifier: Biological and Medical Sciences - Advanced Communities: Research infrastructures in aquaculture
Topic: INFRAIA-01-2018-2019
Grant Agreement No: 871108
Dissemination level: PU
Date: 30.04.2025

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871108 (AQUAEXCEL3.0). This output reflects only the author's view and the European Commission cannot be held responsible for any use that may be made of the information contained therein.

Contents

1. Objective

This document is part of the AQUAEXCEL3.0, WP4 ‘Technological tools for improved experimental procedures’ which aims at developing a virtual laboratory system, comprising several mathematical models like fish growth, hydrodynamic flow field, water quality, and fish behaviour that enables virtual experiments in aquaculture research facilities. The focus of this document is to present an overview of the outcome of subtask 4.1.1. The task encompassed integration of the models developed in the other subtasks into a common framework that allows the models to work together, extension of the virtual laboratory developed in AQUAEXCEL2020 (Bjørnson et al. 2020) and an artificial agent that can assist users in running virtual experiments. As such the main deliverable D4.8 is the virtual laboratory that users can access at https://aevirtuallab.online/. This document provides a brief technical overview of the solution. Specifically, it contains background information for the modelling framework, descriptions of the solution chosen for integration of the models, the solution for the virtual laboratory and the artificial agent, as well as selected examples of use of the laboratory.

2. Background

In AQUAEXCEL^{2020}, one of the main research activities was the development of a virtual laboratory system that could enable virtual experiments in aquaculture research facilities. This resulted in three models: growth (Lika et al. 2020), water treatment (Abbink et al. 2020) and flow fields (Alver et al. 2020). These models were combined in a framework that allowed them to synchronize at given times and thus work together to produce combined simulations. A simple web interface with limited flexibility for combining the different models (Bjørnson et al. 2020) was developed to demonstrate the integrations.

In AQUAEXCEL3.0 we wished to continue our work from AQUAEXCEL^{2020}. The growth model was extended to more life stages and two new species were added, European sea bass and pikeperch (Lika et al. 2024). The water treatment module was extended with a new CO_2 module and a new model for ponds were added (Gyalog et al. 2024). Additional flow fields were also simulated, adding to the existing ones (Alver et al. 2024). In addition, a new model for fish behaviour was developed for Atlantic salmon (Endresen et al. 2024).

The main goal of the subtask described in this document was to ensure that the different models could communicate and provide a platform for making them available to users. Thus, this deliverable makes use of all the four models and provides the integration and presentation of all of them.

3. Framework and system architecture

For completeness we include sections from the previous report (Bjørnson et al. 2020) about the underlying principle of the framework for developing the virtual laboratory:

The standard called Functional Mock-up Interface (FMI) defines how different simulation models realized in different simulation environments may be integrated in common simulations. Based on the extensive use of this standard in other industry segments (e.g., automotive and maritime industries), and the ability to handle models implemented in different systems/tools (see https://www.fmi-standard.org/tools for a list of eligible tools), FMI was chosen as a basic framework for model integration in the AQUAEXCEL^{2020} virtual laboratory and continued to work as our foundation in AQUAEXCEL3.0.

FMI defines an interface to be implemented by executables called Functional Mock-up Units (FMU) which contain the submodels. The FMI functions can then be used by a simulation environment to create one or more instances of an FMU and simulate them, typically together with other models. An FMU may contain its own solver, in which case it is possible to use *FMI for Co-Simulation*, where the submodels communicate by exchanging output values at each communication time step. Alternatively, if the FMU does not contain a solver, it is recommended to use *FMI for Model Exchange*, where the simulation environment connects two or more submodels to a common solver. Figure 1 illustrates the co-simulation model.

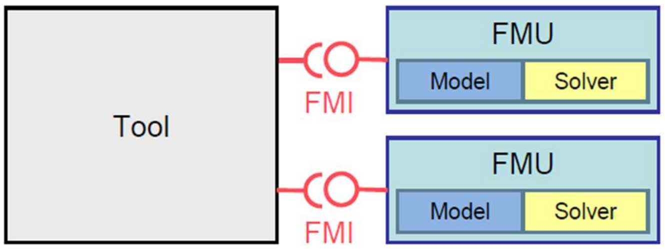


Figure 1 FMI for Co-Simulation (From Blochwitz 2014)

Irrespective of whether FMI for Co-Simulation or Model-Exchange is used, the integration between and execution of FMUs is governed by an FMI-master application. This application is responsible for synchronizing the submodels at regular *communication time steps*, and for collecting all outputs and assigning all inputs of all FMUs in the system. The information flow between submodels that are to be interconnected through their respective inputs and outputs is thus maintained by the FMI-master algorithm rather than being realized as direct information exchange between the FMUs containing them.

In AQUAEXCEL^{2020} we used an open source integrator called Coral to integrate and run the different models. Work on this code has since been discontinued, so in AQUAEXCEL3.0 we switched to another open source integrator called libcosim developed by the Open Simulation Platform (OSP) initiative[1]. The OSP initiative also provides a closed source co-simulation tool free of charge that we are using as our main component for the examples in the virtual laboratory. This tool handles setting up, combining, simulating and presenting results of the simulations.

[1] https://open-simulation-platform.github.io/

4. Integrated models

The following numerical models are the main components to be included in the virtual laboratory developed in task 4.1 of AQUAEXCEL3.0.

- Behaviour model of Atlantic salmon. (Task 4.1.2)
- Modelling of hydrodynamic flow fields in tanks and cages (task 4.1.3)
- Growth, nutrition and waste production models for different fish species (task 4.1.4)
- Water quality and water treatment modelling (task 4.1.5)

These models have been realized as FMUs using the principle of Co-Simulation. Detailed outlines, validation and discussions on the models delivered in tasks 4.1.2, 4.1.3, 4.1.4 and 4.1.5 can be found in Endresen et al. (2024), Alver et al. (2024), Lika et al. (2024) and Gyalog et al. (2024), respectively. In this section, we give a short summary of these deliverables for completeness of this report, most of the summaries are taken from Bjørnson (2020) for work done in AQUAEXCEL2020, while new additions are added for the work done in AQUAEXCEL3.0. We also describe how they are integrated in the Virtual Laboratory.

4.1. Behaviour model

The objective of the behaviour model presented by Endresen et al. (2024) was to develop an individual-based fish behaviour model for simulating full-scale fish populations (e.g., 200 000 fish) in open sea cages and closed tanks. It was based on the further development of an existing model for salmon behaviour in open net cages such that it could be applied to other TNA-infrastructures.

The behaviour model considers different factors or parameters that affect the fish swimming behaviour. These are the cage limits or boundaries including the water surface with waves and cage walls affected by currents. The model also considers how the fish react to the current passing through the cage or induced in a tank as well as three dynamic parameters, which are feeding, temperature and light. Temperature and light will only vary vertically in the model. The fish will also try to avoid colliding with each other, which is modelled through defining a preferred range of distance to neighbouring fish and having a set of rules to decide how the fish will react dependent on how close the fish is to other individuals. An example simulation for a production cage can be seen in Figure 2.

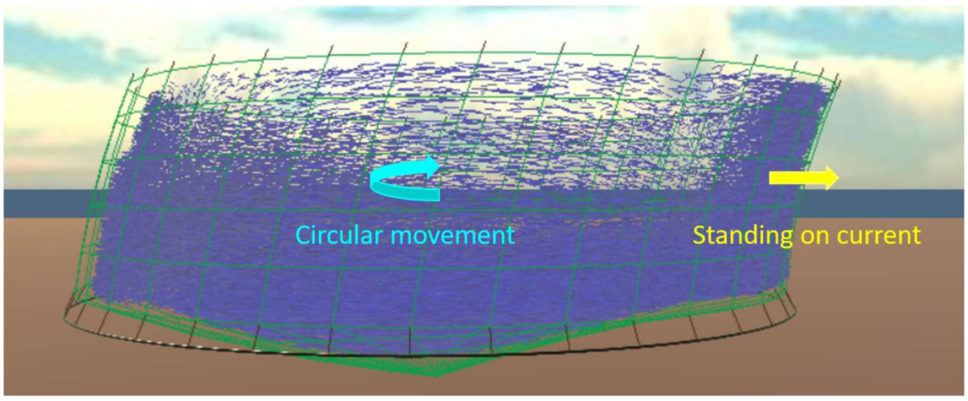


Figure 2: Fish population in water current, from Endresen et al. 2024

The model is implemented in a simulation software called FhSim, developed by the project partner SINTEF Ocean. The software contains an option to package the entire simulation and export it as an FMU. One FMU was created for cages and one for tanks. These models consider factors like waves, current, number and size of fish as well as feeding. They output variables including fish density, swim speed and pellet ingestion.

One major obstacle to integrating the behaviour model with the other models was the difference in timesteps. The behaviour model operates in the magnitude of seconds per time step while the other models operate in the magnitude of hours. The solution, described in Endresen et al. (2024), and will be described further in chapter 4.5 was to develop a surrogate model and wrap this in an FMU.

4.2. Flow model

The objective of the flow field model presented by Alver, M. O. et al. (2020, 2024) was to represent the water currents within the production unit (fish cage or tank), presenting key information related to the current to the other model components. The developed flow field model uses one approach for current in tanks – precomputed flow fields from a CFD model (see Figure 3) – and another for open sea cages – current profiles depending on ambient current conditions. Previously, three tanks and four sea cage locations were presimulated to provide flow fields. Based on a survey of tank sizes across the AQUAEXCEL consortium this was extended with two new tank sizes and more variations in design and flow rates to provide a better coverage of the tank designs. The model interacts with the other model components either by providing the current speed and direction vector for given locations, or through providing descriptive numbers for the overall flow field in the production unit.

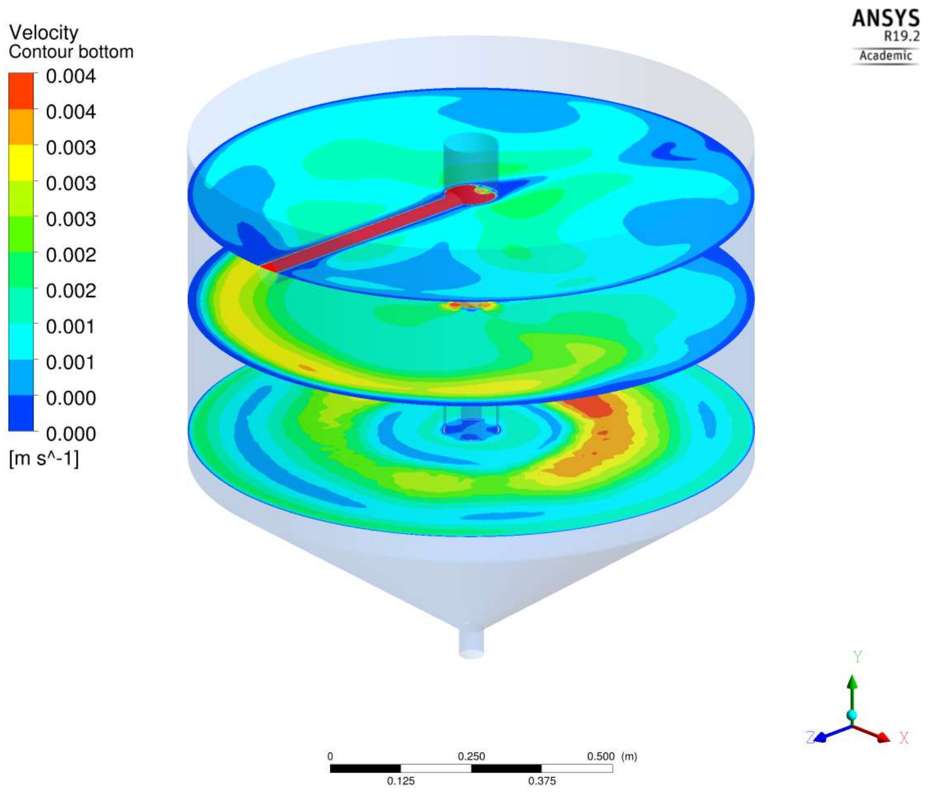


Figure 3 Example of velocity flow field in HCMR tank, computed using Ansys Fluent. (From Alver 2020)

The flow field model component is written in C++ and packaged both as a Functional Mock-up Unit (FMU), and as a dynamic library (DLL). The model utilizes the NetCDF 4 library[2] which provides functionality for reading and writing NetCDF files. The main interaction of the flow model with other models was through the behavior model. It was determined that this was suboptimal using the FMU framework, thus the flow model is accessed programmatically directly from the behavior model.

4.3. Growth model

The AquaFishDEB model presented in Lika et al. (2020, 2024) is designed to predict growth, feed consumption and waste production for Atlantic salmon, seabream, rainbow trout, European sea bass and pikeperch. Specifically, the model predicts 1) fish growth for different feeds (quantity and composition) and water temperature, and 2) oxygen consumption and waste production (nitrogen, CO_2, solids) at different fish sizes, temperatures, feed rations and diet compositions for individual fish or groups.

The model is based on the Dynamic Energy Budget (DEB) theory for metabolic organization, which provides a conceptual and quantitative framework to study the whole life cycle of individual animals while making explicit use of energy and mass balances (Lika et al., 2020). The model covers all life stages of a fish (including larvae, juveniles and market size fish) and is explicitly tied with feed and temperature. It accommodates different feeding strategies (e.g., ad libitum or restricted, feeding frequency, adaptive feeding) and feed compositions. The output of the model includes fish growth characteristics (number of fish, mean body-size, total biomass, feed intake, specific growth rate and feed conversion efficiency), waste production (fecal dry matter and nitrogen-loss, as well as non-fecal nitrogen loss) and gaseous exchange (O_2 consumption and CO_2 production).

As presented in Lika et al. (2020), predictions made by the AquaFishDEB model are the end products of a two-step modeling procedure (Figure 4). The first step involves the parameterization of the DEB model for each species. In the second step, the DEB parameters are used in the prototype AquaFishDEB model that then simulates the dynamics for a group of fish exposed to user input regarding fish and feed characteristics, and the specified experimental conditions. The developed prototype model is thus able to predict growth, feed consumption and waste production for the fish.

The first step of the modeling procedure is described for three chosen species in Lika et al. (2020): rainbow trout, seabream and Atlantic salmon. The same procedure is described for new species of AQUAEXCEL3.0 in Lika et al. (2024): European seabass and pikeperch.

[2] NetCDF libraries are open source and can be downloaded at https://www.unidata.ucar.edu/downloads/netcdf/index.jsp.

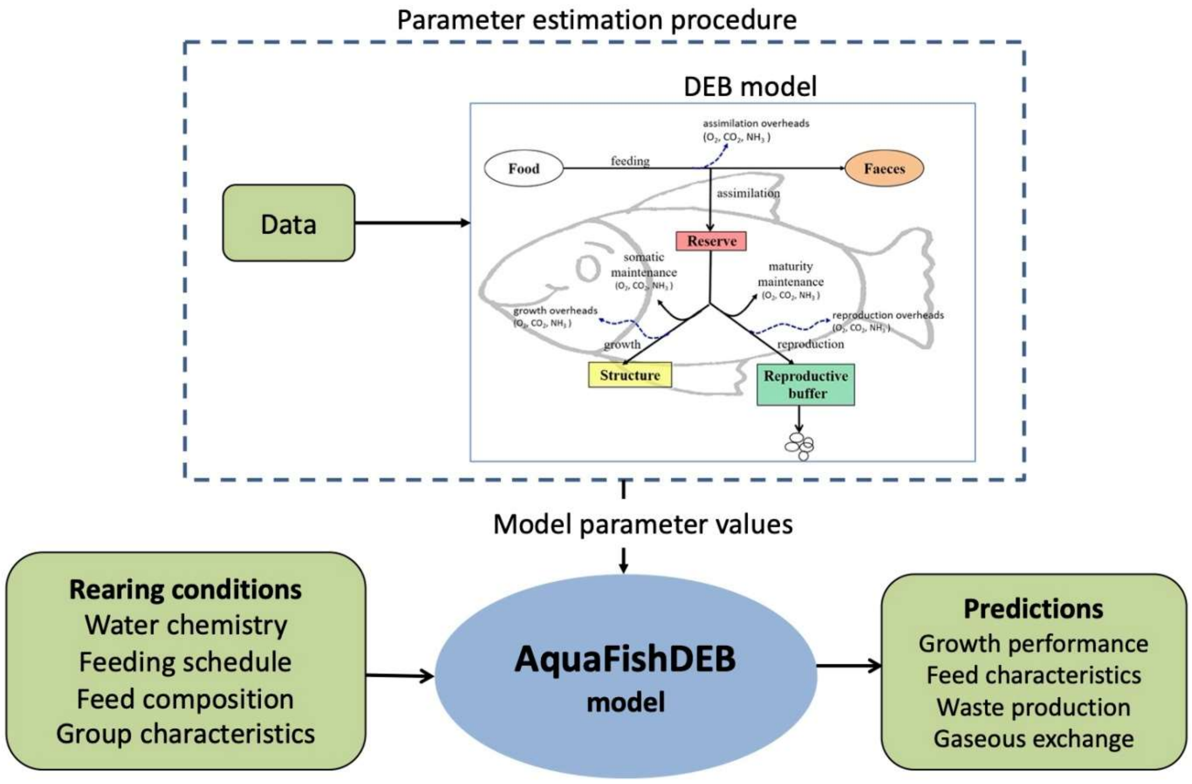


Figure 4 Schematic representation of the two-step procedure for the development of the AquaFishDEB model. (From Lika, K. et al. (2020))

The AquaFishDEB model was first implemented as a stand-alone model in Matlab, and all model tuning, verification and validation was done using this version of the model. After this, the Matlab code was converted into C++ using Matlab Coder, thus enabling simulations of the model in C++. This code was then linked into an FMU-interface implemented in C++, that was compiled, resulting in an FMU containing the AquaFishDEB-model. The functionality of this version of the model was finally verified by comparing model outputs from the FMU with those obtained with the original Matlab implementation. Since pikeperch differs much in the eating habits from the other species, this species was given a separate FMU.

4.4. Water quality model

Abbink et al. (2020) presents a model that predicts the water quality and water treatment effects in research infrastructures such as tanks. The model was designed as a generic tool that users of research facilities could use prior to the start of an experiment to predict the expected water quality during the experiment. In addition, the model could be a tool for (re-) designing systems so that they result in the desired water quality for the experiment envisioned. This makes the model a potential tool for teaching TNA users, research infrastructure technicians and others involved the principles of water quality control in fish culture units.

The sub-model computes water quality based on input parameters that describe the production plan and the experimental design. These inputs may either come from the growth model or be provided as direct inputs to the model if they are known in advance. The model outputs describe water quality using the most crucial parameters related to ammonia and nitrate in the system (tanks and filters). Figure 5 provides an overview of the major components of the model. For each communication time step, the model calculates values such as ammonia production by the fish, nitrification rate, nitrification capacity, ammonia load to the biofilter, ammonia removal rate, ammonia concentration in the water, nitrate production, and nitrate in the tanks. Gyalog et al. (2024) describes an addition to the model where a new component specializing in computing CO_2 was added, see Figure 6.

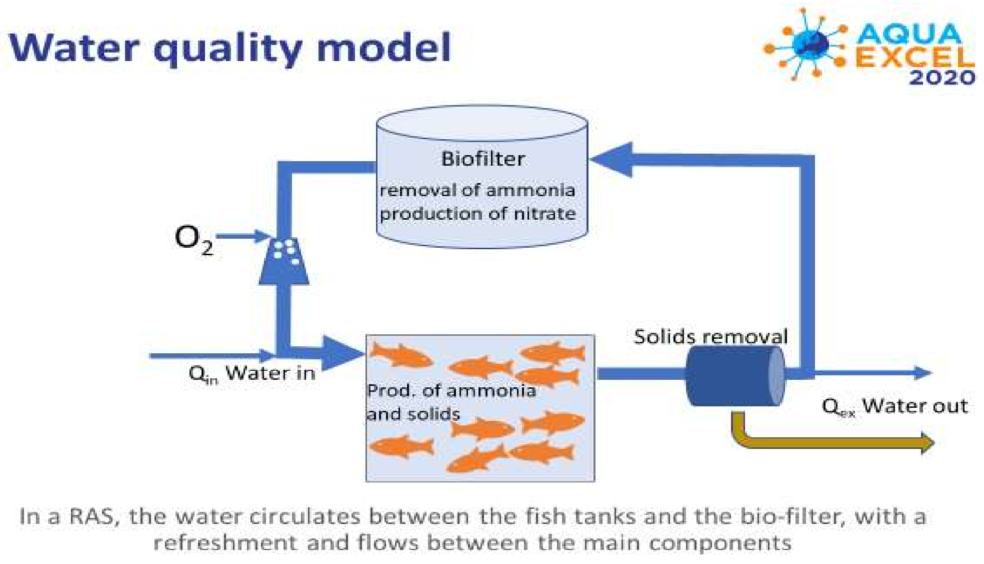

Water quality model
AQUA EXCEL 2020
Biofilter
removal of ammonia
production of nitrate
O2
Solids removal
Qin Water in
Prod. of ammonia and solids
Qex Water out
In a RAS, the water circulates between the fish tanks and the bio-filter, with a refreshment and flows between the main components

Figure 5 Water quality model (From Abbink et al. 2020)

The water quality model was first implemented in Excel, and all tests and tuning of the model was done using this implementation. For the final version of the water quality model, the model was fully reimplemented in C++ to enhance performance and convert it into an FMU.

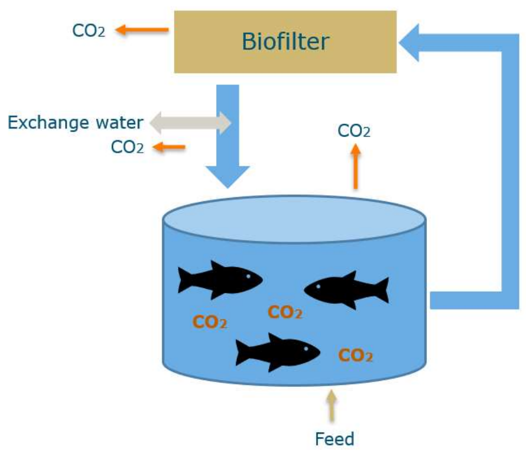


Figure 6: Design of the CO2 module, from Gyalog (2024)

Gyalog et al. (2024) also describes the development of a pond model. It was determined early in the project that this model was significantly different from the other models and not suitable for integration. It was therefore determined that this model would be a standalone model.

4.5. Interactions

The interactions between growth and water treatment were developed and described in Bjørnson (2020) and is shown in Figure 7. The growth model describes the fish and feeding regime and the water treatment takes the output from the growth model as input for calculating the quality of the environment for the fish. With the new solution chosen for integration of FMUs in AQUAEXCEL 3.0 it is possible to create more advanced combinations of models as shown in Figure 8. The example shows two groups of fish in the same tank, simulating a different feeding regime for each group to simulate the effect of “winner” and “loser” fish.

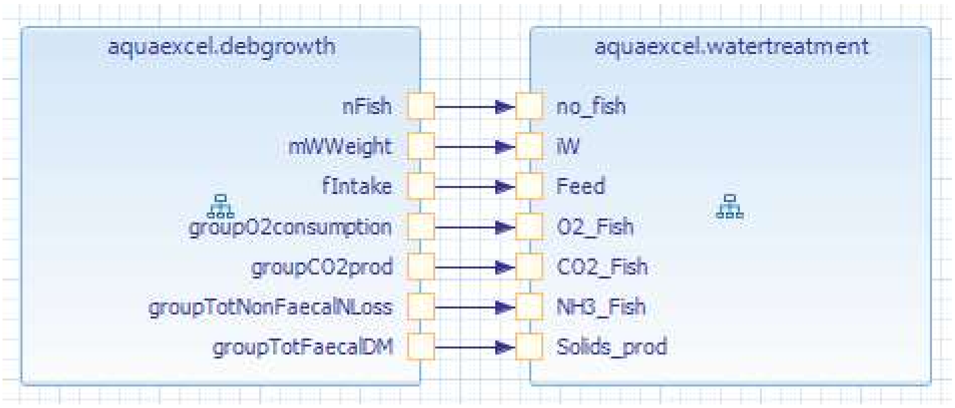


Figure 7: growth coupled to water treatment

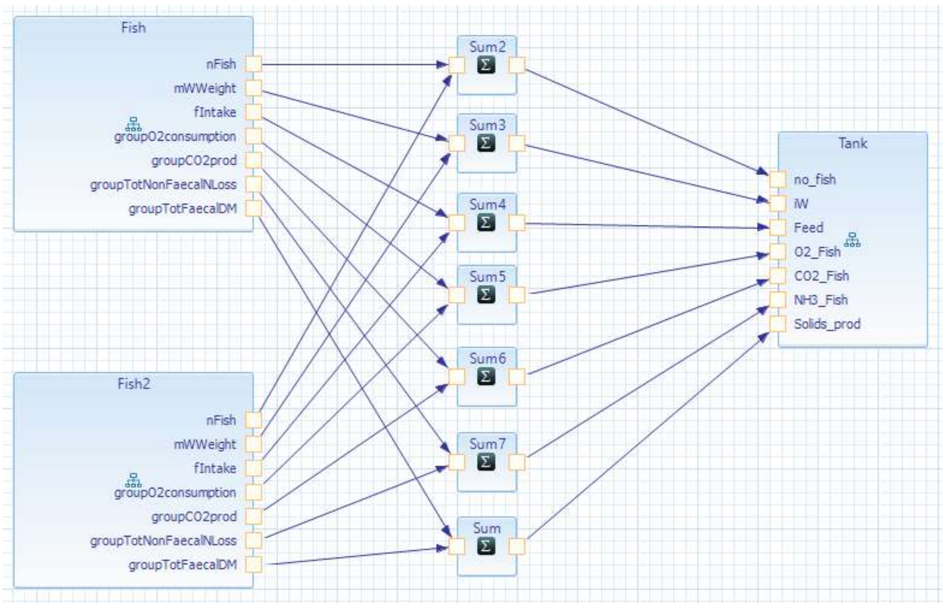


Figure 8: Advanced example, two growth models coupled to water treatment

As mentioned in chapter 4.1, one major obstacle for co-simulation of behaviour and the other models was the difference in timesteps. The solution was to create a surrogate model, the process of creating the surrogate model is further described in Saad et al. (2023) and Endresen et al. (2024).

The surrogate model is a substitute or surrogate for the real behaviour model which can estimate results based on cage conditions and pass those to the growth model while also being able to communicate in the same time interval as the growth model. It was created based on 2000 different simulations with the behaviour model within a given input parameter set. As such, the results of the surrogate model contains both the behaviour and flow model. A schematic for the surrogate modelling process is presented in Figure 9. The surrogate model can be used instead of the actual behaviour model if the simulation inputs and net cage metrics are within the variable space the surrogate model was developed for.

By developing this surrogate model, we can get estimates in seconds rather than hours when combining the model with growth. To make the two models talk to each other the growth model added two more inputs: fishAppetite which describes how much of the feed the fish are able to eat in the given timestep, and fishVelocity which affects the energy use of the growth model. To get estimates on appetite and speed from the surrogate model the growth model has to pass predictions of upcoming feeding periods and how much feed is planned, the number of fish and their weight and stomach content. With this information the surrogate model can predict the swimming speed and how much of the planned feed the fish will consume. Figure 10 illustrates the coupled FMU system.

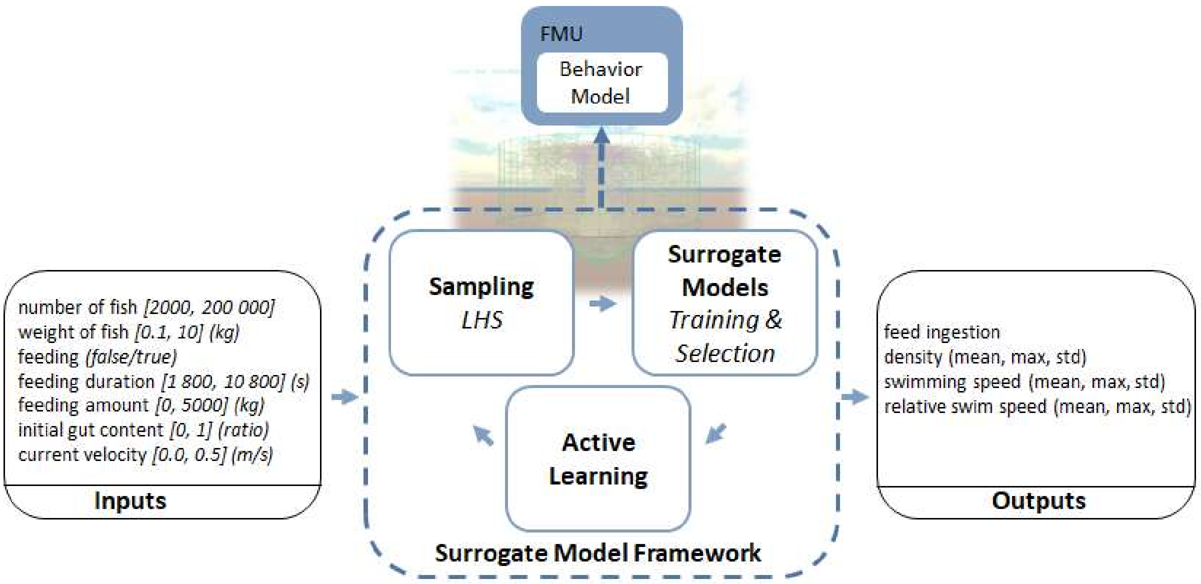

FMU
Behavior Model
Sampling
LHS
Surrogate Models
Training & Selection
Active Learning
Surrogate Model Framework
number of fish [2000, 200 000]
weight of fish [0.1, 10] (kg)
feeding (false/true)
feeding duration [1 800, 10 800] (s)
feeding amount [0, 5000] (kg)
initial gut content [0, 1] (ratio)
current velocity [0.0, 0.5] (m/s)
Inputs
feed ingestion
density (mean, max, std)
swimming speed (mean, max, std)
relative swim speed (mean, max, std)
Outputs

Figure 9: Schematics describing workflow of surrogate model development, input and output parameters of the surrogate model. Figure from Saad et al. (2023).

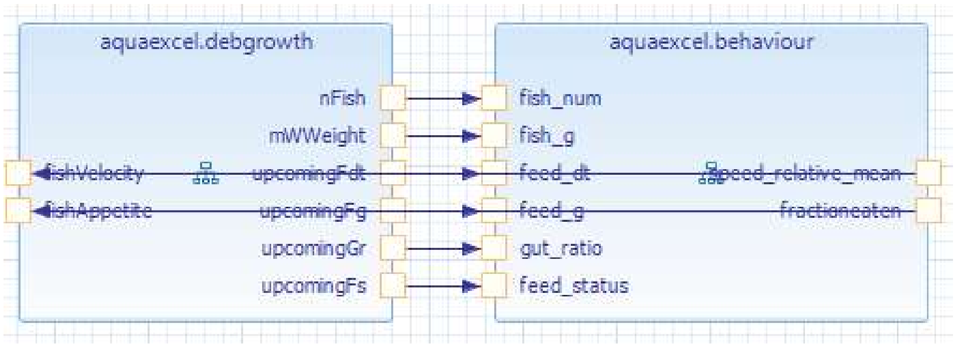
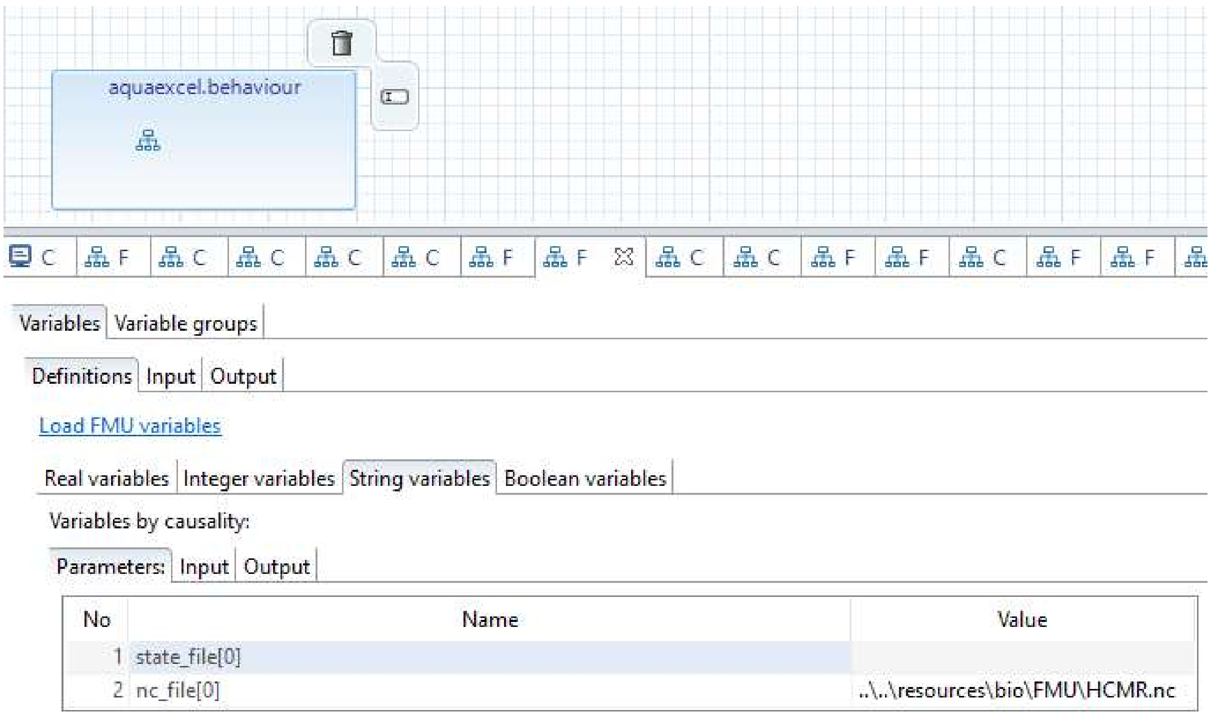


Figure 10: Surrogate behaviour model coupled to growth

As mentioned in chapter 4.2, the flow model provides a set of current profiles for the entire cage or tank volume. The behaviour model simulates up to 200.000 fish and making an FMU call for each of these to ask for the current at their position would be too time consuming. Instead, the flow model is integrated directly into the behaviour FMU so the calls can be executed internally and thus speed up the communication. Figure 11 shows the behaviour model with the possibility to directly specify where the current profile file is located.



No	Name	Value
1	state_file[0]	
2	nc_file[0]	..\..\resources\bio\FMU\HCMR.nc

Figure 11: Behaviour with integrated flow model. The input presimulated flow file can be specified as nc_file.

Figure 12 provides an overview of the connections between all models developed in WP4.1 during AQUAEXCEL3.0.

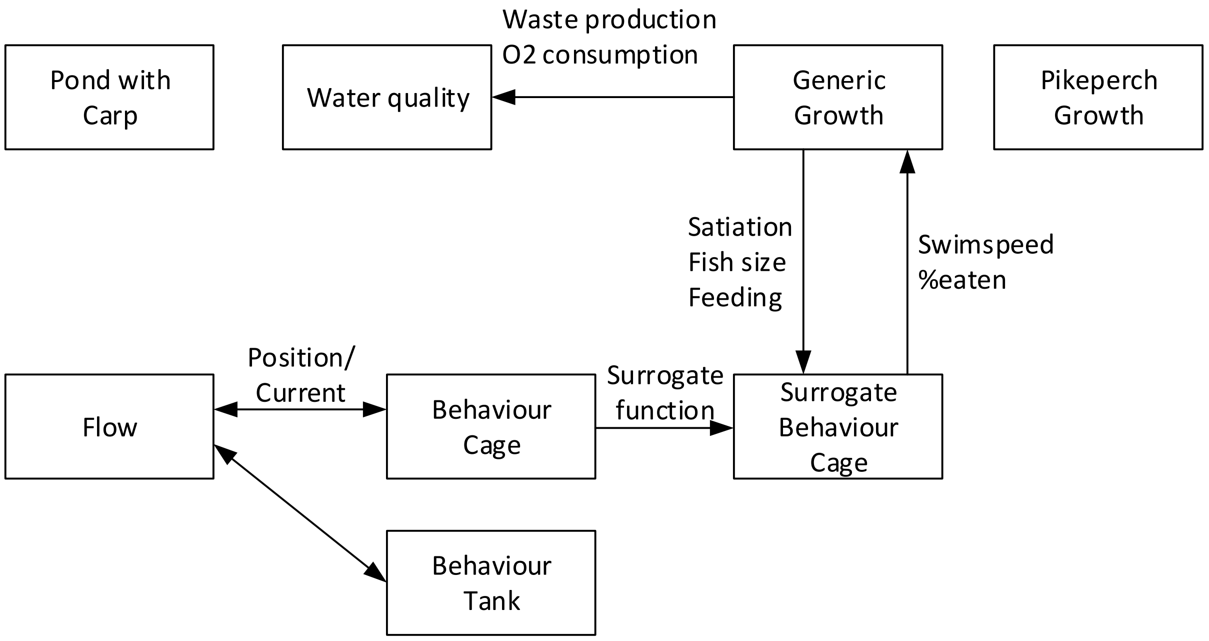


Figure 12: Overview of connections between models

5. Virtual laboratory version 3.0

The virtual laboratory version 3.0 consists of three major parts: A web interface that provides advice and tutorials to users who want to set up a virtual experiment, the necessary components to set up a virtual laboratory, and a virtual assistant that provides chat functionality for advising users. The first two will be described in this chapter while we dedicate the next chapter to the artificial agent which provides the most novel functionality compared with the virtual laboratory version 2.0.

5.1. Web

The user interface is implemented in Django[3], which is running in a Python environment which in turn is running on top of a MySQL database. To improve the user experience of the Virtual Laboratory, we make use of Bootstrap[4], an open source toolkit for developing with HTML, CSS and JS. To have good modularity in the user interface, we have separated functionality into different applications running in a common project environment. Figure 13 provides an overview of the data packages currently running on the system. The Tutorials app handles tutorials for different simulations, Downloads handles all downloads, the Assistant packages the communication interface for the intelligent agent, and Admin provides an interface for admin users.

[3]https://www.djangoproject.com/
[4] https://getbootstrap.com/

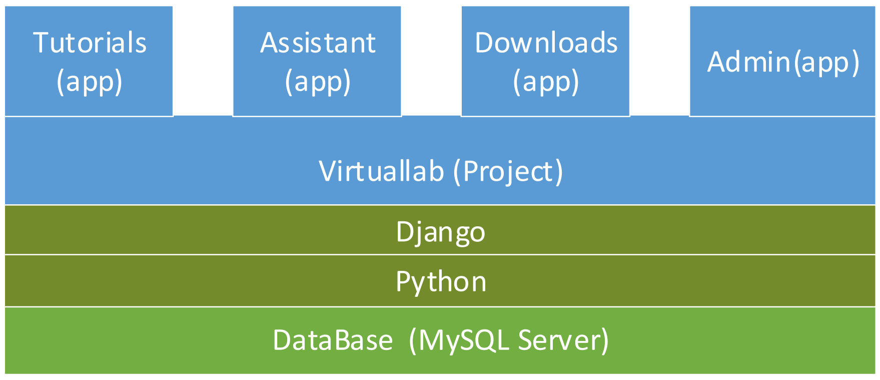


Figure 13: Modules of the website

Figure 14 describes the overall flow of information in the Virtual Laboratory. To understand the flow, we need to go into the communication flow of the Django framework we are building upon. The Django framework closely follows the Model View Controller Architecture[5]. However, since the Control part is covered by the framework, and most of the action happens in the views and template layer, it is often referred to as a Model Template View architecture. The Model layer handles everything related to the data: access, validation, behavior and relationships. The template layer contains presentation logic, how content should be presented to the user through Web pages or other types of documents. The View layer contains the business logic, it functions as a bridge between the data in the models and the presentations in the templates.

Communication between the Model layer and the database is abstracted away in the Django framework. We only need to access the data in the Model layer and the underlying framework will update the database for us. Functions in the View layer has full access to variables and methods in the Model layer. Data from the View may be passed to appropriate templates which are then rendered to be presented for the user in a browser. The user then provides inputs which are transferred through the URL dispatcher back into an appropriate View.

[5] https://en.wikipedia.org/wiki/Model-view-controller

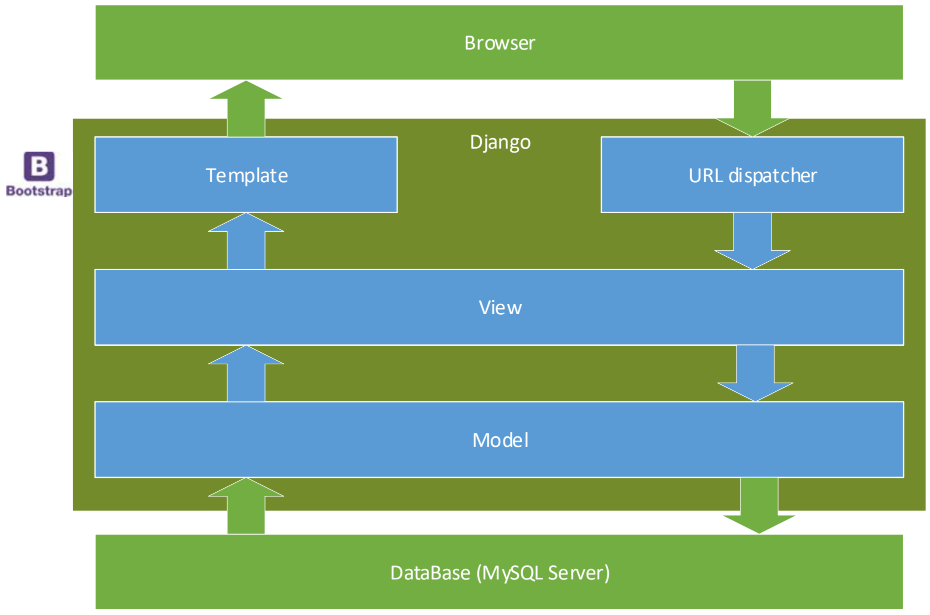


Figure 14: High level communication

5.2. Kopl and FMUs

In the virtual laboratory 2.0, we implemented a web interface for running a simulation. This traded flexibility for usability. In version 3.0 we wanted to increase flexibility of the tools we provided so that users could create their own virtual laboratories independent of our web solution and our predetermined configurations. Thus, the decision was made to use the Open Simulation Platform[6] framework. In addition to providing an open source numerical solver for our FMU the platform also provides a simulation tool that can be downloaded free of charge, Kopl[7].

With this tool and the provided FMU's a user can build their own virtual laboratory and perform virtual experiments. Figure 15 provides an overview of the major modules that can be downloaded from the website of the virtual laboratory.

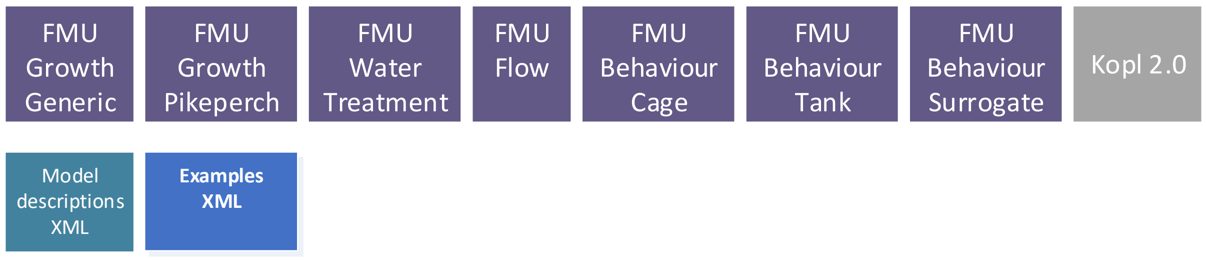
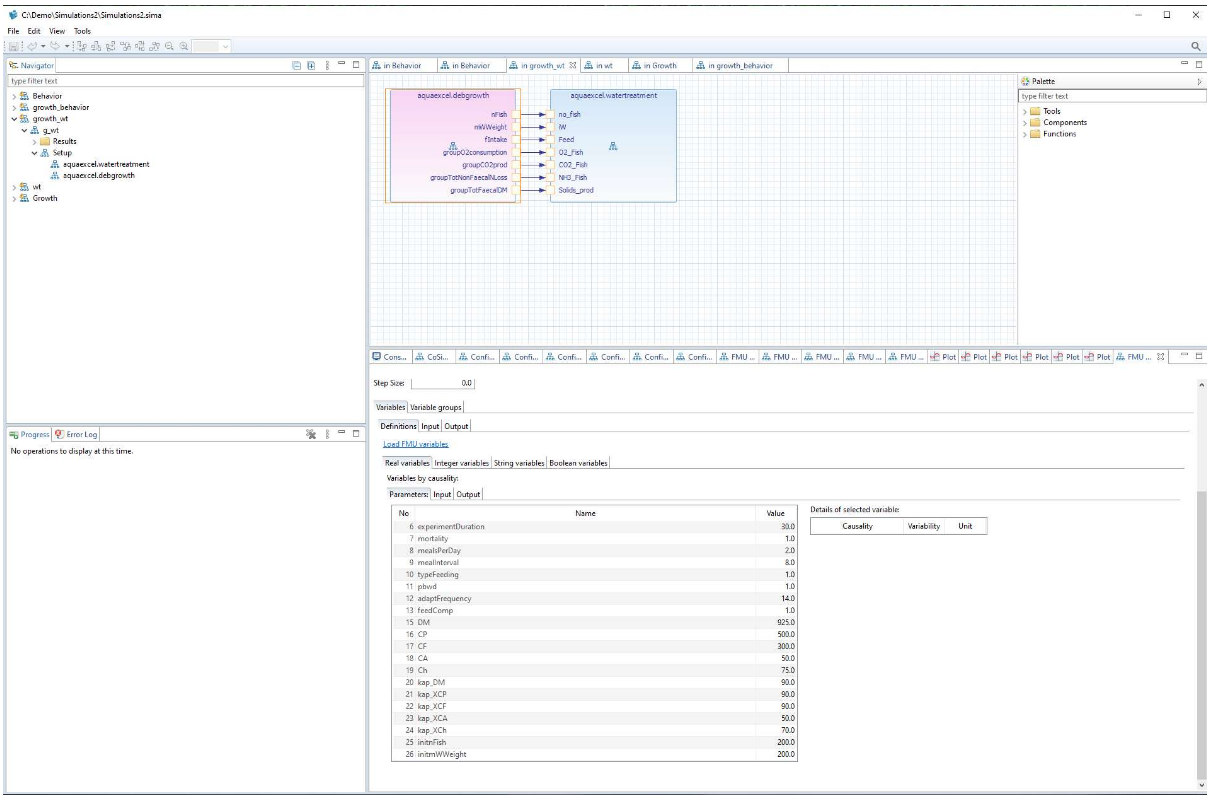


Figure 15: Modules for creating a virtual lab

With Kopl (Figure 16) a user can drag and drop different FMU's, create multiple experimental setups, simulate them and study the results. They can also create their own FMU's and connect these to the FMU's provided through AQUAEXCEL. In essence, this provides the "direct framework access"

[6] https://opensimulationplatform.com/
[7] https://open-simulation-platform.github.io/kopl

described for the virtual laboratory 2.0 in Bjørnson (2020) but with a graphical user interface to support the user through the process.



Step Size: 0.0

Variables | Variable groups

Definitions | Input | Output

Load FMU variables

Real variables | Integer variables | String variables | Boolean variables

Variables by causality:

Parameters: | Input | Output

No	Name	Value
6	experimentDuration	30.0
7	mortality	1.0
8	mealsPerDay	2.0
9	mealInterval	8.0
10	typeFeeding	1.0
11	pbwd	1.0
12	adaptFrequency	14.0
13	feedComp	1.0
15	DM	925.0
16	CP	500.0
17	CF	300.0
18	CA	50.0
19	Ch	75.0
20	kap_DM	90.0
21	kap_XCP	90.0
22	kap_XCF	90.0
23	kap_XCA	50.0
24	kap_XCh	70.0
25	initnFish	200.0
26	initmWWeight	200.0

Details of selected variable:

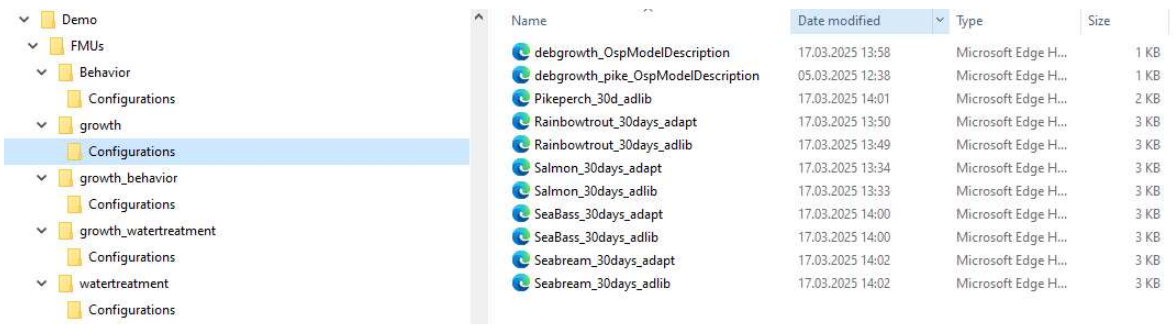
Causality	Variability	Unit

Figure 16: Kopl example, setting up an experiment

This solution increased flexibility but lowered usability. To compensate for this loss of useability we have provided preconfigured cases that users can use as a starting point to tune their experiments (see Figure 17 for how the configurations are structured for each example, and Figure 18 for how a typical configuration file is set up). Another benefit of using the Kopl tool is that anyone who wishes to provide a virtual version of their laboratory can set it up in the kopl-tool and export it with configurations and FMUs to any potential user of the infrastructure.

We've also created tutorials in the online part of the virtual lab explaining both in text and video how to set up an experiment. In addition, the virtual assistant has been trained on the documentation of all models to provide a chat interface to users wanting to set up their virtual experiment.

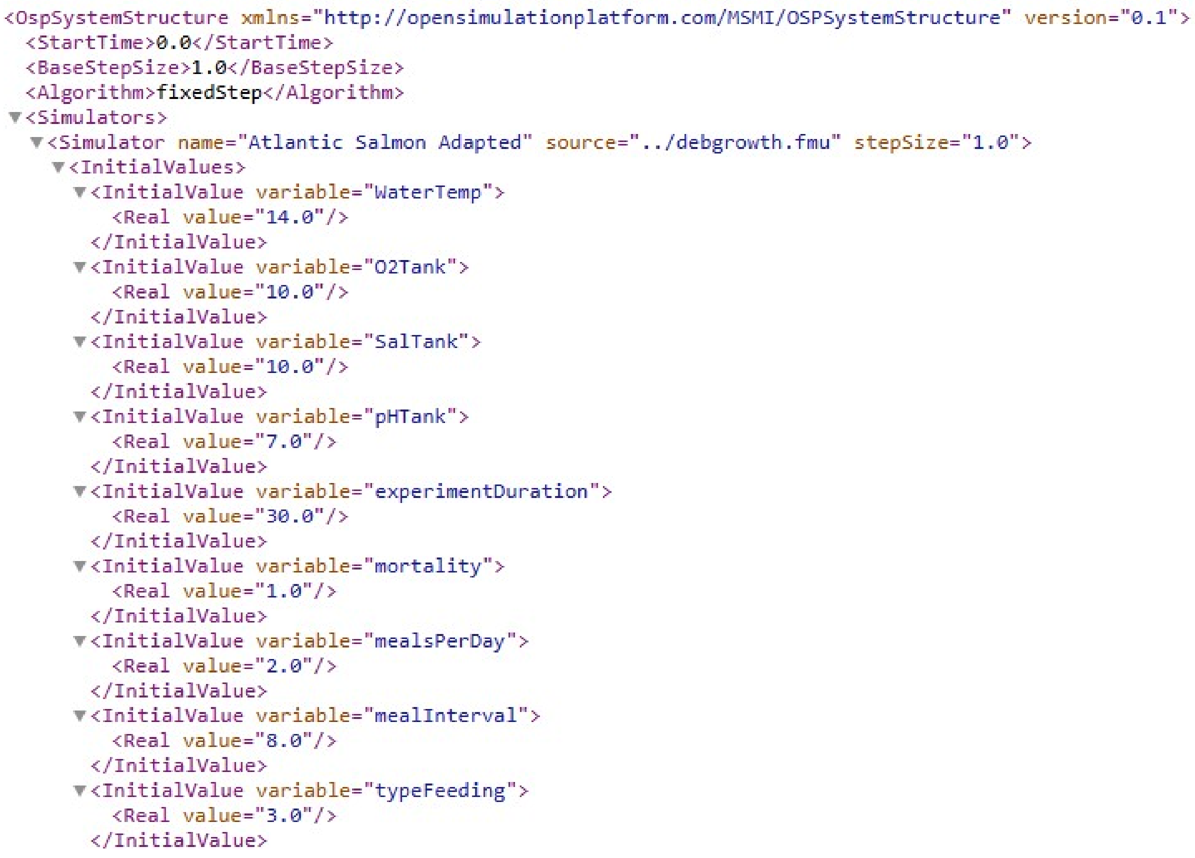
Documentation of all models are provided in modeldescription.xml files as per the FMI standard. These files contain the explanation of all variables in and out of the models as well as the units of the parameters being passed.



```
Demo
  FMUs
    Behavior
      Configurations
    growth
      Configurations
    growth_behavior
      Configurations
    growth_watertreatment
      Configurations
    watertreatment
      Configurations
```

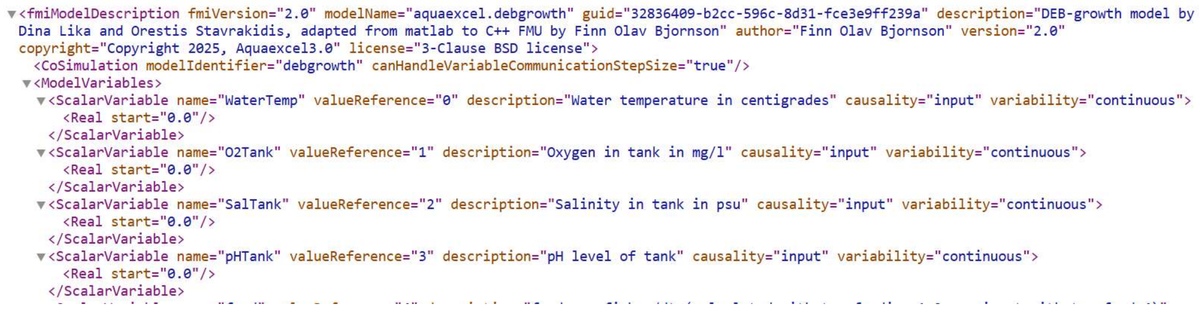
Name	Date modified	Type	Size
debgrowth_OspModelDescription	17.03.2025 13:58	Microsoft Edge H...	1 KB
debgrowth_pike_OspModelDescription	05.03.2025 12:38	Microsoft Edge H...	1 KB
Pikeperch_30d_adlib	17.03.2025 14:01	Microsoft Edge H...	2 KB
Rainbowtrout_30days_adapt	17.03.2025 13:50	Microsoft Edge H...	3 KB
Rainbowtrout_30days_adlib	17.03.2025 13:49	Microsoft Edge H...	3 KB
Salmon_30days_adapt	17.03.2025 13:34	Microsoft Edge H...	3 KB
Salmon_30days_adlib	17.03.2025 13:33	Microsoft Edge H...	3 KB
SeaBass_30days_adapt	17.03.2025 14:00	Microsoft Edge H...	3 KB
SeaBass_30days_adlib	17.03.2025 14:00	Microsoft Edge H...	3 KB
Seabream_30days_adapt	17.03.2025 14:02	Microsoft Edge H...	3 KB
Seabream_30days_adlib	17.03.2025 14:02	Microsoft Edge H...	3 KB

Figure 17: Directory structure, FMUs with different configurations



```
<OspSystemStructure xmlns="http://opensimulationplatform.com/MSMI/OSPSystemStructure" version="0.1">
  <StartTime>0.0</StartTime>
  <BaseStepSize>1.0</BaseStepSize>
  <Algorithm>fixedStep</Algorithm>
▼<Simulators>
  ▼<Simulator name="Atlantic Salmon Adapted" source="../debgrowth.fmu" stepSize="1.0">
    ▼<InitialValues>
      ▼<InitialValue variable="WaterTemp">
          <Real value="14.0"/>
        </InitialValue>
      ▼<InitialValue variable="O2Tank">
          <Real value="10.0"/>
        </InitialValue>
      ▼<InitialValue variable="SalTank">
          <Real value="10.0"/>
        </InitialValue>
      ▼<InitialValue variable="pHTank">
          <Real value="7.0"/>
        </InitialValue>
      ▼<InitialValue variable="experimentDuration">
          <Real value="30.0"/>
        </InitialValue>
      ▼<InitialValue variable="mortality">
          <Real value="1.0"/>
        </InitialValue>
      ▼<InitialValue variable="mealsPerDay">
          <Real value="2.0"/>
        </InitialValue>
      ▼<InitialValue variable="mealInterval">
          <Real value="8.0"/>
        </InitialValue>
      ▼<InitialValue variable="typeFeeding">
          <Real value="3.0"/>
        </InitialValue>
```

Figure 18: Example configuration



```
▼<fmiModelDescription fmiVersion="2.0" modelName="aquaexcel.debgrowth" guid="32836409-b2cc-596c-8d31-fce3e9ff239a" description="DEB-growth model by
 Dina Lika and Orestis Stavrakidis, adapted from matlab to C++ FMU by Finn Olav Bjornson" author="Finn Olav Bjornson" version="2.0"
 copyright="Copyright 2025, Aquaexcel3.0" license="3-Clause BSD license">
   <CoSimulation modelIdentifier="debgrowth" canHandleVariableCommunicationStepSize="true"/>
 ▼<ModelVariables>
   ▼<ScalarVariable name="WaterTemp" valueReference="0" description="Water temperature in centigrades" causality="input" variability="continuous">
      <Real start="0.0"/>
    </ScalarVariable>
   ▼<ScalarVariable name="O2Tank" valueReference="1" description="Oxygen in tank in mg/l" causality="input" variability="continuous">
      <Real start="0.0"/>
    </ScalarVariable>
   ▼<ScalarVariable name="SalTank" valueReference="2" description="Salinity in tank in psu" causality="input" variability="continuous">
      <Real start="0.0"/>
    </ScalarVariable>
   ▼<ScalarVariable name="pHTank" valueReference="3" description="pH level of tank" causality="input" variability="continuous">
      <Real start="0.0"/>
    </ScalarVariable>
```

Figure 19: modeldescription.xml example

6. Artificial agent version 1.0

A key component of the virtual lab's web structure is the Assistant Module, which features an AI-powered chatbot designed to streamline user interaction. Initially, the platform employed a BERT-based chatbot to interpret natural language queries and guide users in configuring aquaculture simulations (see Figure 20). BERT (Bidirectional Encoder Representations from Transformers) was selected for its deep contextual language understanding, enabling it to analyze user input effectively. The model was fine-tuned on 156,060 phrases categorized into five distinct simulation setup classes (*Growth, Water Treatment, Behavior, Growth-Water Treatment, and Growth-Behavior*), achieving 82% accuracy. Users could describe their simulation scenario in free text, and BERT would direct them to the corresponding tutorial from these predefined classes. To further refine the system, a feedback mechanism collected user interactions, gradually expanding the training dataset to 256,160 samples for ongoing improvement.

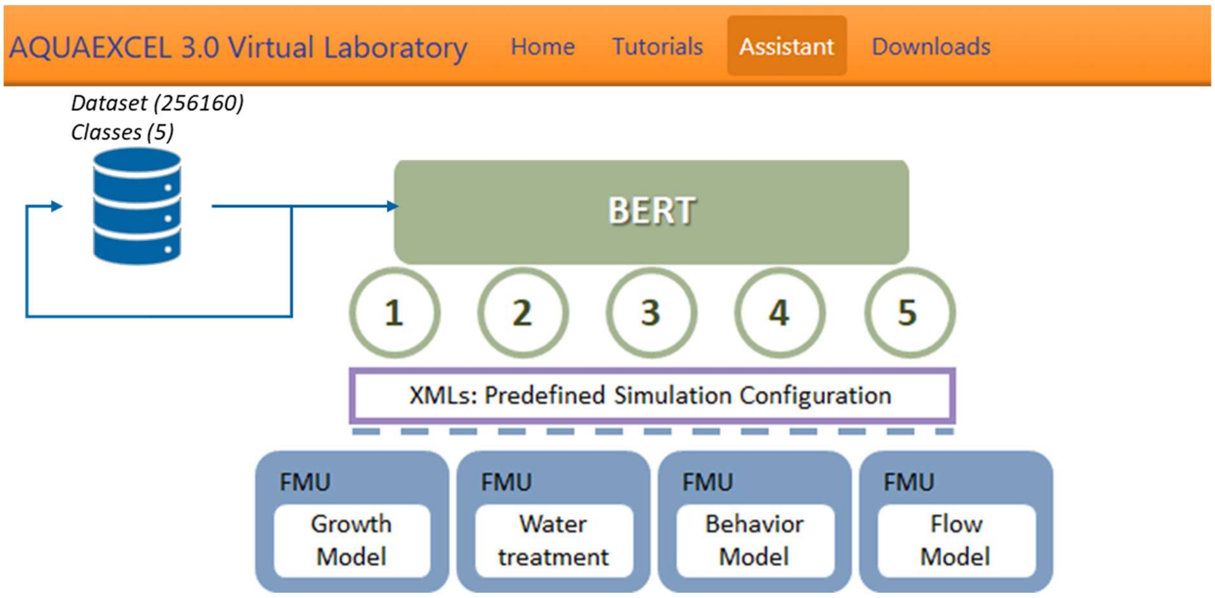


Figure 20 The assistant module version 0.5: the BERT is the core component of the chatbot

To enhance responsiveness and adaptability, the BERT-based assistant was later replaced with a GenAI module powered by OpenAI's API advanced language models, integrated with a Neo4j knowledge graph. This transition followed a three-step pipeline to ensure structured knowledge extraction, enhanced reasoning, and seamless chatbot integration (see Figure 21):

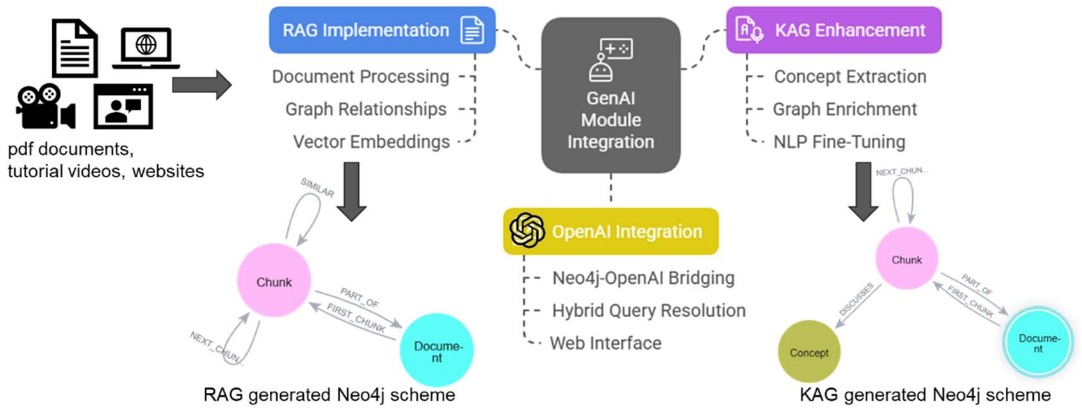


Figure 21. Enhancing AI Assistant with GenAI Integration

Step 1: RAG Implementation in Neo4j

The first phase involved constructing a Retrieval-Augmented Generation (RAG) system to ground responses in domain-specific documents. Key steps included:

- Document Processing: Uploaded materials (e.g., tutorials, videos and urls) were split into chunks and stored in Neo4j Aura as nodes.
- Graph Relationships: Chunks were linked via relationships like `FIRST_CHUNK`, `NEXT`, and `PART_OF` to preserve document structure and enable contextual retrieval.
- Vector Embeddings: Each chunk was embedded using OpenAI's models (e.g., `gpt-4o`) to enable semantic search during query resolution.

This RAG framework allowed the system to retrieve relevant document snippets dynamically, ensuring responses were anchored in authoritative content.

Step 2: Knowledge-Augmented Graph (KAG) Enhancement

To refine the RAG's output, a Knowledge-Augmented Graph (KAG) was built by:

- Concept Extraction: The system identified domain-specific entities (e.g., fish species, simulation parameters, software tools) and relationships (e.g., `SIMULATES`, `DEPENDS_ON`).
- Graph Enrichment: Additional nodes labeled `Concept` were created in Neo4j, linking them to document chunks to form a hybrid structure of raw data and abstract knowledge.
- NLP Fine-Tuning: Custom spaCy models and transformer-based classifiers were trained to improve entity/relationship detection for aquaculture-specific queries.

The KAG enabled multi-hop reasoning—e.g., linking a user's question about "salmon growth in low-oxygen water" to related concepts like hydrodynamics or aeration systems—before generating a response.

Step 3: OpenAI Integration & Chatbot Deployment

The final step replaced BERT with a GenAI-powered assistant by:

- Neo4j-OpenAI Bridging: The `llm_integration.py` module connected OpenAI's API to Neo4j, dynamically retrieving graph context (from RAG/KAG) to augment prompts.
- Hybrid Query Resolution: For each user query (e.g., "How to model algae effects on fish behavior?"), the system:
    1. Retrieved relevant chunks/concepts from Neo4j.
    2. Formulated a structured prompt combining the query, graph context, and conversation history.
    3. Generated a response using OpenAI's model, ensuring accuracy and domain relevance.
- Web Interface: The frontend (templates/fmulab/index.html) provided a chat interface where users could interact naturally, with responses now leveraging both structured knowledge and generative flexibility.

This upgrade enabled more natural, context-aware conversations while reducing reliance on manual fine-tuning.

The GenAI module offered significant improvements:

- Dynamic Knowledge: Unlike BERT's static fine-tuning, the KAG-enabled system adapted to new documents/concepts without retraining.
- Multi-Source Reasoning: Combined retrieved documents, conceptual relationships, and generative AI for nuanced answers.
- Scalability: OpenAI's API handled diverse queries beyond the original 5 simulation classes, supporting open-ended dialogue.

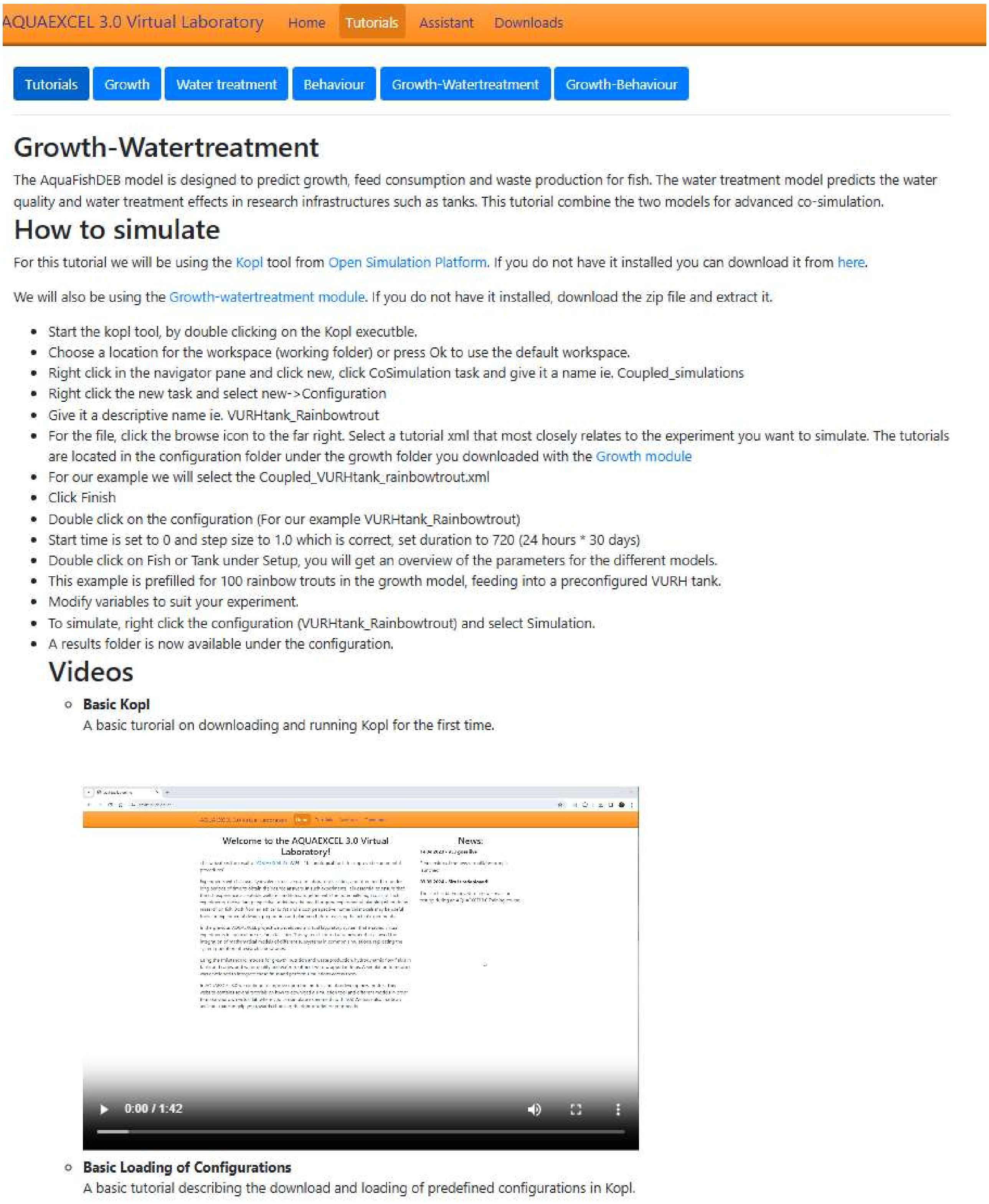
This integration transformed the assistant into a context-aware, self-improving tool, bridging unstructured user queries with structured aquaculture knowledge. By integrating OpenAI's API, the virtual lab now delivers more dynamic and scalable assistance, significantly improving the user experience.

7. Demonstration

For a more dynamic demonstration, visit https://aevirtuallab.online/ and click through the different pages. For reference we provide some examples of use in this report.

7.1. Virtual lab

Figure 22 and Figure 23 provides an example of one of the tutorials that can be found on the virtual laboratory. They contain everything a user needs to create a virtual experiment: a short description of the model or the coupled system and what it can simulate, a textual step by step instruction on how to load one of the provided configurations, tutorial videos on basic operations of the kopl tool as well as advanced instructions for the example in the tutorial, links to necessary modules that needs to be downloaded, xml descriptions of all variables in the model(s) and links to the theoretical foundation of the models.



Growth-Watertreatment

The AquaFishDEB model is designed to predict growth, feed consumption and waste production for fish. The water treatment model predicts the water quality and water treatment effects in research infrastructures such as tanks. This tutorial combine the two models for advanced co-simulation.

How to simulate

For this tutorial we will be using the Kopl tool from Open Simulation Platform. If you do not have it installed you can download it from here.

We will also be using the Growth-watertreatment module. If you do not have it installed, download the zip file and extract it.

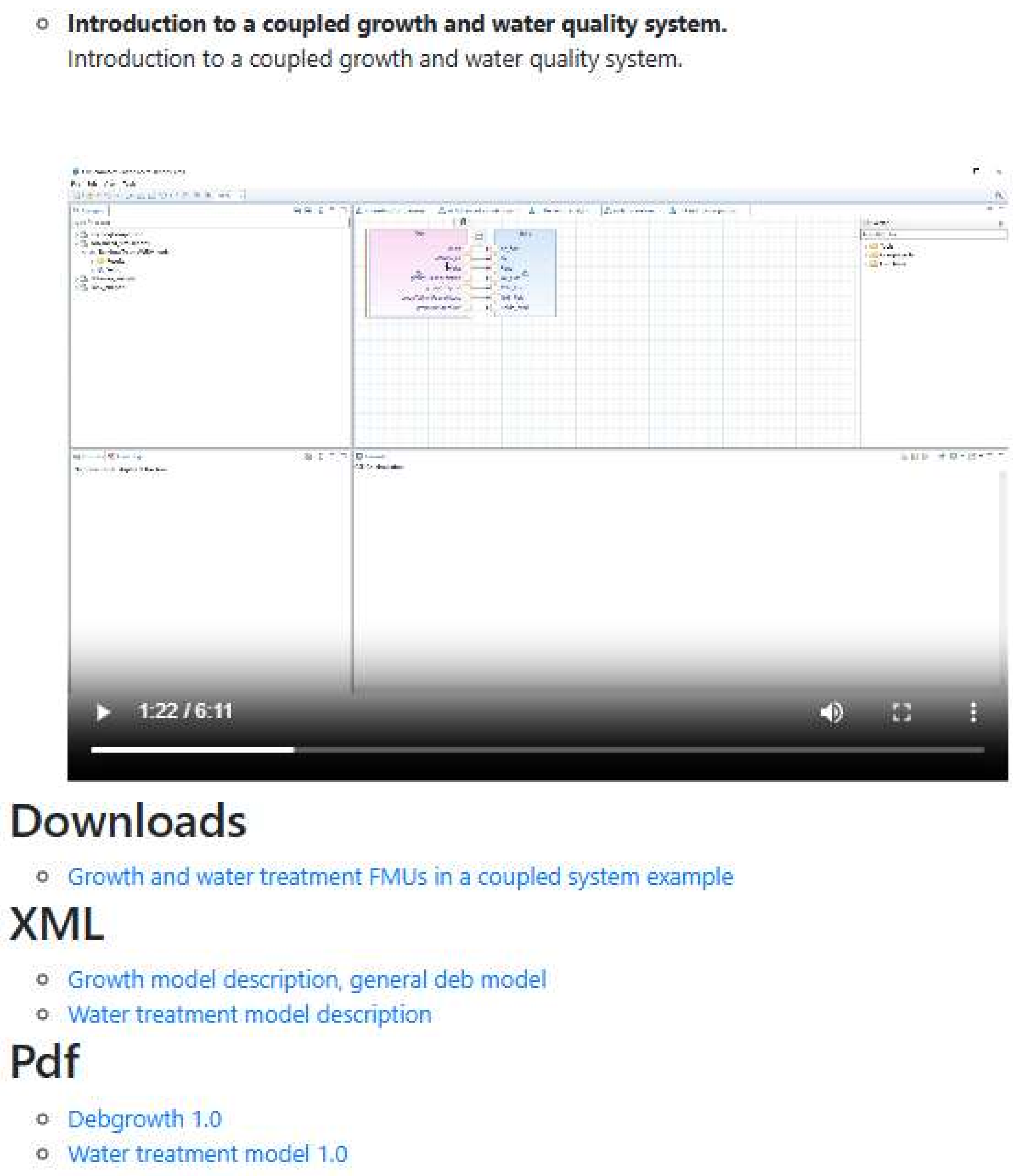
- Start the kopl tool, by double clicking on the Kopl executble.
- Choose a location for the workspace (working folder) or press Ok to use the default workspace.
- Right click in the navigator pane and click new, click CoSimulation task and give it a name ie. Coupled_simulations
- Right click the new task and select new->Configuration
- Give it a descriptive name ie. VURHtank_Rainbowtrout
- For the file, click the browse icon to the far right. Select a tutorial xml that most closely relates to the experiment you want to simulate. The tutorials are located in the configuration folder under the growth folder you downloaded with the Growth module
- For our example we will select the Coupled_VURHtank_rainbowtrout.xml
- Click Finish
- Double click on the configuration (For our example VURHtank_Rainbowtrout)
- Start time is set to 0 and step size to 1.0 which is correct, set duration to 720 (24 hours * 30 days)
- Double click on Fish or Tank under Setup, you will get an overview of the parameters for the different models.
- This example is prefilled for 100 rainbow trouts in the growth model, feeding into a preconfigured VURH tank.
- Modify variables to suit your experiment.
- To simulate, right click the configuration (VURHtank_Rainbowtrout) and select Simulation.
- A results folder is now available under the configuration.

Videos

- **Basic Kopl**
  A basic turorial on downloading and running Kopl for the first time.

- **Basic Loading of Configurations**
  A basic tutorial describing the download and loading of predefined configurations in Kopl.

Figure 22: Web tutorial example 1



- **Introduction to a coupled growth and water quality system.**
  Introduction to a coupled growth and water quality system.

Downloads

- Growth and water treatment FMUs in a coupled system example

XML

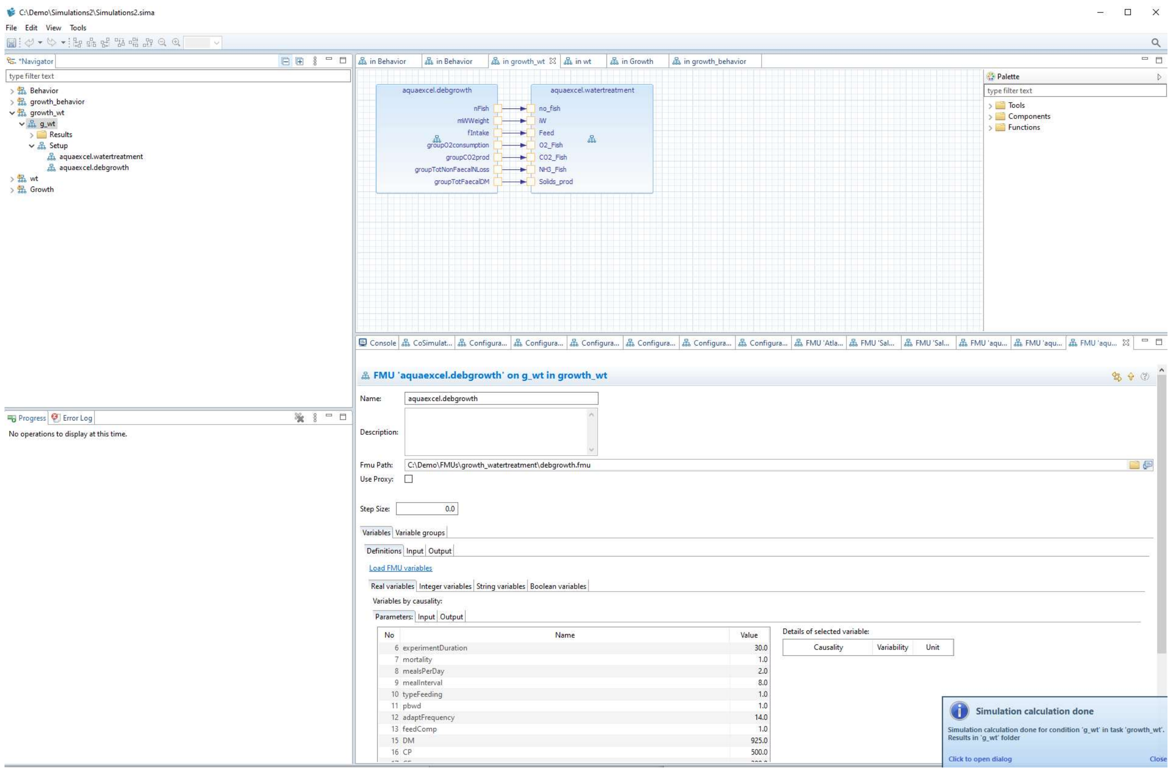
- Growth model description, general deb model
- Water treatment model description

Pdf

- Debgrowth 1.0
- Water treatment model 1.0

Figure 23: Web tutorial example 2

Once everything has been downloaded and loaded into the kopl tool, the user can start to play around with the FMUs, the connections, and the parameters of the model. Figure 24 shows the interface when the example linking growth to water quality is loaded. On the left side, there's a list of different experiments that have been set up. On the top is a graphic interface showing the FMUs. It's possible to copy and paste FMUs here, drag and drop them and connect them to each other using the functions on the top right. The window at the bottom changes to either provide an overview of the selected FMU so the user can tune the parameters or display the main configuration of the setup to tune the length of the experiment. It can also show results once a simulation has been run on the setup, as shown in Figure 25. If a user wants to do more detailed analysis of the simulation, all results are saved in csv files that can be loaded in their analysis software of choice.



FMU 'aquaexcel.debgrowth' on g_wt in growth_wt

Name: aquaexcel.debgrowth

Description:

Fmu Path: C:\Demo\FMUs\growth_watertreatment\debgrowth.fmu

Use Proxy: ☐

Step Size: 0.0

Variables | Variable groups

Definitions | Input | Output

Load FMU variables

Real variables | Integer variables | String variables | Boolean variables

Variables by causality:

Parameters: | Input | Output

No	Name	Value
6	experimentDuration	30.0
7	mortality	1.0
8	mealsPerDay	2.0
9	mealInterval	8.0
10	typeFeeding	1.0
11	pbwd	1.0
12	adaptFrequency	14.0
13	feedComp	1.0
15	DM	925.0
16	CP	500.0

Details of selected variable:

Causality	Variability	Unit

Simulation calculation done

Simulation calculation done for condition 'g_wt' in task 'growth_wt'. Results in 'g_wt' folder

Click to open dialog

Close

Figure 24: Kopl simulation example

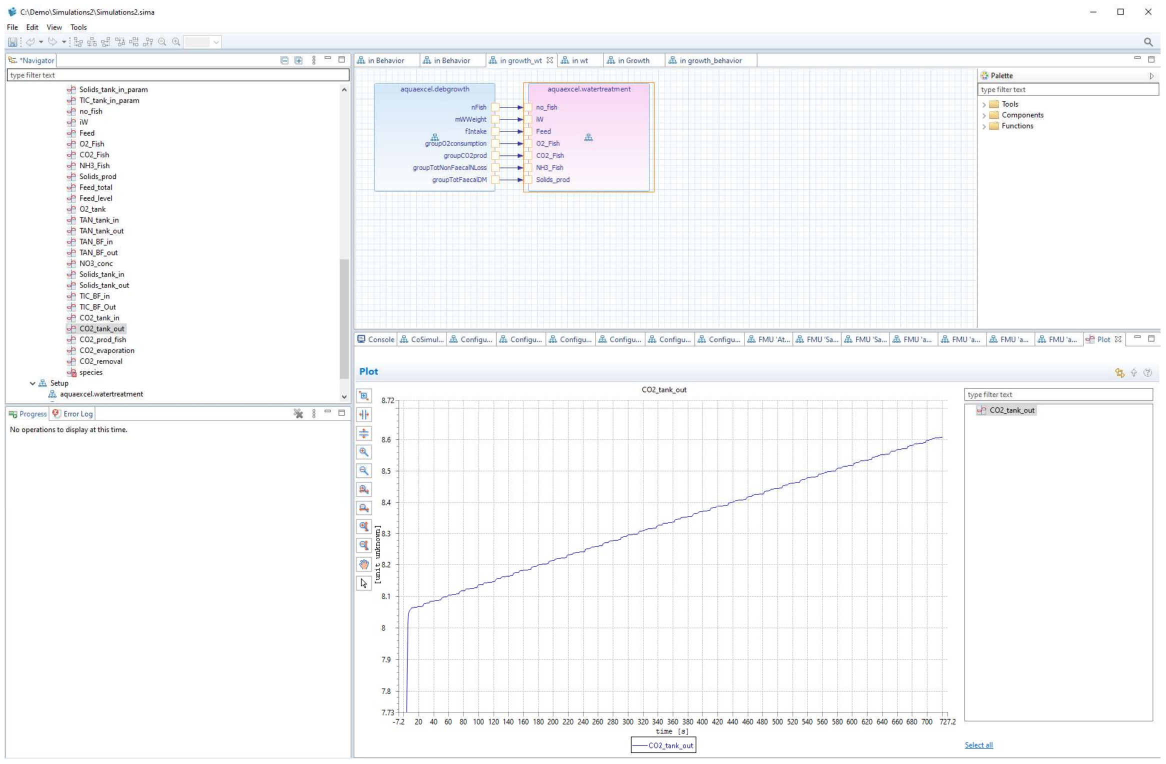


Figure 25: Kopl visualization example

7.2. Artificial agent

The Assistant Module 1.0 exemplifies the core capabilities of an artificial agent: observation, reasoning, and action. Enhanced with OpenAI’s advanced language models and integrated with a Neo4j knowledge graph, it evolves from a simple chatbot into a fully functional agent, as defined in the foundational "Agents" framework Wang (2024). This system goes beyond basic interaction—it actively processes inputs, leverages tools like Large Language Models (LLMs) and knowledge graphs, and delivers intelligent responses, embodying true agentic behavior. Below, we examine how its components (Figure 26, Figure 27) align with these principles.

The Assistant Module operates as an agent by:

- **Observing** user queries through natural language input,
- **Reasoning** over the input using contextual data retrieved from Neo4j,
- **Acting** by generating tailored, knowledge-grounded responses via OpenAI’s LLM.

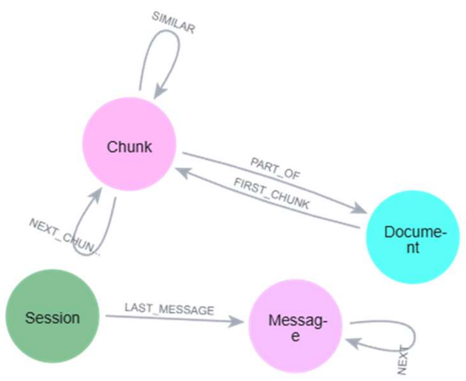


Figure 26 The RAG Neo4j graph database scheme

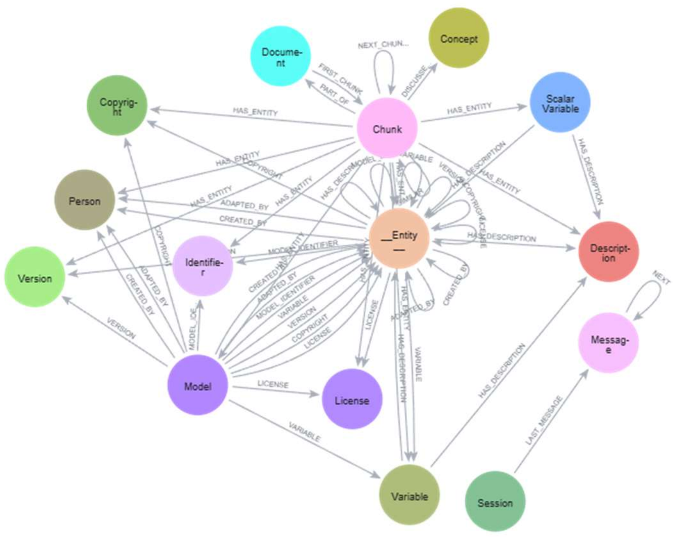


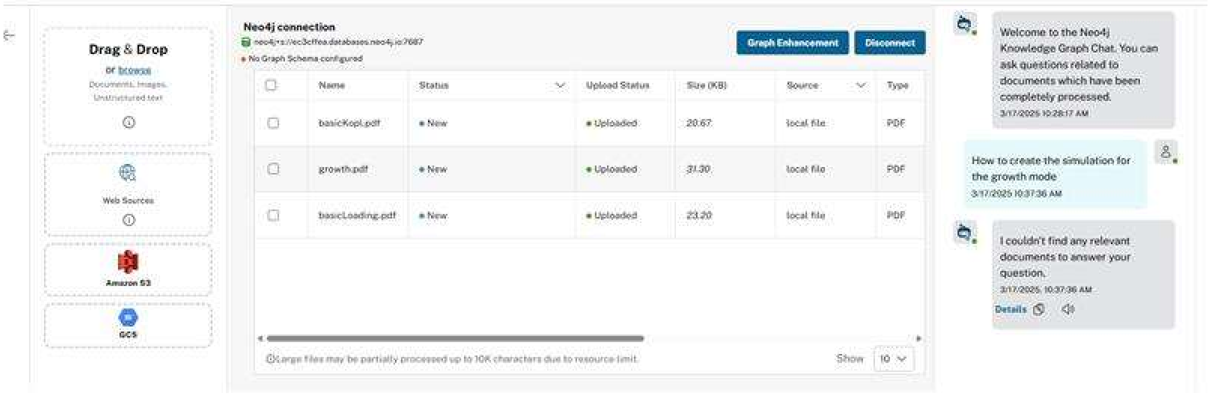
Figure 27 The KAG Neo4j graph database scheme

This loop mirrors the standard architecture of an intelligent agent:

- Figure 26 depicts the RAG (Retrieval-Augmented Generation) generated Neo4j scheme, which forms the agent’s memory system. By chunking documents, generating vector embeddings, and modeling relationships in Neo4j, the agent retrieves precise, domain-relevant information for each query.

- Figure 27 introduces the KAG (Knowledge-Augmented Graph) generated Neo4j scheme, enabling multi-hop reasoning. Explicitly modeled concepts (e.g., fish species, simulation parameters) allow the agent to infer connections beyond direct query terms, enhancing response depth.

- Figure 28 and Figure 29 contrast responses generated with RAG alone (Figure 28) versus RAG + KAG (Figure 29), demonstrating the agent’s evolution from basic retrieval to sophisticated knowledge synthesis.



Neo4j connection
neo4j+s://ec3cffea.databases.neo4j.io:7687
No Graph Schema configured

	Name	Status	Upload Status	Size (KB)	Source	Type
☐	basicKopl.pdf	New	Uploaded	20.67	local file	PDF
☐	growth.pdf	New	Uploaded	31.30	local file	PDF
☐	basicLoading.pdf	New	Uploaded	23.20	local file	PDF

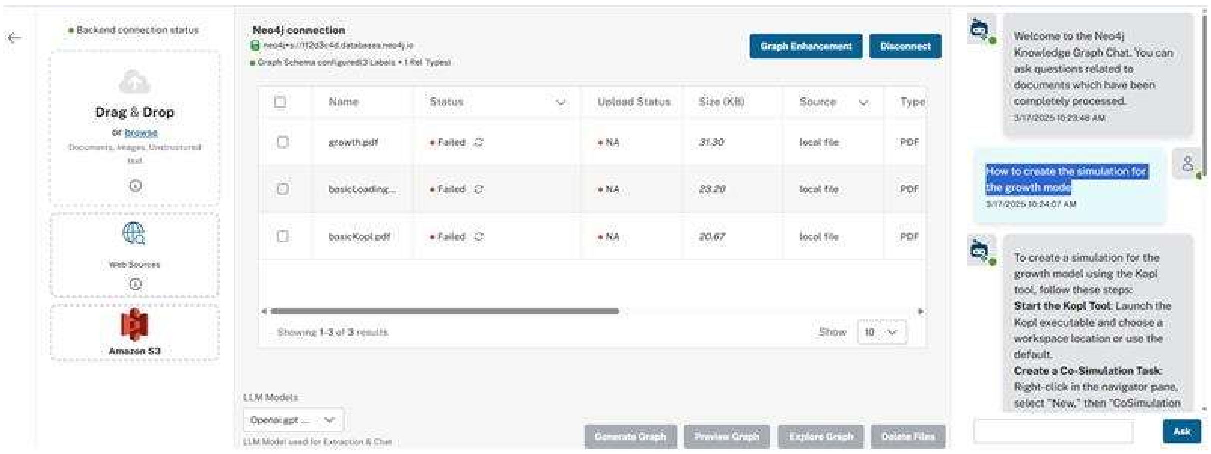
Large files may be partially processed up to 10K characters due to resource limit.
Show 10

Welcome to the Neo4j Knowledge Graph Chat. You can ask questions related to documents which have been completely processed.
3/17/2025 10:28:17 AM

How to create the simulation for the growth mode
3/17/2025 10:37:36 AM

I couldn't find any relevant documents to answer your question.
3/17/2025, 10:37:36 AM

Figure 28 The OpenAI agent response based on the RAG



Neo4j connection
neo4j+s://f12d3c4d.databases.neo4j.io
Graph Schema configured(3 Labels + 1 Rel Types)

	Name	Status	Upload Status	Size (KB)	Source	Type
☐	growth.pdf	Failed	NA	31.30	local file	PDF
☐	basicLoading...	Failed	NA	23.20	local file	PDF
☐	basicKopl.pdf	Failed	NA	20.67	local file	PDF

Showing 1-3 of 3 results
Show 10

LLM Models
Openai gpt ...
LLM Model used for Extraction & Chat

Welcome to the Neo4j Knowledge Graph Chat. You can ask questions related to documents which have been completely processed.
3/17/2025 10:23:48 AM

How to create the simulation for the growth mode
3/17/2025 10:24:07 AM

To create a simulation for the growth model using the Kopl tool, follow these steps:
Start the Kopl Tool: Launch the Kopl executable and choose a workspace location or use the default.
Create a Co-Simulation Task: Right-click in the navigator pane, select "New," then "CoSimulation

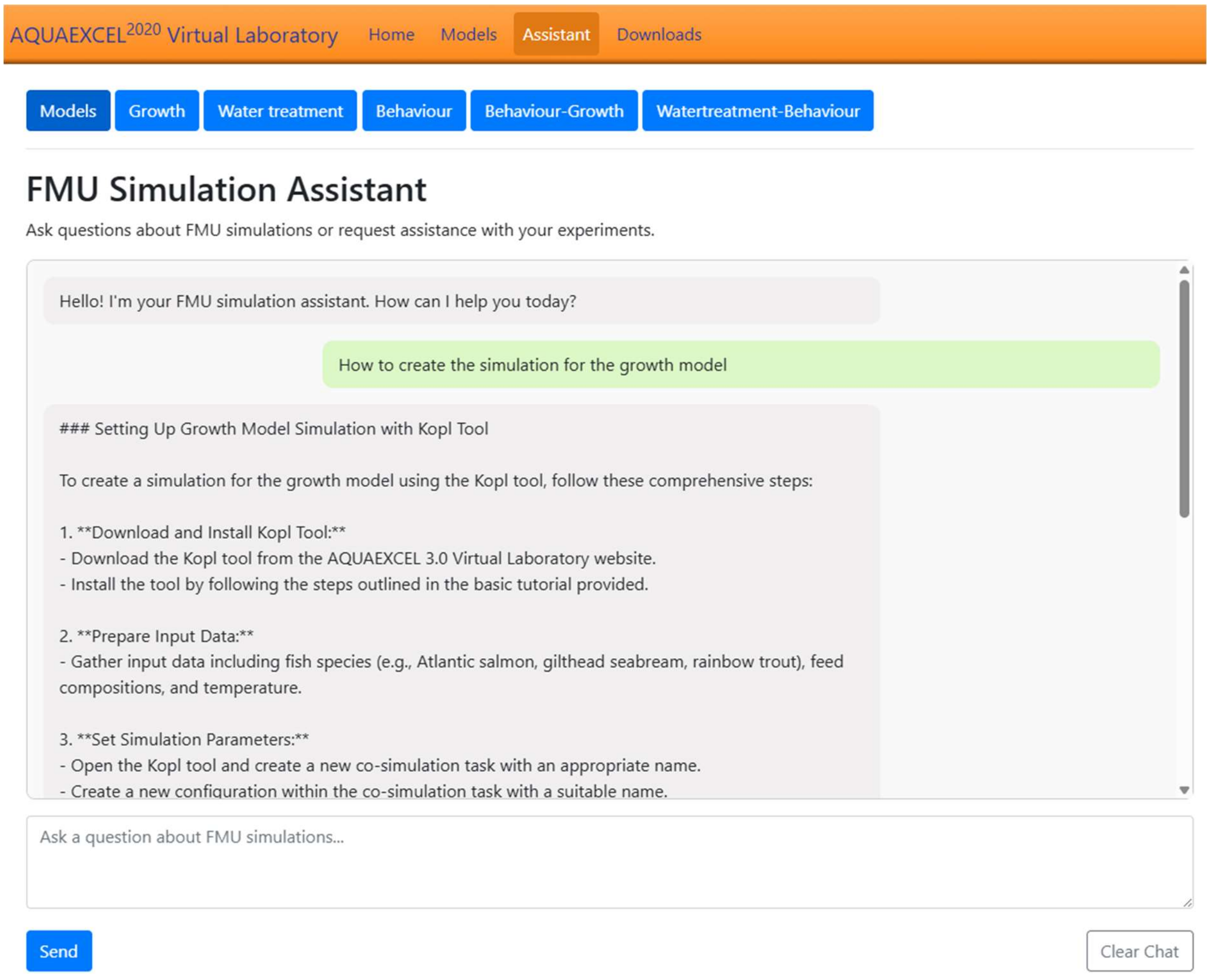
Figure 29 The OpenAI agent response based on the KAG enhancement over the RAG

As highlighted in the foundational "Agents" framework Wang (2024), effective agents rely on tools to extend their capabilities. In the assistant module:

- Neo4j serves as the agent's dynamic memory and retrieval tool,
- OpenAI's LLM acts as the core reasoning engine,
- The LLM integration module orchestrates tool usage, ensuring seamless knowledge retrieval and response generation.

This modular design allows the agent to improve continuously—without retraining—simply by updating documents or refining concept relationships in the graph.

The user-friendly web interface (Figure 30) is where the agent manifests its intelligence. By providing natural, interactive communication, it bridges the gap between the user and the agent's complex reasoning pipelines. This seamless integration showcases how artificial agents can be deployed in domain-specific settings (e.g., aquaculture) to deliver precise, context-aware assistance.



FMU Simulation Assistant

Ask questions about FMU simulations or request assistance with your experiments.

Hello! I'm your FMU simulation assistant. How can I help you today?

How to create the simulation for the growth model

Setting Up Growth Model Simulation with Kopl Tool

To create a simulation for the growth model using the Kopl tool, follow these comprehensive steps:

1. **Download and Install Kopl Tool:**
- Download the Kopl tool from the AQUAEXCEL 3.0 Virtual Laboratory website.
- Install the tool by following the steps outlined in the basic tutorial provided.

2. **Prepare Input Data:**
- Gather input data including fish species (e.g., Atlantic salmon, gilthead seabream, rainbow trout), feed compositions, and temperature.

3. **Set Simulation Parameters:**
- Open the Kopl tool and create a new co-simulation task with an appropriate name.
- Create a new configuration within the co-simulation task with a suitable name.

Figure 30 The assistant user interface

The GenAI-powered Assistant Module is a fully realized artificial agent, combining RAG, KAG, OpenAI's reasoning, and an intuitive interface. It observes, reasons with structured knowledge, and acts through informed responses—embodying the modern agent paradigm in a practical, user-centric application.

8. Conclusion

The virtual laboratory 3.0 has been implemented, integrating new and upgraded models developed in AQUAEXCEL3.0. The laboratory consists of three major parts: The website which offers resources, tutorials and assistance to users. Several different downloadable modules that users can utilize to create their own virtual laboratories and perform virtual experiments. And finally, a chatbot assistant that can provide users with context aware assistance through a chat interface.

The major difference from version 2.0 to 3.0 is to make users responsible for downloading modules and setting up their own simulations. This has led to an increase in flexibility and generalisation of the tool. The drawback is a decrease in usability, we have sought to mediate this through an increased focus on tutorials, both written and videos, providing example configurations that acts as a starting point, and providing a chat interface to a context aware, self-improving AI module that can answer questions regarding setup and simulations.

9. References

- Abbink, W. et al. (2020) *D5.7 Final model on water quality and water temperature for experimental facilities*, AQUAEXCEL2020 Report.
- Alver, M. O. (2020). *D5.8 Final flow field model after testing period*, AQUAEXCEL2020 Report
- Alver, M. O. et al. (2024). *D4.5 Modelled flow fields, turbulence and residence time in experimental units*, AQUAEXCEL3.0 Report
- Bjørnson, F. O. et al. (2020). D5.9 Virtual laboratory version 2, AQUAEXCEL2020 Report
- Blochwitz, T., *Tutorial: Functional Mockup Interface 2.0 and HiL Applications*, presentation from the 10th International Modelica Conference 2014 available at https://www.fmi-standard.org/literature
- Endresen, P.C. et al. (2024) *D4.4 Final model for fish behaviour*, AQUAEXCEL3.0 Report
- Gyalog, G. et al. (2024) *D4.7 Final models for water quality*, AQUAEXCEL3.0 Report
- Lika, K. et al. (2020) *D5.6 Final model for growth, feed consumption and waste production simulation*, AQUAEXCEL2020 Report
- Lika, K. et al. (2024) *D4.6 Final models for growth*, AQUAEXCEL2020 Report
- Saad, A., Su, B. and Bjørnson, F.O., 2023. A web-based platform for efficient and robust simulation of aquaculture systems using integrated intelligent agents. Procedia Computer Science, 225, pp.4560-4569.
- Wang, S., Liu, W., Chen, J., Zhou, Y., Gan, W., Zeng, X., ... & Hao, J. (2024). Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*.

Document Information

EU Project	No 871108	Acronym	AQUAEXCEL3.0
Full Title	AQUAculture infrastructures for EXCELlence in European fish research 3.0		
Project website	www.aquaexcel.eu		

Deliverable	N°	D4.8	Title	Virtual Laboratory version 3.0 with Artificial Agent 1.0
Work Package	N°	4	Title	JRA1 - Technological tools for improved experimental procedures
Work Package Leader	Finn Olav Bjørnson			
Work Participants	SINTEF, NTNU, HCMR, WU, NOFIMA			

Lead Beneficiary	SINTEF, 11
Authors	Finn Olav Bjørnson, SINTEF Ocean, finn.o.bjornson@sintef.no Aya Saad, SINTEF Ocean, Aya.Saad@sintef.no
Reviewers	Martin Føre, NTNU, martin.fore@ntnu.no

Due date of deliverable	30.04.2025
Submission date	30.04.2025
Dissemination level	PU
Type of deliverable	R-Other

Version log			
Issue Date	Revision N°	Author	Change
24.03.2025	0.1	Finn Olav Bjørnson	Document created
28.03.2025	0.4	Finn Olav Bjørnson	First version of chapter 1, 2, 3 and 4
03.04.2025	0.8	Finn Olav Bjørnson, Aya Saad	First version of chapter 5, 6, 7, 8
10.04.2025	0.9	Finn Olav Bjørnson	Revised based on input from reviewer
29.04.2025	1.0	Finn Olav Bjørnson	Minor edits for final version.