



AQUAculture infrastructures for EXCELlence
in European fish research towards 2020 —
AQUAEXCEL2020

D5.5 Virtual laboratory version 1

*Finn Olav Bjørnson, Eleni Kelasidi, Martin Føre,
Gunnar Senneset, Morten Omholt Alver*



Executive Summary

Objectives:

This deliverable describes the first version of the virtual laboratory developed in WP5 of AQUAEXCEL²⁰²⁰. The document describes the technical framework used to integrate the different submodels that are part of the virtual laboratory, short descriptions of these submodels, and demonstrations of the integration through simple simulation experiments. Since the development of the virtual laboratory will continue throughout the remainder of the project period, this report will also serve as a starting point for the remaining work in WP5.

Rationale: One of the main research activities in AQUAEXCEL²⁰²⁰ WP5 is to develop a virtual laboratory system that enables virtual experiments in aquaculture research facilities. This system will feature a framework that allows the integration of mathematical models of different subsystems in common simulations, replicating the system operation of research laboratories.

Main Results:

The first version of the virtual laboratory is now complete, and we have demonstrated that it is possible to run virtual experiments by using the submodels included in WP5 in integrated simulations. A first version of a web-interface for setting up and executing virtual experiments has been completed. The interface is linked with the technical framework integrating the models such that users can execute virtual experiments without needing deep insight into how the framework and submodel integration is implemented. For the first version we have made constraints in the web-interface so the user is restricted to co-simulate models that have been pre-selected, however advanced users with direct access to the underlying framework have no restrictions on how they want to combine the submodels.

Authors/Teams involved:

SINTEF: Finn Olav Bjørnson (lead author), Eleni Kelasidi, Martin Føre, Gunnar Senneset

NTNU: Morten Omholt Alver

HCMR: Nikos Papandroulakis, Orestis Stavrakidis-Zachou, Dina Lika

DLO-IMARES: Wout Abbink, Edward Schram

JU: Stepan Papacek, Petr Cisar

NOFIMA: Åsa Espmark

WU: Ep Eding

Table of contents

Executive Summary	2
Table of contents.....	3
1. BACKGROUND	4
2. FRAMEWORK AND SYSTEM ARCHITECTURE	5
3. INTEGRATED MODELS.....	6
4. VIRTUAL LABORATORY VERSION 1	9
5. DEMONSTRATION	15
6. CONCLUSION	18
Glossary.....	19
Definitions	20
Document information	21
Annex 1: Check list	22

1. BACKGROUND

This document is a deliverable of the AQUAEXCEL²⁰²⁰, WP5/Joint Research Activity 1 – "Virtual laboratories and modelling tools for designing experiments in aquaculture research facilities".

Experiments with fish usually involve extensive use of laboratory facilities, and often need to run for long periods of time to obtain the desired answers. In such experiments, it is essential to ensure that the fish experience acceptable welfare conditions. Together with the potentially high costs of such experiments, the welfare perspective underlines the need for good experimental planning when doing research on fish. Both from an ethical (3R's) and a cost perspective, numerical models may be useful tools for experimental design, preparation and planning before realizing the actual experiments.

One of the main research activities in AQUAEXCEL²⁰²⁰ WP5 is to develop a virtual laboratory system that enables virtual experiments in aquaculture research facilities. This system will feature a framework (see Bjørnson et al., 2016) that allows the integration of mathematical models of different subsystems in common simulations, replicating the system operation of research laboratories. The overall system concept is shown in Figure 1.

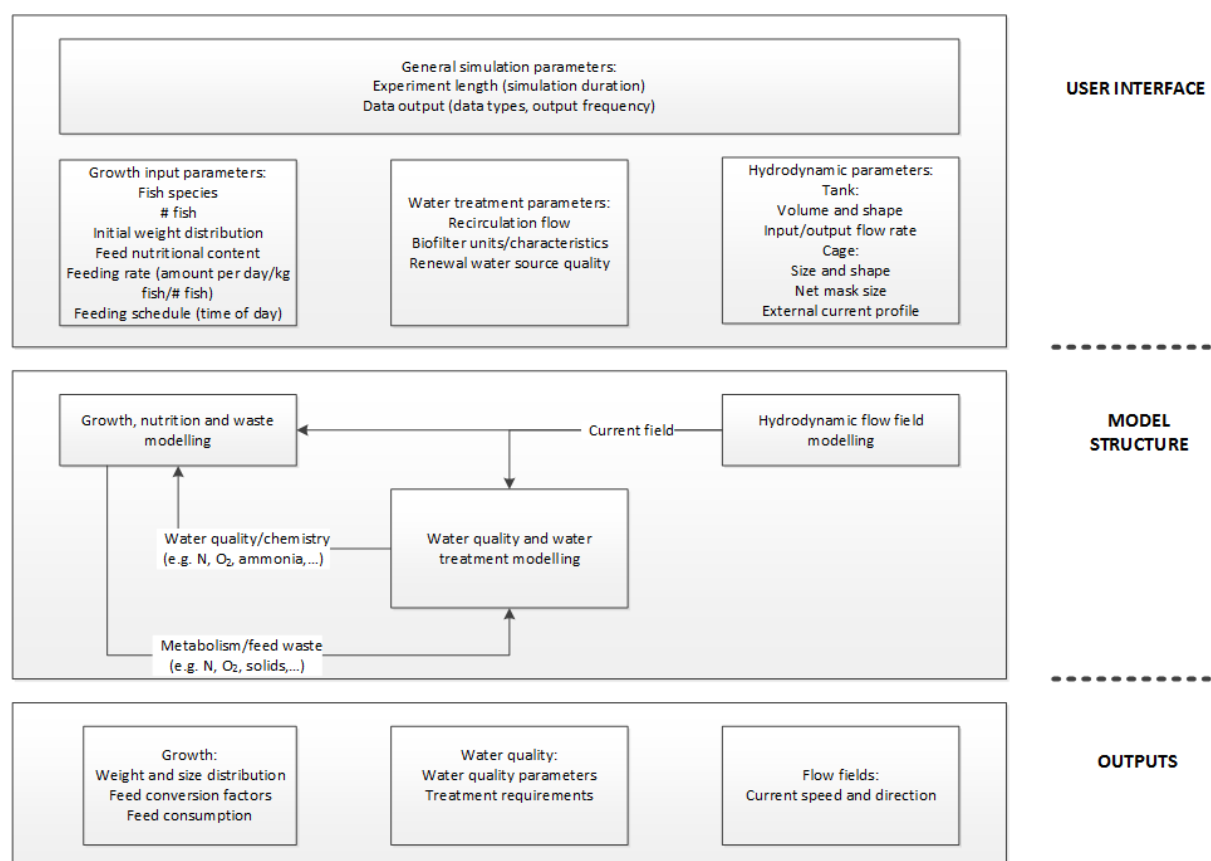


Figure 1 Virtual laboratory system concept

The purpose of this report is to present the first version of the virtual laboratory developed in WP5 of AQUAEXCEL²⁰²⁰. First, the general framework and system architecture used to integrate the different submodels is presented. This is followed by short descriptions of each submodel, covering their function and outputs, and some information on how they were implemented into the common framework. We then present the implementation of the Virtual Laboratory Version 1, first from a technical viewpoint, then through a use case where a user conducts a new experiment. Finally, we sum up the implementation of the virtual lab in the conclusion.

2. FRAMEWORK AND SYSTEM ARCHITECTURE

This section briefly presents the results reported in Bjørnson et al. (2016) describing the system architecture/technical framework that was chosen for developing virtual laboratory solution for the aquaculture domain.

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/downloads> for a list of eligible tools), FMI was chosen as a basic framework for model integration in the AQUAEXCEL²⁰²⁰ virtual laboratories.

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 2 illustrates the co-simulation model.

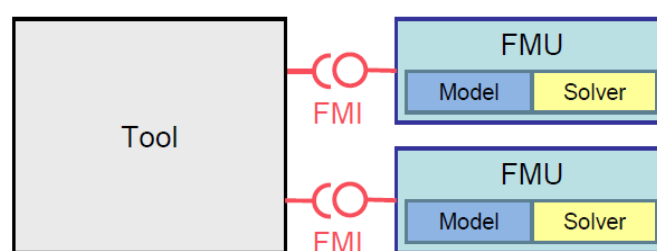


Figure 2 FMI for Co-Simulation (From Blochwitz 2014)

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 synchronising 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.

As shown by Bjørnson et al. (2016), the overall initiation of models can be handled with the architecture described in Figure 3. The Master algorithm receives the experiment setup from the user interface and then asks FMU providers for the necessary FMUs (Stage 1). The FMU providers then handle the initiation and setup of the necessary FMUs and returns them to the Master which then executes the simulation (Stage 2).

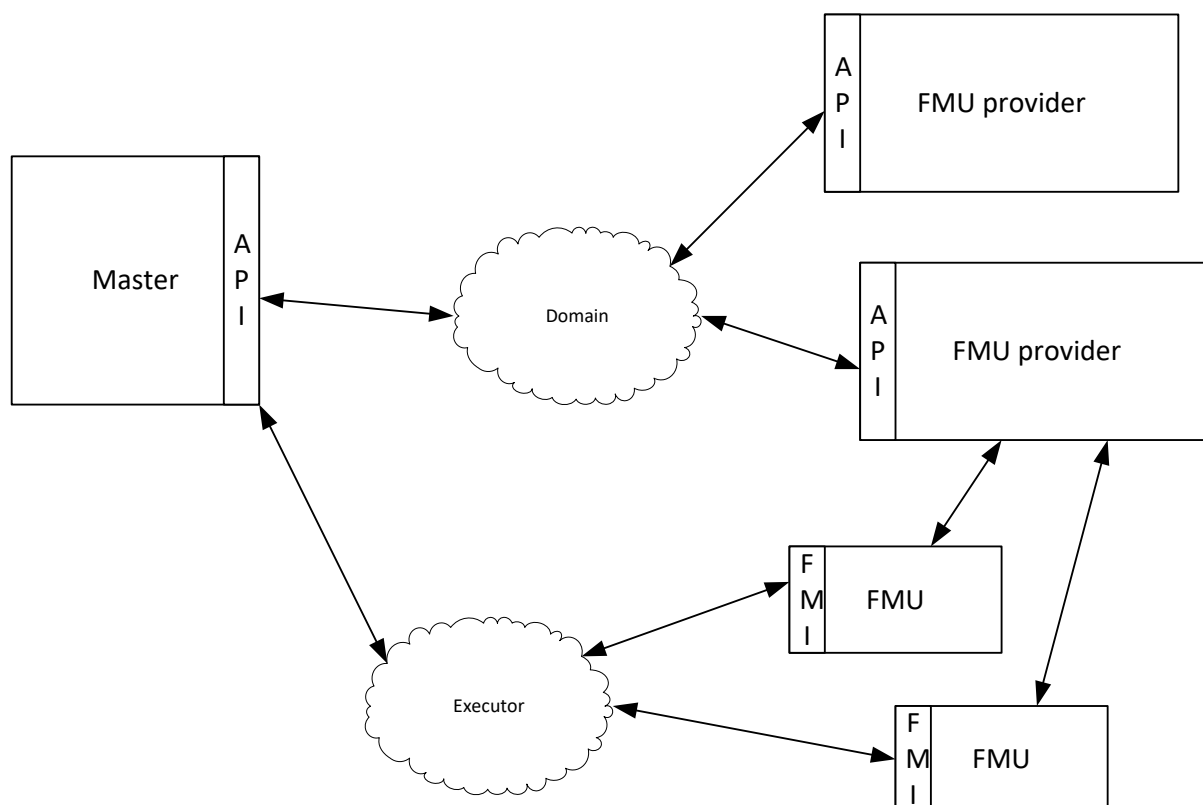


Figure 3 Master initiation architecture

The AQUAEXCEL²⁰²⁰ virtual laboratory is built upon an open-source FMI-master application, Coral, developed by SINTEF Ocean, NTNU and a consortium of industry participants from the maritime segment (see <https://viproma.no/> for more information). At present, three different FMUs are possible to use in the virtual laboratory, simulating fish growth, water treatment and tank/cage hydraulics, respectively. In AQUAEXCEL²⁰²⁰ the aim is to primarily use FMI for co-simulation, and this entails that the FMUs developed for the virtual laboratory need to output their variable values at each communication time step.

Since the essential idea behind FMI is to facilitate integration of independent models in common simulations, future expansion and adaptation of the AQUAEXCEL²⁰²⁰ virtual laboratory to other types of experiments is an easy task. This can either be done by expanding the portfolio of eligible submodels with new FMUs or by modifying the existing FMUs. Furthermore, Bjørnson et al. (2016) observed that when using FMI, the ability to develop models using any programming language or modelling framework (as long as they appear in the list of eligible tools at <https://www.fmi-standard.org/downloads>), is a large advantage with respect to collaborative work between institutions. This also fits well with the format of WP5 in the AQUAEXCEL²⁰²⁰ project, where all submodels are to be developed by different partners. To ensure security, backup and versioning history of the code of all FMUs, SINTEF have provided the project partners with access to their code collaboration server code.sintef.no.

3. INTEGRATED MODELS

The following numerical models are the main components to be included in the virtual laboratory developed in WP5 of AQUAEXCEL²⁰²⁰:

- Growth, nutrition and waste production models for different fish species (task 5.1.)
- Water quality and water treatment modelling (task 5.2.)
- Modelling of hydrodynamic flow fields in tanks and cages (task 5.3.)

These models have been realised as FMUs using the principle of Co-Simulation.

Detailed outlines and discussions on the first prototype models delivered in tasks 5.1, 5.2 and 5.3 can be found in Lika, D. et al. (2018), Abbink, W. et al. (2018) and Alver, M. O. et al. (2018), respectively. In this section, we give a short summary of these deliverables for completeness of this report in summarising the results and outcomes from WP5.

Fish model

The AquaFishDEB model presented in Lika, D. et al. (2018) is designed to predict growth, feed consumption and waste production for Atlantic salmon, seabream and trout. Specifically, the model predicts 1) fish growth for different feeds (quantity and composition) and water temperature, and 2) oxygen consumption and waste production (nitrogen, CO₂, solids) at different fish sizes, temperatures, feed rations and diet compositions for individual fish or groups. The AquaFishDEB model has been developed and tested against validation datasets, and it has been shown that this model component can be integrated with the other main components. 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, D. et al., 2018). 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 (faecal dry matter and nitrogen-loss, expressed in g/h or in g/Kg of feed, as well as non faecal nitrogen loss in g/h) and gaseous exchange (O₂ consumption and CO₂ production).

As presented in Lika, D. et al. (2018), 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 has been accomplished for the three chosen species. The DEB parameters for the rainbow trout model were retrieved from the AmP collection (AmP *Oncorhynchus mykiss* version 2017/10/30 bio.vu.nl/thb/deb/deblab/add_my_pet/), parameters for seabream were estimated simultaneously from zero- and uni-variate data sets provided by HCMR while those for Atlantic salmon were obtained using a combination of data provided by NOFIMA and data from literature.

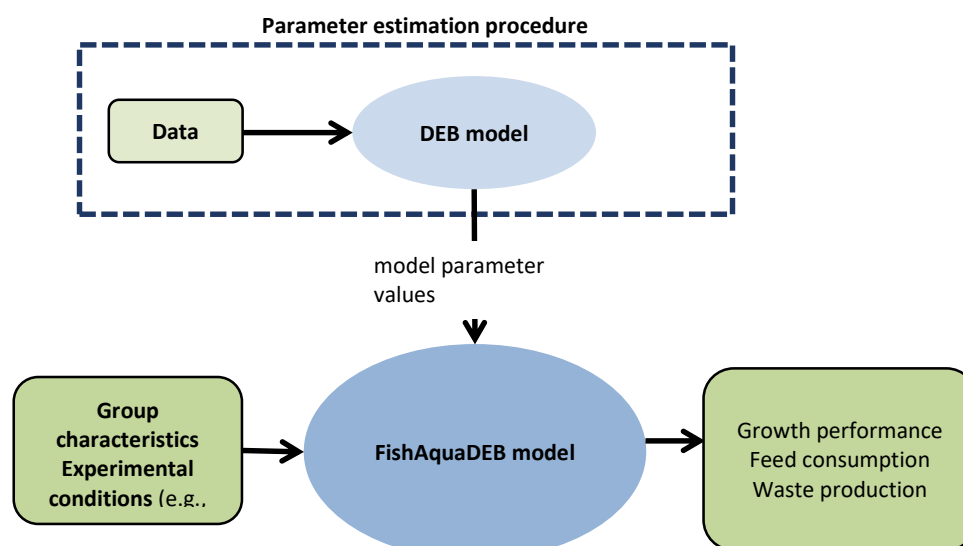


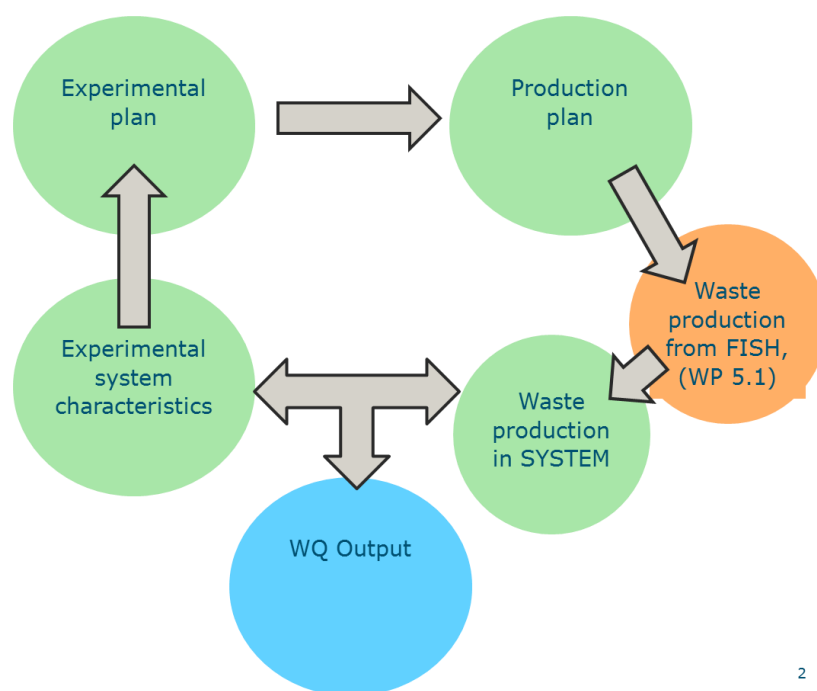
Figure 4 Schematic representation of the two-step procedure for the development of the prototype AquaFishDEB model

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.

Water treatment model

Abbink, W. et al. (2018) presented a model that predicted 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 model uses input on waste production from task 5.1 as a starting point, and since the hydraulic conditions at tank level are important for determining water quality, input from task 5.3 will also be used. This model was thus intended to serve a role in interconnecting the submodels in the virtual laboratory.

Abbink, W. et al. (2018) demonstrated that this model component can be integrated with the other main components developed in tasks 5.1 and 5.3. The sub-model computes water quality based on input parameters that describe the production plan and the experimental design (Figure 5). These inputs may either come from task 5.1 or be provided as direct inputs to the model if they are known in advance. Additional parameters that influence the water quality, such as temperature, are expected to be incorporated in a later version of the model. The model outputs describe water quality using the most crucial parameters related to ammonia and nitrate in the system (tanks and filters). 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, nitrate in the tanks.



2

Figure 5 The main components of the model, with the input parameters; the experimental plan, production plan, waste production of the fish and the system, and the experimental system characteristics, leading to a water quality output as a result

The water quality model was first implemented in Excel, and all tests and tuning of the model was done using this implementation. To convert this model into an FMU, a C++-FMU-interface containing all necessary FMI functions was created. This interface communicated with the Excel implementation by linking in functionality provided by the LibreOffice API and containing a LibreOffice runtime environment. In conclusion, this resulted in an FMU that used the original implementation in Excel to simulate water quality and treatment, meaning that the FMU operated identically to the validated first implementation.

Hydrodynamic model

The objective of the flow field model presented by Alver, M. O. et al. (2018) 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 flow field prototype model has been tested and was found possible to integrate with the other main components from tasks 5.1 and 5.2. The developed flow field model uses one approach for current in tanks – precomputed flow fields from a CFD model – and another for open sea cages – current profiles depending on ambient current conditions. The model interacts with the other model components either through providing the current speed and direction vector for given locations, or through providing descriptive numbers for the overall flow field in the production unit.

The flow field model component is written in C++ and includes a similar FMU-interface as the FMUs for fish growth and water quality. Internal mechanisms for reading outputs from CFD-simulations were programmed directly in the C++ implementation. By reviewing the outputs from the FMU against the outputs from CFD it was possible to ensure that the data were properly read.

4. VIRTUAL LABORATORY VERSION 1

To describe the Virtual Laboratory Version 1, we will look at the software from different perspectives. First, we look at the main components and data packages, and then we consider the view point of a user and look at the user interface flow. Afterwards, we take a high level technical view on communication and the data model, before going into details on the data flow for a simulation.

Main components and data packages

There are three main components of the virtual laboratory: The web interface, built on Django¹ technology, the FMUs from the sub-workpackages built on different technologies but all providing a C interface, and Coral², an open source co-simulation software with support for FMI, built and maintained by SINTEF Ocean.

For details on the FMU implementations we refer to section 3. For more information on Coral we refer to the online documentation². This section will focus on the implementation of the user interface and the integration between the components to realize a complete simulation.

The user interface is implemented in Django 2.1.1, which in turn is running in a Python 3.6 environment. Django is compatible with a multitude of databases, and in our case, we have chosen to run a MySQL database for the project. To separate functionality of the user interface, we have separated functionality into different applications running on a common project environment. Figure 6 provides an overview of the data packages currently running on the system. The Users application handles authentication and personalization of users. The Simulation application is responsible for setting up and running simulations. The Results application handles everything related to showing results of the simulations to the user. And, finally, the Admin application provides super-users with an interface to set user privileges.

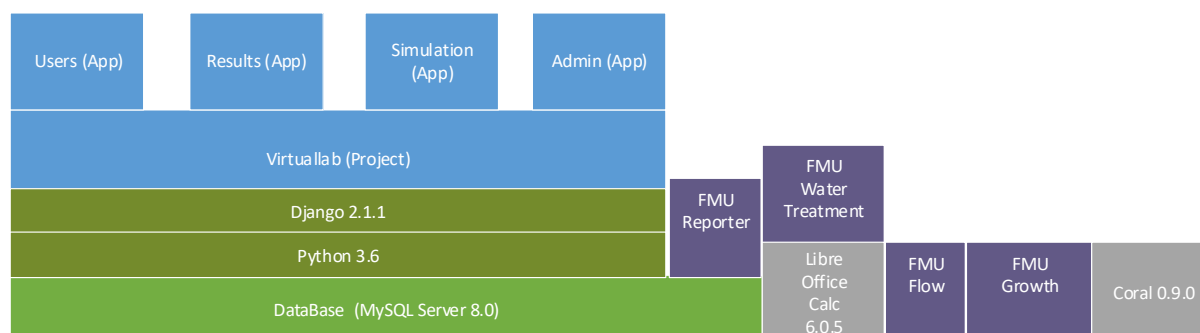


Figure 6 Overview of data packages

User interface flow

Figure 7 provides an overview of the user interface flow. For the user authentication, we are using Django's built in authentication system. Thus, we route any access to the site to the Home page if a user is not authenticated. An anonymous user has few options at the site, limited to logging in, requesting a new password or registering a new user. As an added security measure a new user needs to be authenticated by an administrator before gaining access to the rest of the site.

Once registered and authenticated a user has several more options. They are first routed to the Home page for authenticated users which provides more information about the virtual laboratory. From there the user can choose to log out which returns them to the anonymous Home page. In addition, they can access their personal information and change passwords, while at the same time being able to peruse more information about the different Models currently implemented in the virtual laboratory. The main point of interest will, however, be the Simulate and Results pages. These pages provide the user with their personal configured experiments and results thereof.

The Simulate page provides access to all previous experiments the user has run, as well as access to a simulation wizard to configure new experiments. For the Virtual Laboratory Version

¹ <https://www.djangoproject.com/>

² <https://github.com/viproma/coral/>

1, we have made the constraint on the system that the user is only allowed to simulate a growth model together with a water quality model. The user specifies the infrastructure setup together with the biomass and feeding regime, before selecting the length of the experiment. All parameters are validated in web forms before the simulation starts and the user is returned to the start of the Simulation page.

Once a simulation has finished the experiment results will be available at the Results page. The user can choose which experiment and which parameter to review, as well as downloading the complete dataset in csv format for further analysis offline.

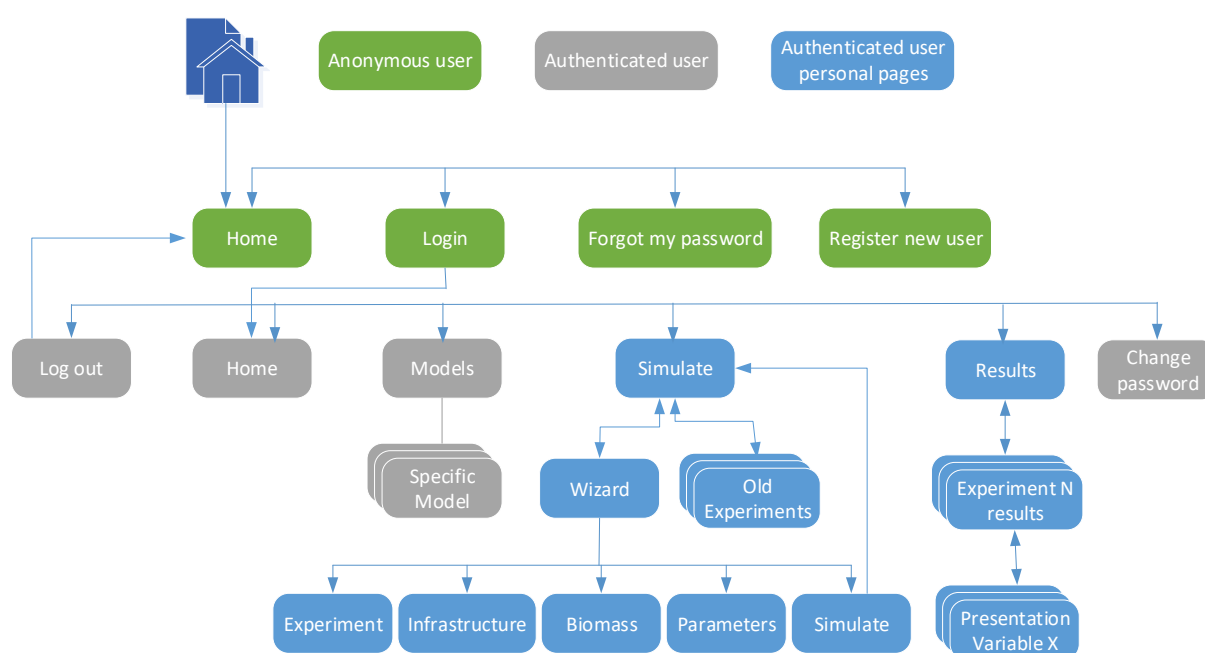


Figure 7 Web structure

High level communication

Figure 8 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 Architecture³. 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.

³ <https://en.wikipedia.org/wiki/Model-view-controller>

To improve the user experience of the Virtual Laboratory, we make use of Bootstrap⁴, an open source toolkit for developing with HTML, CSS and JS, as well as Highcharts⁵ an SVG-based, multi-platform charting library for presenting the data in the template layer. In order to process the data at the View layer, we make use of Pandas⁶, an open source library providing data structures and data analysis tools for Python.

To enable co-simulation, we use the Coral package. It is initiated from the View layer and updates the virtual laboratory through a Reporter FMU which writes results back to the database. We will examine this process in more detail in the Detailed data flow section.

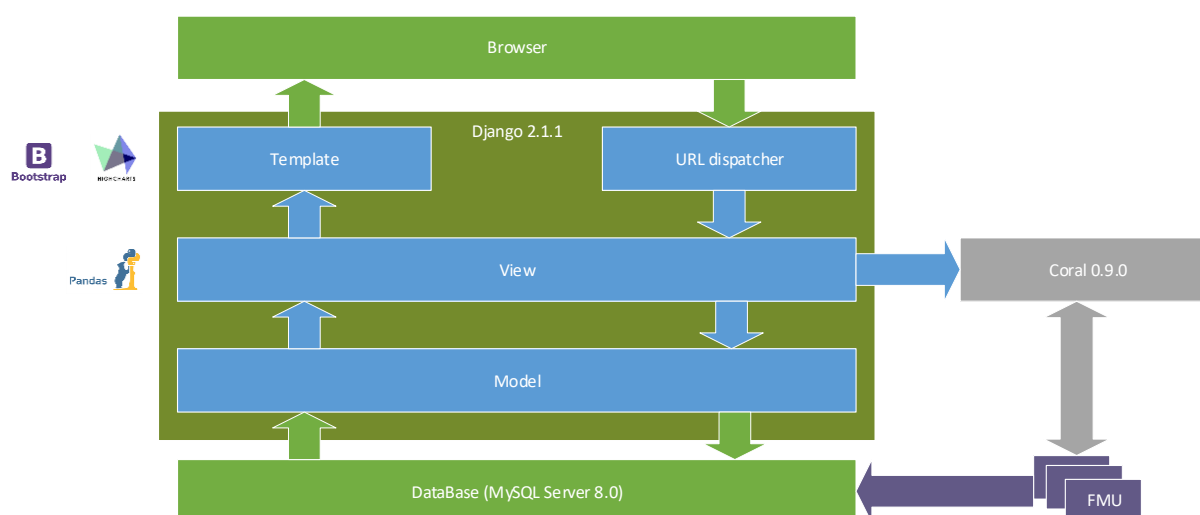


Figure 8 High level communication

Data structures

Data structures for the virtual laboratory is handled by the Model layer and mirrored in the MySQL database (see Figure 8). A data structure in Django is called a Model, not to be confused by the simulation models implemented in FMUs. A Model is mirrored in a table in the database. The naming convention of the data table is "AppName_ModelName". All models have access to their related models even if they are defined in different applications. An overall view of the data structures and their relationship in the database is described in Figure 9 in crowfoot notation.

At the core of the data structure is the experiment, defined in `Simulation_experiment`, this data model contains information on whether the experiment has been validated, if it has started, and what the progress of the simulation is. A user defined in `User_customuser` can link to multiple experiments. For the Version 1 of the virtual laboratory an experiment has a water treatment model defined in `Simulation_watertreatment`, a fish growth model defined in `Simulation_fishgrowth` and a list of core parameters defined in `Simulation_experimentcore`.

`Simulation_watertreatment` and `Simulation_fishgrowth` contains the input parameters for the respective models that the user may set for an experiment along with information on validation ranges. `Simulation_experimentcore` contains information on the start and stop time for the simulation as well as the integration step size. Once run the results of the simulation is stored in the `Results_simulationsresults` table.

⁴ <https://getbootstrap.com/>

⁵ <https://www.highcharts.com/>

⁶ <https://pandas.pydata.org/>

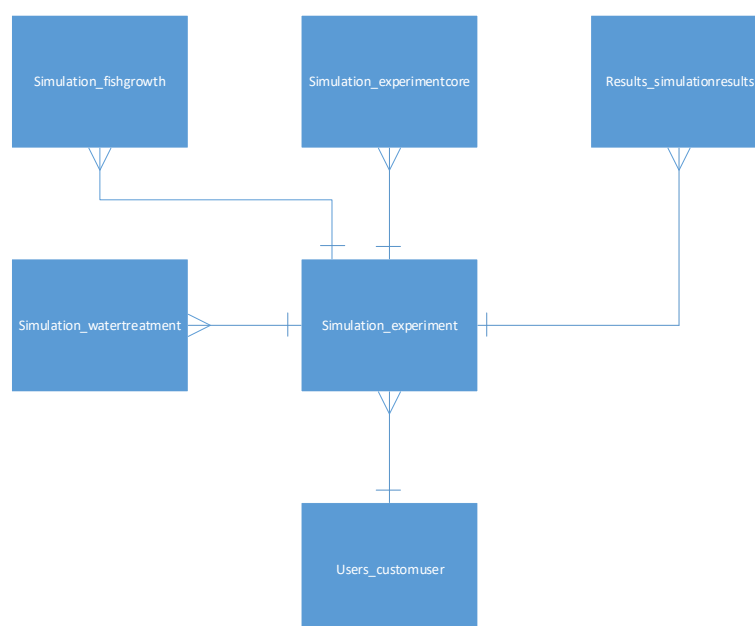


Figure 9 High level data structure in crowfoot notation

Detailed data flow

The main challenge of the Virtual Laboratory Version 1 has been to achieve a seamless integration with the underlying co-simulation software Coral and the FMUs provided by the sub-workpackages. In this section, we delve deeper into our solution for integration of the underlying simulation software. Figure 10 provides an overview of the detailed data flow which can be divided into three different stages we go through these stages in detail below.

The first stage of integration is using Corals integrated SlaveProvider and registering the FMUs with the provider as shown in Figure 10 Stage 1. This makes the FMUs available on the localhost for simulation.

The second stage takes place within the webframework of the virtual laboratory (Figure 10 Stage 2). A user registers a new experiment and validates the input data, this happens through Django's integrated form validation procedures. Once all input has been validated and set in the Model layer, the user gains access to the Simulate button which activates the simulation.

In the third stage, a call is sent to the underlying experiment model which generates an execonf and sysconf file for the experiment based on the parameters set in the Model layer. For details on what goes into these files see Bjørnson et al. (2016). These files are then sent to the CoralMaster software for simulation as illustrated in Figure 10 Stage 3. The CoralMaster checks the localhost for available FMUs. The CoralSlaveProvider responds with available FMUs and if these matches the FMUs required by the CoralMaster, new slave instances of the FMUs are activated for the CoralMaster. The CoralMaster then administers the co-simulation, providing a databus for data exchange between the CoralSlaves at each integration step. The CoralSlaves handles simulation between integration steps and queries and delivers data at the integration steps. In order to feed the data back into the virtual laboratory, we make use of a Reporter FMU which is tasked with reading parameter data at each integration step and feeding that data directly into the database.

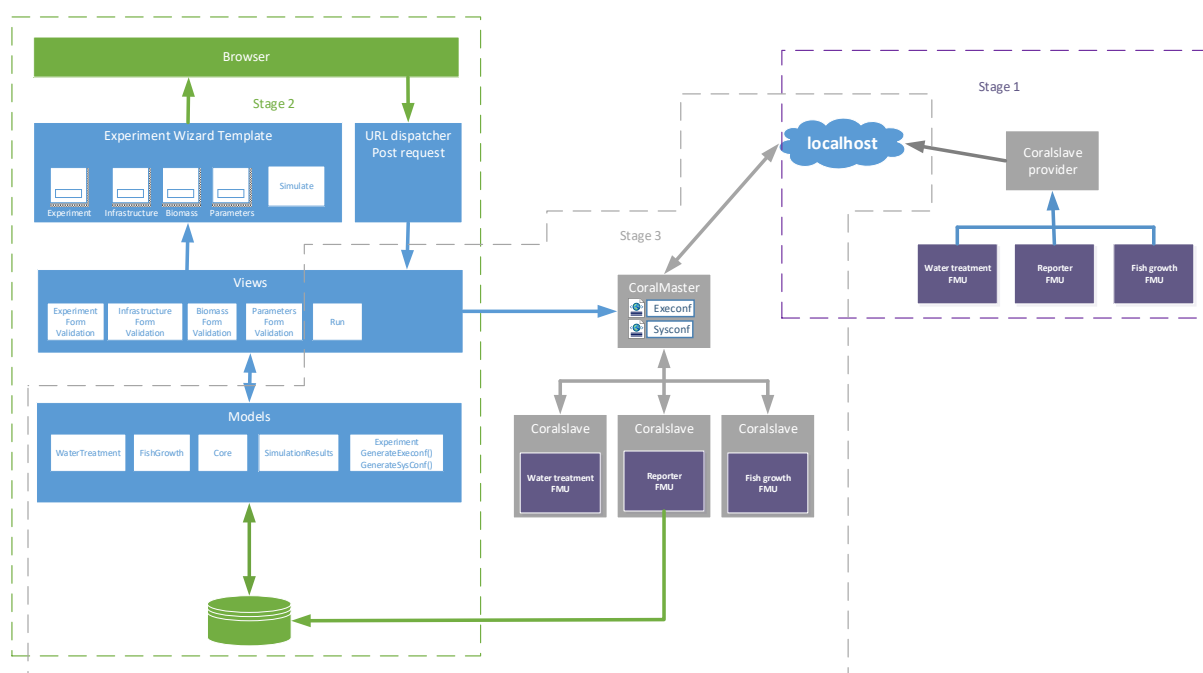


Figure 10 Detailed data flow for a simulation

Direct framework access

The webframework of the Virtual Laboratory Version 1 provides an easy to use access for users who want to simulate experiments. However, the tradeoff we make for increased usability is a decrease in flexibility.

The underlying framework does not have any limitation on experiment setup however, so it is possible for advanced users to configure more elaborate experiments. This is done by bypassing the webframework as illustrated in Figure 11. The loading of FMUs by the CoralSlaveProvider is the same as described above, but Stage 2 is bypassed by specifying an experiment execonf and sysconf file. These configuration files are then sent directly to the CoralMaster in Stage 3. Results of the experiment is then available as csv files, and the user needs to post-process the files themselves.

In order to take this approach, a user needs insight into each of the underlying model FMUs and how they might be connected. In addition, they need to setup their own Coral environment and know how to specify their own configuration files. This might be beyond most common users of the system, but it allows project participants to experiment with advanced simulations while awaiting a more flexible version two of the Virtual Laboratory.

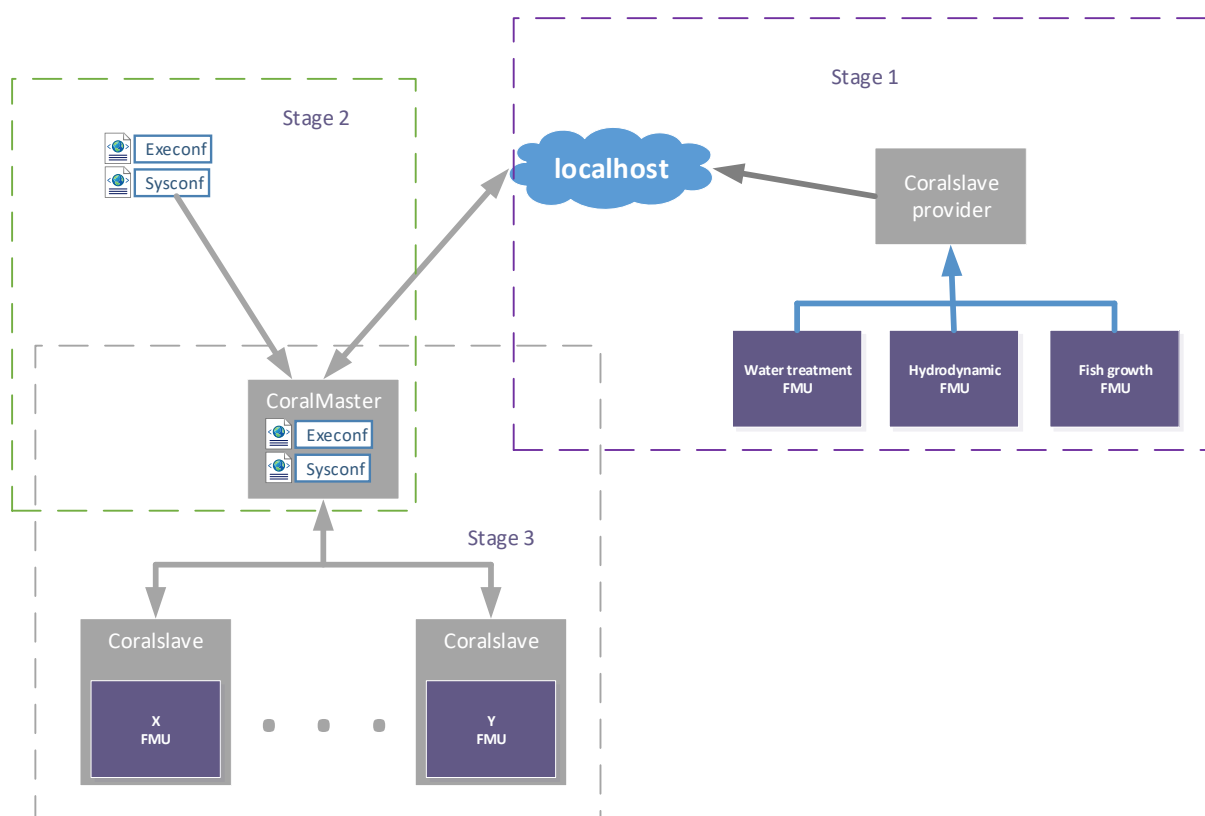


Figure 11 Direct Framework Access

5. DEMONSTRATION

In this section, we demonstrate the user interface of the Virtual Laboratory, through the configuration of a simple experiment. Figure 12 shows the main page of the simulation area. The user may choose to review old experiments or launch a new one using the Experiment Wizard button. For our demonstration, we will continue on to the Wizard.

AQUAEXCEL2020 Virtual Laboratory
User: admin Log out

Home
Models
Simulate
Results

Configure Experiment

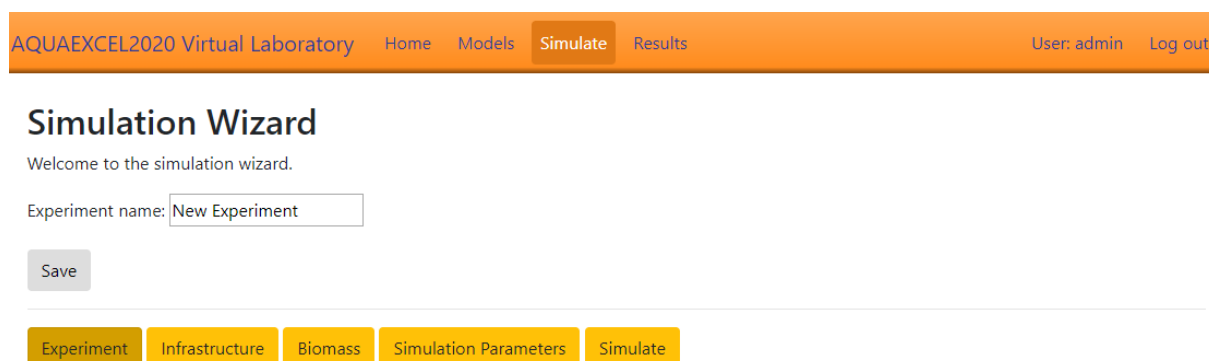
Experiment Wizard

Load previous experiment

Status	Experiment name
✖	New Feeding regime
✖	Atlantic Salmon 2
✔	Experiment 23
✔	Experiment 22

Figure 12 Simulation main page

Figure 13 shows the page that loads when the user starts the simulation wizard. The wizard is split into five subpages: *Experiment*, *Infrastructure*, *Biomass*, *Parameters* and *Simulate*. The user is first sent to the Experiment page where they may choose a name for the new experiment. This is the name that will be linked in the main simulation page under Previous Experiments, see Figure 12.



AQUAEXCEL²⁰²⁰ Virtual Laboratory Home Models **Simulate** Results User: admin Log out

Simulation Wizard

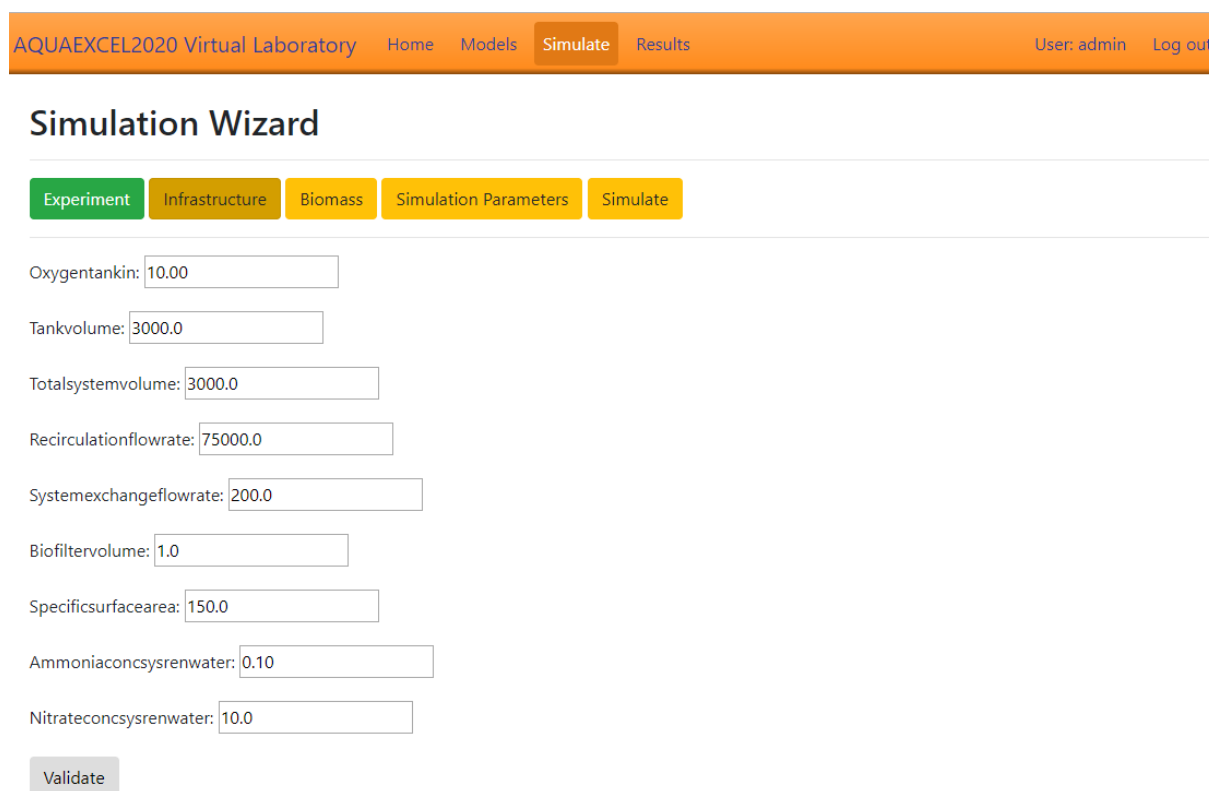
Welcome to the simulation wizard.

Experiment name:

Experiment Infrastructure Biomass Simulation Parameters Simulate

Figure 13 Simulation Wizard

All subpages of the wizard have a validation state. Once the user has confirmed the input, the Virtual Laboratory will validate if the parameters are within acceptable ranges and if they are turn the button from yellow (unvalidated) to green (validated). Figure 14 demonstrates this where the user has validated a new experiment name and is currently configuring the water treatment model.



AQUAEXCEL²⁰²⁰ Virtual Laboratory Home Models **Simulate** Results User: admin Log out

Simulation Wizard

Experiment Infrastructure Biomass Simulation Parameters Simulate

Oxygentankin:

Tankvolume:

Totalsystemvolume:

Recirculationflowrate:

Systemexchangeflowrate:

Biofiltervolume:

Specificsurfacearea:

Ammoniaconcsysrenwater:

Nitrateconcsysrenwater:

Figure 14 Simulation Wizard Infrastructure Configuration

It is possible to go through the experiment wizard in any order by pushing the five different buttons. Figure 15 demonstrates this when we have validated the parameters for the water treatment model and skipped ahead to the Simulate button. Here we get a status of what has been configured. The Virtual Laboratory will not let us simulate until all parameters are set and validated, at which point the Simulate button will activate and the user may start the simulation.

Simulation Wizard

[Experiment](#) [Infrastructure](#) [Biomass](#) [Simulation Parameters](#) [Simulate](#)

- Growth model coupled to watertreatment model ok
- Name validated ok
- Watertreatment model validated ok
- Please configure the fish growth model.
- Please configure simulation parameters.

[Simulate](#)

Figure 15 Simulation Wizard Simulation Start

Once the simulation finishes the experiment results will be available from the Results page. Figure 16 shows the listing of different experiments that have been successfully simulated.

Results

Experiment name

[Experiment 23](#)

[Experiment 22](#)

Figure 16 Results Main Page

The user may browse individual data series from the subpages of the individual experiments. Figure 17 illustrates the growth rate of individual fish for the first 60 days of our experiment.

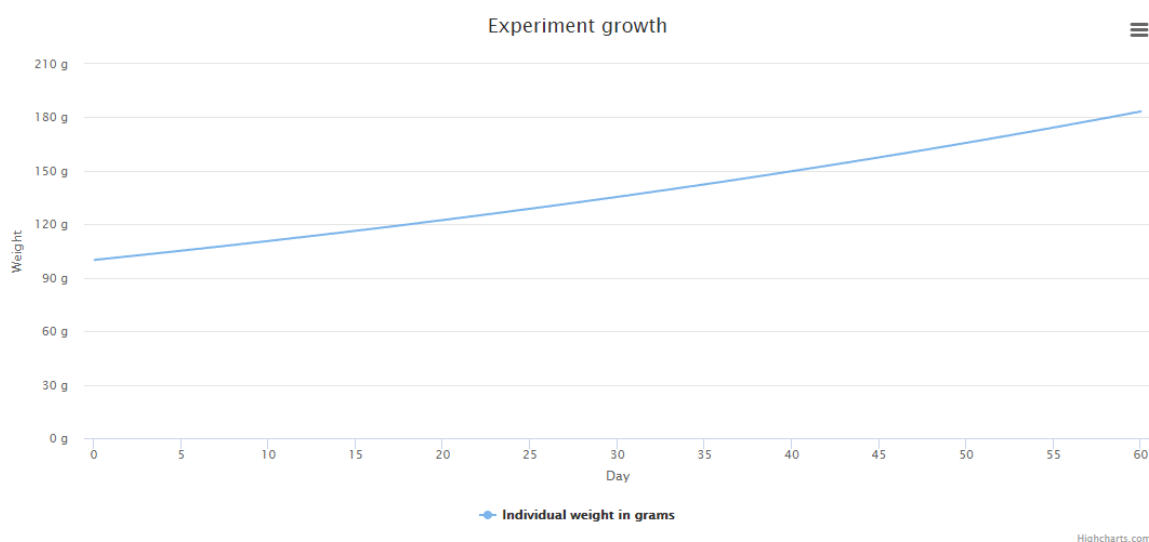


Figure 17 Results Page

6. CONCLUSION

The first version of the Virtual Laboratory has been implemented and we have successfully been able to demonstrate co-simulation of models developed in the sub-workpackages using the architectural principles laid out in Bjørnson et al. (2016). The major constraint of the software in version one, is that regular users are restricted to co-simulate models that have been pre-selected. However, more advanced simulations are possible for advanced users who know how to operate the underlying framework. Future expansions will focus on making the software more dynamic and generic.

References

- Alver, M. O. et al. (2018). *D5.4 First prototype flow field model established*, AQUAEXCEL2020 Report.
- Abbink, W. et al. (2018) *D5.3 First prototype model on water quality and water temperature for experimental facilities*, AQUAEXCEL2020 Report.
- Lika, D. et al. (2018) *D5.2 First prototype model on AquaFishDEB*, AQUAEXCEL2020 Report
- Bjørnson, F. O. et al. (2016). *D5.1 Model development guidelines*, 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>

Glossary

AQUAEXCEL²⁰²⁰: AQUAculture Infrastructures for EXCELlence in European Fish Research towards 2020

FMI: Functional Mock-up Interface

FMU: Functional Mock-up Unit

CFD: Computational Fluid Dynamics

HTML: Hyper Text Markup Language

CSS: Cascading Style Sheet

JS: Java Script

SVG: Scalable Vector Graphics

Definitions

Domain: Particular area of activity or interest

Software framework: Reusable software environment that facilitates development of software applications, products and solutions

System architecture: Conceptual model that defines the structure and behaviour of a system

Virtual laboratory: An interactive environment for creating and conducting simulated experiments

3R: Guiding principles for more ethical use of animals in testing (Replacement, Reduction, Refinement)

Document information

EU Project N°	652831	Acronym	AQUAEXCEL ²⁰²⁰
Full Title	AQUAculture Infrastructures for EXCELlence in European Fish Research towards 2020		
Project website	www.aquaexcel.eu		

Deliverable	N°	D5.5	Title	Virtual laboratory version 1
Work Package	N°	5	Title	Virtual laboratories and modelling tools for designing experiments in aquaculture facilities

Date of delivery	Contractual	30/09/2018 (Month 36)	Actual	24/09/2018 (Month 36)
Dissemination level	X	PU Public, fully open, e.g. web		
		CO Confidential, restricted under conditions set out in Model Grant Agreement		
		CI Classified, information as referred to in Commission Decision 2001/844/EC.		

Authors (Partner)				
Responsible Author	Name	Finn Olav Bjørnson	Email	Finn.O.Bjornson@sintef.no

Version log			
Issue Date	Revision N°	Author	Change
12/09/2018	0	Finn Olav Bjørnson	First version
	1	Eleni Kelasidi	First review by WP leader
21/09/2018			Revision after review by signed reviewer and coordinator

Annex 1: Check list

Deliverable Check list (to be checked by the “Deliverable leader”)

	Check list		Comments
BEFORE	I have checked the due date and have planned completion in due time	X	<i>Please inform Management Team of any foreseen delays</i>
	The title corresponds to the title in the DOW	X	<i>If not please inform the Management Team with justification</i>
	The dissemination level corresponds to that indicated in the DOW	X	
	The contributors (authors) correspond to those indicated in the DOW	X	
	The Table of Contents has been validated with the Activity Leader	X	<i>Please validate the Table of Content with your Activity Leader before drafting the deliverable</i>
	I am using the AQUAEXCEL ²⁰²⁰ deliverable template (title page, styles etc)	X	<i>Available in “Useful Documents” on the collaborative workspace</i>
The draft is ready			
AFTER	I have written a good summary at the beginning of the Deliverable	X	<i>A 1-2 pages maximum summary is mandatory (not formal but really informative on the content of the Deliverable)</i>
	The deliverable has been reviewed by all contributors (authors)	X	<i>Make sure all contributors have reviewed and approved the final version of the deliverable. You should leave sufficient time for this validation.</i>
	I have done a spell check and had the English verified	X	
	I have sent the final version to the WP Leader, to the 2 nd Reviewer and to the Project coordinator (cc to the project manager) for approval	X	<i>Send the final draft to your WPLLeader, the 2nd Reviewer and the coordinator with cc to the project manager on the 1st day of the due month and leave 2 weeks for feedback. Inform the reviewers of the changes (if any) you have made to address their comments. Once validated by the 2 reviewers and the coordinator, send the final version to the Project Manager who will then submit it to the EC.</i>