



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

## **D5.1 Model development guidelines**

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



## Executive Summary

### Objectives:

The purpose of this report is to describe guidelines for model developers involved in developing virtual laboratory solutions for the aquaculture domain. This allows for developing new models as well as integration of existing models. The framework developed and implemented will be flexible in terms of including new mathematical models, facilitating easy future expansion and adaptation to other types of experiments.

### Rationale:

One of the main research activities in AQUAEXCEL2020 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. To ensure that ongoing work is taken into account, a survey of existing models and development tools has been done among the WP5 partners.

### Main Results:

Based on work in other sectors on integrating different simulation models, and comparisons of key features, the framework FMI for Model Exchange and Co-Simulation is chosen for use in the AQUAEXCEL2020 project. The standard allows for integration of models developed using different tools, and is available at <https://www.fmi-standard.org/downloads>. The deliverable includes examples and guidelines for model development. To ensure security, backup and versioning history of the code, SINTEF will provide partners access to their code collaboration server [code.sintef.no](http://code.sintef.no).

### Authors/Teams involved:

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

NTNU: Morten Omholt Alver

HCMR: Nikos Papandroulakis, Dina Lika

DLO-IMARES: Wout Abbink

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. CHOOSING A FRAMEWORK .....	4
3. FUNCTIONAL MOCK-UP INTERFACE.....	6
4. SYSTEM ARCHITECTURE .....	9
5. MODEL DEVELOPMENT GUIDELINES .....	12
6. CODE REPOSITORY .....	16
7. CONCLUSION .....	16
Glossary.....	18
Definitions .....	19
Document information .....	20
Annex 1: Check list .....	21
Annex 2: Survey .....	22

## 1. **BACKGROUND**

This report is part 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 run for long periods of time. Both from an ethical perspective (3R's) and from a cost perspective, tools for design and planning of experiments are increasingly important. In aquaculture research as well as other domains, numerical models are increasingly used preparatory to the actual experiments.

One of the main research activities in AQUAEXCEL2020 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.

The purpose of this report is to describe a system architecture/technical framework to be used for developing virtual laboratory solutions for the aquaculture domain. This will provide guidelines for developing new models and for the integration of existing models. The framework developed and implemented will be flexible in terms of including new mathematical models, facilitating easy future expansion and adaptation to other types of experiments.

To ensure that ongoing work is taken into account, a survey of existing models and development tools has been done among the WP5 partners (see Annex 2).

## 2. **CHOOSING A FRAMEWORK**

Some work has been done in other sectors on integrating different simulation models for different purposes, we have identified three primary domains that have been leading within this field.

The domain which has the longest record is military command-and-control, and several frameworks have been developed and used for integrating different simulations into a single scenario for wargame purposes. Starting in the middle of the 80's and running until today, several frameworks has come and gone: SIMulator Network (SIMNET)<sup>1</sup>, Distributed Interactive Simulation (DIS)<sup>2</sup>, and Aggregated Level Simulation Protocol (ALSP)<sup>3</sup>. The current successor to these frameworks is the High Level Architecture (HLA)<sup>4</sup> which is one of the frameworks we have considered for use in this project.

The second domain that has been leading within this field is the computer game industry, Jain and McLean (2005) describe an architecture for integrating simulation and gaming architecture in order to facilitate incident training. However, frameworks within this category are either described at an abstract level and not implemented, or proprietary. We have not been able to identify open source frameworks within this domain that might be applicable for this project.

---

<sup>1</sup> <https://en.wikipedia.org/wiki/SIMNET>

<sup>2</sup> [https://en.wikipedia.org/wiki/Distributed\\_Interactive\\_Simulation](https://en.wikipedia.org/wiki/Distributed_Interactive_Simulation)

<sup>3</sup> [https://en.wikipedia.org/wiki/Aggregate\\_Level\\_Simulation\\_Protocol](https://en.wikipedia.org/wiki/Aggregate_Level_Simulation_Protocol)

<sup>4</sup> [https://en.wikipedia.org/wiki/High-level\\_architecture](https://en.wikipedia.org/wiki/High-level_architecture)



The final domain which has been a driver for model integration is the car industry. Their vision has been to be able to create a virtual product by assembling different models that simulate different parts of a physical system in a common simulation. Through the MODELISAR project that ran from 2008 to 2011, the Functional Mock-up Interface (FMI)<sup>5</sup> was defined. Today FMI is managed and developed as a Modelica Association Project. This is the second framework we consider for use in this project.

There have also been some academic attempts to construct alternatives to the HLA, e.g. by Leila et al. (2011) and Jain and McLean (2005). However, these frameworks and middleware are not implemented in available frameworks for us to consider.

For the remainder of this chapter we will compare the High Level Architecture and the Functional Mock-up Interface with respect to our needs in developing a virtual laboratory for AQUAEXCEL2020.

## **High Level Architecture**

The High-Level Architecture (HLA) is a general purpose architecture for distributed computer simulation systems. Using HLA, computer simulations can interact (that is, to communicate data, and synchronize actions) with other computer simulations regardless of computing platforms. The interaction between simulations is managed by a run-time infrastructure (RTI). HLA is an interoperability standard for distributed simulation used to support analysis, engineering and training in a number of different domains in both military and civilian applications.

The high-level architecture consists of three main components:

- Interface specification. This interface defines how HLA compatible simulations cooperate within a run-time infrastructure. An RTI must be compliant with this interface.
- Object Model Template (OMT). Describes what information is transmitted between simulations, and how.
- Rules. The simulation must conform to these in order to be HLA compliant.

The HLA is supported through IEEE standard 1516:

- IEEE 1516–2010 – Standard for Modelling and Simulation High Level Architecture – Framework and Rules
- IEEE 1516.1–2010 – Standard for Modelling and Simulation High Level Architecture – Federate Interface Specification
- IEEE 1516.2-2010 – Standard for Modelling and Simulation High Level Architecture – Object Model Template (OMT) Specification

The HLA uses a publish-subscribe model, where different simulation models, or federates, can join together in a federation. A federate can register an object, and when it changes the attributes of the object, other federates that are listening will be modified. Coordination of the time step is handled by the RTI.

## **Functional Mock-up Interface**

Functional Mock-up Interface (FMI) is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of xml-files and compiled C-code. The FMI standard provides the means for model based development of systems and is used for example for designing functions that are driven by electronic devices

---

<sup>5</sup> <https://www.fmi-standard.org/>

inside vehicles. Activities from systems modelling, simulation, validation and test can be covered with the FMI based approach.

The typical FMI approach is described in the following stages:

- A modelling environment describes a product sub-system by differential, algebraic and discrete equations with time, state and step-events. These models can be large for usage in off-line or on-line simulation or can be used in embedded control systems
- As an alternative, an engineering tool defines the controller code for controlling a vehicle system
- Such tools generate and export the component in an FMU (Functional Mock-up Unit)
- An FMU can then be imported in another environment to be executed
- Several FMUs can – by this way – cooperate at runtime through a co-simulation environment, thanks to the FMI definitions of their interfaces.

The FMI specifications are distributed under open source licenses:

- The specifications are licensed under CC-BY-SA (Creative Commons Attribution-Sharealike 3.0 Unported)
- The C-header and XML-schema are available under the BSD license with the extension that modifications must also be provided under the BSD license.

## Comparison

	Pros	Cons
HLA	<ul style="list-style-type: none"> <li>• Distributed</li> <li>• IEEE standard</li> <li>• Components can join and withdraw in real-time</li> <li>• Several open source RTI's available</li> </ul>	<ul style="list-style-type: none"> <li>• Standard designed specifically for military domain.</li> <li>• Stuck to a specific RTI provider</li> </ul>
FMI	<ul style="list-style-type: none"> <li>• Centralized and distributed</li> <li>• De facto industry standard</li> <li>• Open source licenses</li> <li>• Supports several tools for modelling</li> <li>• Models developed for FMI and exported as FMU's are available in any tool supporting FMI</li> </ul>	<ul style="list-style-type: none"> <li>• Standard designed for automotive industry.</li> <li>• Need to build our own master algorithm for running the FMUs.</li> </ul>

Considering the two frameworks, it should be noted that they are not mutually exclusive. An FMU could be compiled as a HLA federate, and a HLA federate could deliver data through an FMU. However, for our use we have chosen to put weight on the interoperability between different simulation tools, and FMI has a clear advantage here since it supports all tools with FMI support as long as it is compiled according to the standard, while HLA is requiring a potential lock-in with a specific open source community or vendor, depending on the choice of RTI provider.

Based on this comparison, FMI will be used as the framework for developing the AQUAEXCEL2020 virtual laboratories.

## 3. FUNCTIONAL MOCK-UP INTERFACE

The Functional Mock-up Interface (FMI) defines an interface to be implemented by an executable called Functional Mock-up Unit (FMU). The FMI functions are used by a simulation environment to create one or more instances of the FMU and to simulate them,

typically together with other models. An FMU may contain its own solver, in which case we use FMI for Co-Simulation. Alternatively, if the FMU does not contain a solver, we use FMI for Model Exchange. This requires the simulation environment to perform the numerical integration. The goal of the FMI is that calling an FMU within a simulation environment should be reasonably simple.

The FMI for Model Exchange defines an interface to the model of a dynamic system described by differential, algebraic and discrete-time equations and provides an interface to evaluate these equations as needed in different simulation environments. The interface is designed to allow the description of large models. Figure 1 illustrates this interface model.

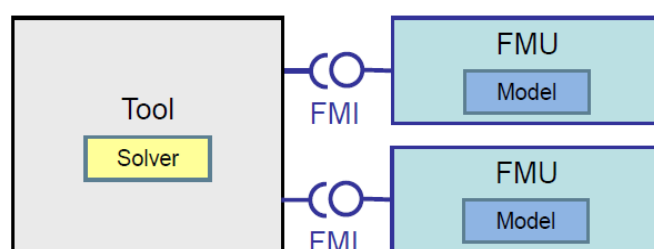


Figure 1: FMI for Model Exchange (From Blochwitz 2014)

Figure 2 shows a schematic view of a model in FMI for Model Exchange format, with data flow between the environment and an FMU for Model Exchange. Blue arrows indicate information provided by the FMU, red arrows information provided to the FMU.

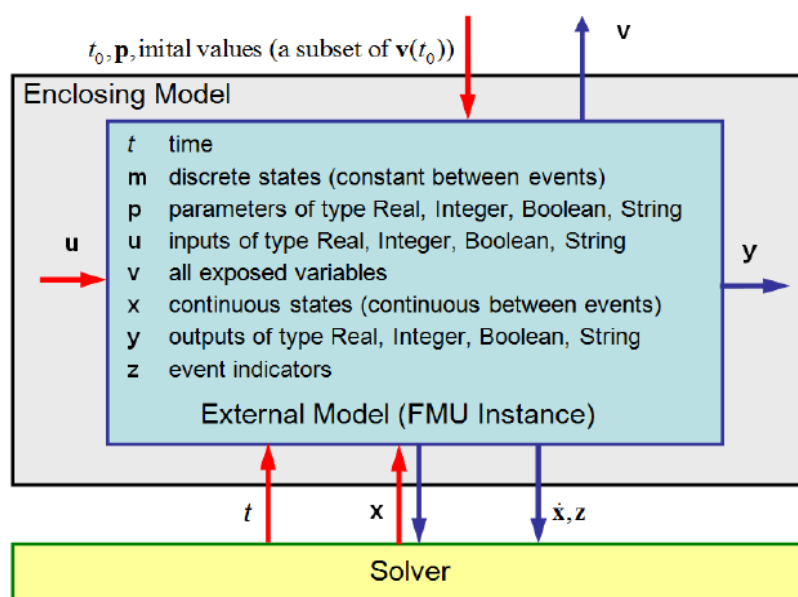


Figure 2: FMI for Model Exchange Signals (From Blochwitz 2014)

The FMI for Co-Simulation is an interface designed both for the coupling of simulation tools as well as coupling with subsystem models, which have been exported by their simulators together with their solvers as runnable code. When coupling different models or tools, the modular structure of coupled problems is exploited in all stages of the simulation process. The individual subsystems handle setup and pre-processing. During time integration the simulation is performed independently for all subsystems, restricting data exchange between subsystems to discrete communication points. Identifying these communication points and the time step for integration will be one of the major research challenges during this project. Finally, visualization and post processing of simulated data is done for each subsystem in its own native simulation tool. Figure 3 illustrates this interface model, and Figure 4 illustrates data flow at communication points.

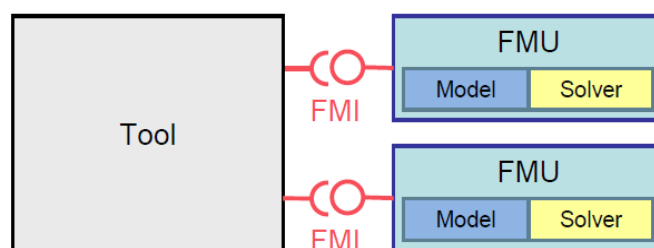


Figure 3: FMI for Co-Simulation (From Blochwitz 2014)

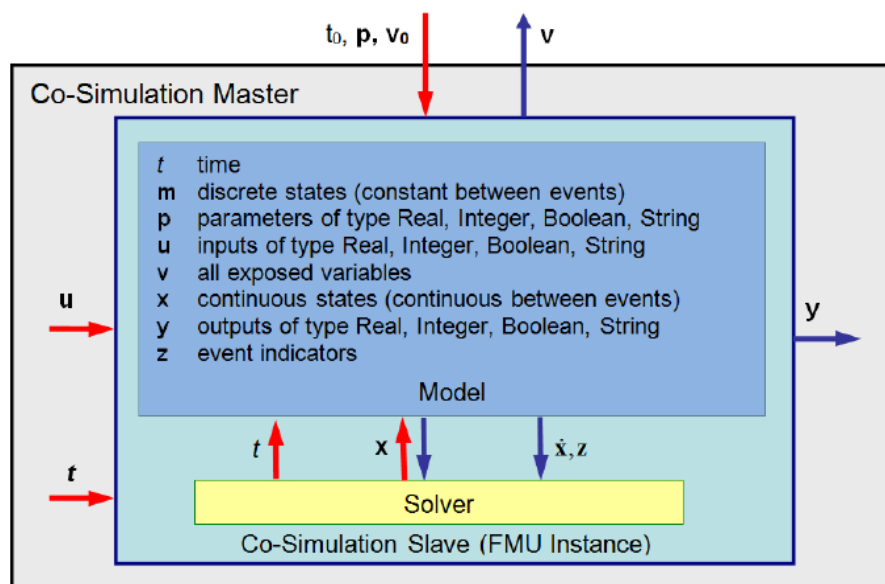


Figure 4: FMI for Co-Simulation Signals (From Blochwitz 2014)

The two interfaces have large parts in common, they were previously specified in two standards, but as of FMI 2.0 they have merged into one standard with a large common structure and two specific structures that can be used depending on which interface is needed.

A component which implements the interface is called a Functional Mockup Unit. It is in essence a zipped file renamed \*.fmu, which contains:

- modelDescription.xml – a description of interface data
- Functionality API in C
- Implementation in source and/or binary form
- Additional data and functionality

One FMU can contain implementations of both interfaces. Figure 5 is an example of the structure within an .FMU file.

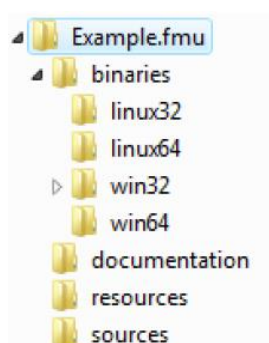


Figure 5: FMU file structure (From Blochwitz 2014)



Since the FMI only specifies the interface needed for a model to communicate, several use cases are possible within this framework.

Figure 6 illustrates a single process run at a single computer. For simplicity only one slave is shown. A master algorithm knows about the FMU and initiates it. The FMU is an FMU for Co-Simulation and is self-contained in how to simulate its model. The Master only needs to set initial values and instruct the FMU how long the simulation should run. If more FMUs were to be simulated together, the master would need to specify communication steps and which data is to be delivered from and to each FMU.

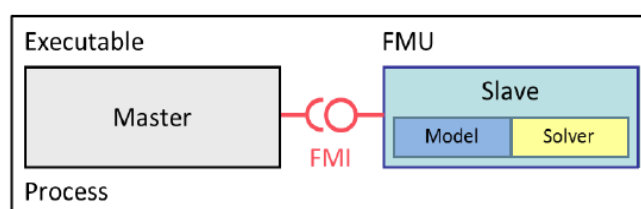


Figure 6: Standalone use case (From Blochwitz 2014)

Figure 7 illustrates the use of a tool used in simulation, e.g. a model compiled as an executable within the Matlab runtime environment. The Master process still initiates the simulation through an FMI interface but instead of being self contained, the FMU contains communication protocols for initiating the second process on the same computer.

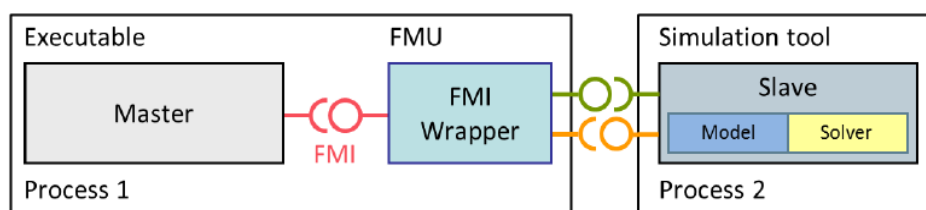


Figure 7: Tool based use case (From Blochwitz 2014)

In its most general form, a tool coupling based co-simulation is implemented on distributed hardware with subsystems being handled by different computers with potentially different OS. The definition of the communication layer is not part of the FMI standard, however distributed co-simulation scenarios can be implemented using FMI as shown in Figure 8. The master then has to implement the communication layer.

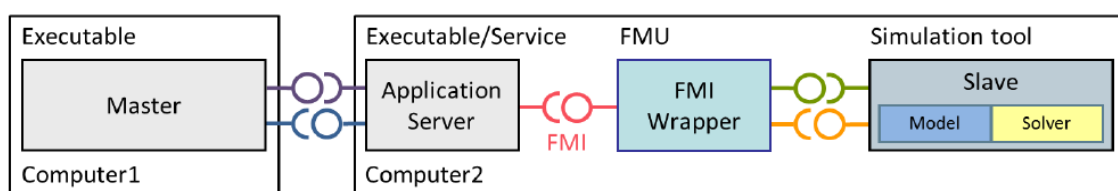


Figure 8: Distributed use case (From Blochwitz 2014)

## 4. SYSTEM ARCHITECTURE

At the highest abstraction level, we refer to Figure 9. The overall system idea is to develop a user interface that will let users specify an experiment in terms of growth, water treatment and hydrodynamic parameters. The user interface will then initiate a master algorithm that will select and initiate the necessary FMU to run the simulation and process the experiment based on the underlying model structure. Results will then be presented to the user through the user interface.

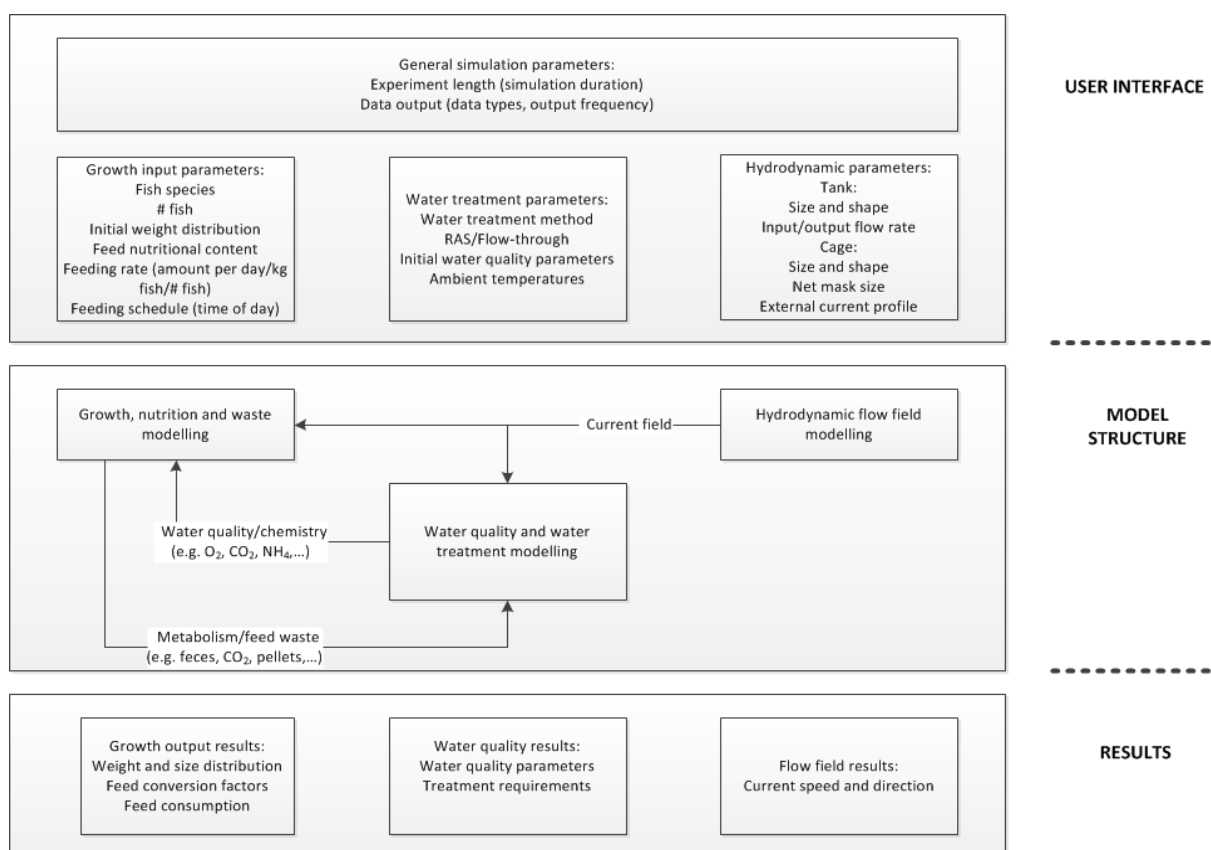


Figure 9: Overall System Specification

The overall initiation of models can be handled with the architecture described in Figure 10. 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). Through previous projects at SINTEF, this middleware has been implemented in an open source library. All that is required to use this system is to create the new FMU's and set up their providers.

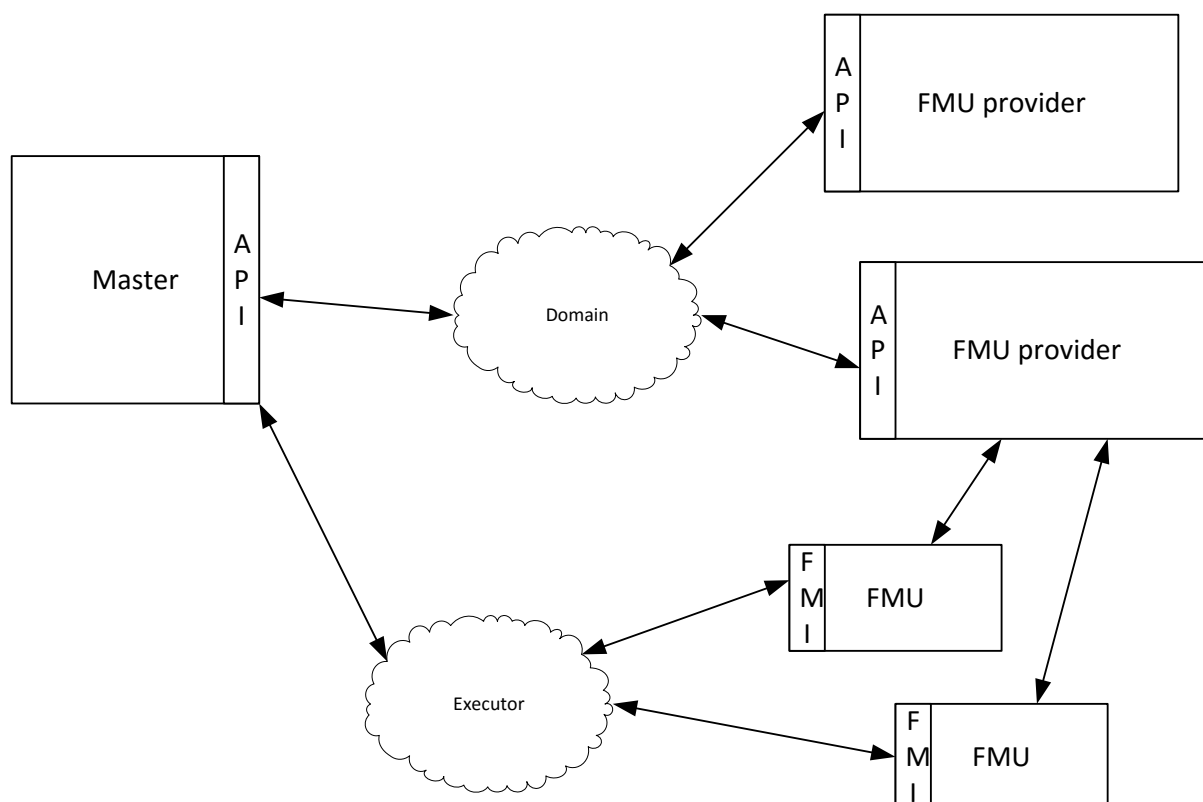


Figure 10: Master initiation architecture

An example simulation after the Master simulation has initiated an experiment with two water tanks running the same hydrodynamic flow field model, but with different growth and water quality models (Stage 2 in Figure 10) would look something like Figure 11.

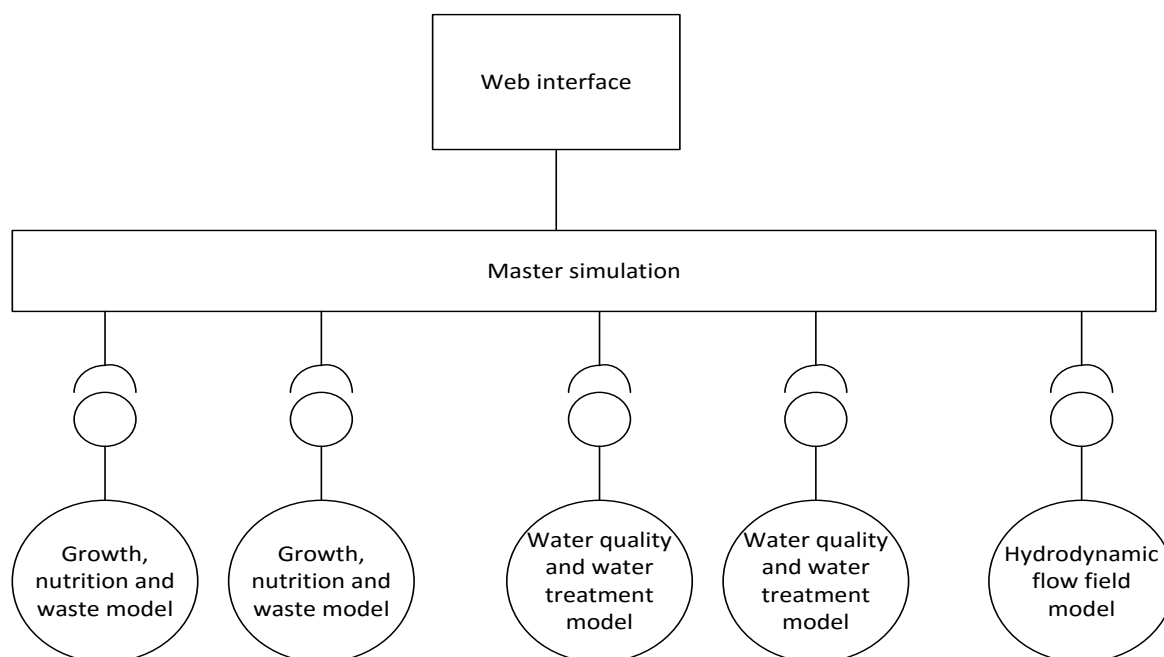


Figure 11: Example simulation

## 5. MODEL DEVELOPMENT GUIDELINES

When using FMI, models may in principle be developed using any programming language or modelling framework as long as it is possible to export models from these languages or tools to FMUs. In AQUAEXCEL2020 we aim 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. Furthermore, the FMUs must offer functions the FMI master can use to:

1. Instantiate the FMU
2. Initialise the FMU
3. Set the variables of the FMU
4. Read the variables of the FMU
5. Stepping the integration internally in the FMU forward in time

There exist today several software tools and packages designed for generating FMUs based on implemented models (<https://www.fmi-standard.org/tools>), many of which are aimed at models implemented in modelling frameworks including MATLAB SIMULINK, Modelica and ANSYS. The number of such tools is at present rapidly increasing together with the number of FMI users within research and industry, and some tools are open source while others require a license. In addition to framework specific tools, FMU support is possible to implement directly into programming languages such as Java, C++ and C by including and linking the source code files with the FMI SDK (<https://www.fmi-standard.org/downloads>). It is also possible to generate FMUs containing models that are implemented in tools or languages that do not support FMU export by wrapping these models in a C or C++ shell which provides FMU functionality. However, this solution will be limited to languages and tools which retain some degree of C/C++ compatibility, and would probably be more labour intensive than using languages or tools with FMU support.

To exemplify how FMUs must be designed to fit with the FMI system used in AQUAEXCEL2020, we will in the following use a hypothetical example where a simple simulation is set up using a model of a fish population coupled with a model of feed consumption. Since FMUs may be generated from models implemented in a wide variety of different languages and platforms, we will primarily focus on the aspects that are parts of the FMI standard rather than looking into the details on how to generate FMUs from different models. The example will adhere to the properties of the FMI master used in AQUAEXCEL2020.

As stated previously, an FMU is essentially a zip-file containing several files and sub-folders. While some of these may only be required in particular cases, or for documentation purposes, others are required for the FMU to be executable by an FMI master:

- *binaries*: folder containing compiled runtime versions of the model
- *sources*: folder containing the source code for the model
- *modelDescription.xml*: file describing the input/output interface of the model, and some of the model properties

While the contents of the "binaries" and "sources" will mostly be determined by the model implementation and platform/language used for implementation, *modelDescription.xml* must be designed such that it clearly states the input/output interface of the model to the FMI master. This file must also offer any additional information the FMI system will need, such as which version of the FMI standard the FMU supports. Some of the information provided in *modelDescription*-files may be similar between FMUs, whereas other parameters are model dependent, and hence pertain to the individual FMUs. Some of the most important of these are:

- *modelName*: textual name used to refer to the FMU type/class when setting up simulations

- *guid*: unique identifier string used to distinguish between different FMUs in the FMI system
- *<ModelVariables>*: segment describing the model variables by providing a unique textual name, a unique reference number, whether it is variable or not, whether it is internal, an input to or an output from the model, as well as an initial value

The *modelDescription*-file for FMU containing the biomass model in our hypothetical example would hence contain parameters reflecting the properties of the model:

```
<?xml version="1.0" encoding="utf-8" ?>
<fmiModelDescription
  fmiVersion="1.0"
  modelName="AQUAEXCEL2020.fishPopulation"
  modelIdentifier="fishPopulation"
  guid="91b2eb04-d47e-4afb-a92d-396b8291ef21"
  author="Martin Føre"
  description="Model of fish population converting feed input to biomass"
  version="1.1"
  variableNamingConvention="structured"
  numberOfContinuousStates="0"
  numberOfEventIndicators="0">

  <ModelVariables>
    <ScalarVariable name="FeedConsumption" valueReference="0" variability="continuous"
      causality="input" ><Real start="0.0" /></ScalarVariable>
    <ScalarVariable name="Biomass" valueReference="1" variability="continuous"
      causality="output"><Real start="2.0" /></ScalarVariable>
  </ModelVariables>

  <Implementation>
    <CoSimulation StandAlone>
      <Capabilities>
        canHandleVariableCommunicationStepSize="true"
        canBeInstantiatedOnlyOncePerProcess="false" />
      </CoSimulation_StandAlone>
    </Implementation>
  </fmiModelDescription>
```

In this example we've chosen a naming convention describing both which project the FMU pertains to and the scope of the model contained within the FMU, hence *modelName* is set to "AQUAEXCEL2020.fishPopulation". Further, our hypothetical model has a very simplified input/output regime, where the model only has one scalar input variable (*FeedConsumption*) and one scalar output variable (*Biomass*). Both of these are defined in the *modelDescription*-file as continuous input/output variables, with initial *Biomass* being set to 2.0 kg.

Correspondingly, the *modelDescription*-file for the feed model FMU would contain the information required to describe the contained feed model:

```
<?xml version="1.0" encoding="utf-8" ?>
<fmiModelDescription
  fmiVersion="1.0"
  modelName="AQUAEXCEL2020.feed"
  modelIdentifier="feed"
  guid="91b2eb04-d47e-4afb-a92d-396b8291ef22"
  author="Martin Føre"
  description="Model of feed consumption based on fish population biomass"
  version="1.1"
  variableNamingConvention="structured"
  numberOfContinuousStates="0"
  numberOfEventIndicators="0">

  <ModelVariables>
    <ScalarVariable name="Biomass" valueReference="0" variability="continuous"
      causality="input" ><Real start="2.0" /></ScalarVariable>
    <ScalarVariable name="FeedConsumption" valueReference="1" variability="continuous"
      causality="output"><Real start="0.0" /></ScalarVariable>
  </ModelVariables>

  <Implementation>
```



```

    <CoSimulation_StandAlone>
      <Capabilities
        canHandleVariableCommunicationStepSize="true"
        canBeInstantiatedOnlyOncePerProcess="false" />
      </CoSimulation_StandAlone>
    </Implementation>
  </fmiModelDescription>

```

Using the same naming convention, this FMU is given the *modelName* "AQUAEXCEL2020.feed". Further, the *guid* value is different from that used in the "AQUAEXCEL2020.fishPopulation" FMU, such that these two are possible to distinguish. This FMU has two variables which mirror those in the biomass-FMU, however with *Biomass* being an input variable and *FeedConsumption* being an output.

To set up a simulation using FMI, it is necessary to specify which FMUs should be used in the simulation, and how these models should be interconnected. This information needs to be provided to the FMI system so that the FMI master is able to instantiate and interconnect the FMUs properly. Since the contents of the *modelDescription*-files represents the only information the FMI master has on the FMUs, it is important that the simulation setup description adheres to this information in using the same model names (*modelName*) and variable names (in *<ModelVariables>*) as used in the *modelDescription*-files of the desired FMUs. The *Executor* application in the FMI master used in the virtual laboratory in AQUAEXCEL2020 (Figure 10) requires this information to be delivered in the form of a *sysconf*-file, which is a text file listing the model instances that will be included in the simulation, and how these model instances will be interconnected. In our hypothetical example setup this file could be set up in this manner:

```

name "DEMO: Interacting fish population and feed models"

slaves {
  Population1 {
    type AQUAEXCEL2020.fishPopulation
  }
  Feed1 {
    type AQUAEXCEL2020.feed
  }
}

connections {
  Population1.FeedConsumption Feed1.FeedConsumption
  Feed1.Biomass                Population1.Biomass
}

```

The segment *slaves* in this file lists the models to be used in the simulation and which types of FMU these should be instances of. Although this example only contains single instances of the FMU types AQUAEXCEL2020.fishPopulation (*Population1*) and AQUAEXCEL2020.feed (*Feed1*), it is possible to create an arbitrary number of instances of any given FMU type in an FMI system (as illustrated in Figure 11). When the *Executor* application has parsed this information, it will request instances of the selected FMU types from an appropriate *FMU Provider* (see Figure 10). The *Executor* then assigns values to the input variables of the FMUs as described in the *connections* segment of the file. This is done by writing the name of an input port and the output port connected to this input port in the same line spaced by a void. For our hypothetical case this is done by setting the input variable *FeedConsumption* in the *fishPopulation* instance named "Population1" to the *FeedConsumption* output variable in the *feed* instance named "Feed1", and vice versa for the *Biomass* variables in both FMUs. This results in a simple model setup with two models interacting through their respective input/output variables (Figure 12).

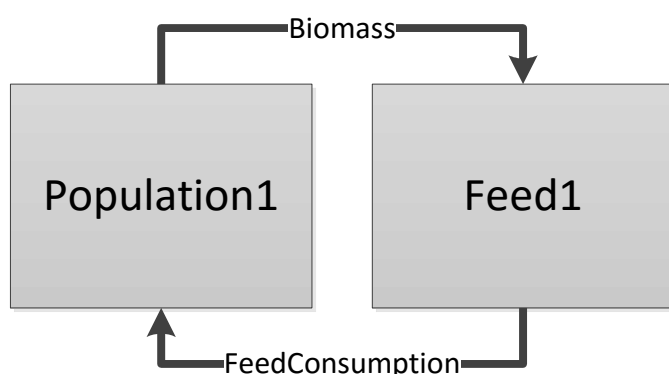


Figure 12: Model setup of the simple hypothetical example.

Since FMI for co-simulation assumes that all models handle their state advancement and integration internally, there are only three parameters that need to be specified to initialise a simulation, namely the start time, stop time and communication time step size used in the simulation. The *Executor* application used in this project requires this information to be provided through a simple text file with *.execonf* extension, and that these times are given in seconds. For example, the execution of a simulation lasting 1 day (=24 h = 86400 s) with a time step of 100 s would require an *execonf*-file containing three lines:

```
start 0.0
stop 86400
step_size 100
```

Once the *Executor* has parsed this information the simulation is ready to start, and the user is prompted to start the simulation at his/her leisure by a key stroke.

The outputs from a simulation using FMI will typically include all the variables defined in the FMUs at all communication times. In the FMI system used in AQUAEXCEL2020 this is realised by creating one csv-file per FMU used in a simulation, and that all variables pertaining to this FMU are stored in that csv-file. The output files from the FMUs used in the hypothetical simulation could hence look like this:

Time,FeedConsumption,Biomass	Time,Biomass,FeedConsumption
100,0,2	100,2,0
200,1,2.6	200,2.6,1
300,1.1,3.5	300,3.5,1.1
300,0.6,4.1	300,4.1,0.6
400,0.95,4.7	400,4.7,0.95
500,0,4.7	500,4.7,0
600,0,4.7	600,4.7,0
700,1.2,5.5	700,5.5,1.2
800,1.5,6.5	800,6.5,1.5
900,0.5,6.8	900,6.8,0.5
1000,0.6,7.0	1000,7.0,0.6

Table 1: Possible contents of the output files from a fishPopulation FMU (left column) and a feed FMU (right column) over a simulation of 1000 seconds. All data are fictional.

The time series for the variables *FeedConsumption* and *Biomass* occur in both output files as they are input variables in one FMU and output variables in the other. In our hypothetical

simulation scenario, the biomass starts at 2.0 kg and increases to 7.0 kg through the simulation, while the feed consumption also increases with increased biomass (Table 1).

## 6. CODE REPOSITORY

In order to ensure proper security, backup and versioning history of the code developed in this project, SINTEF will provide access to their code-collaboration server (code.sintef.no) for the project partners. Primarily we will use Bitbucket<sup>6</sup> (previously Stash) which provides code management through a Git versioning system. Other services that are available on the server and might be considered for use in the project are Confluence<sup>7</sup> for knowledge sharing, Jira<sup>8</sup> for issue tracking and Bamboo<sup>9</sup> as build server. Secure access will be handled by requiring all participants to have a valid X.509 certificate issued by SINTEF and direct Git-access to Bitbucket will only be available through ssh-keys. It will be up to the project participants which git-client they prefer to access the server in order to download or upload code. For those unfamiliar with the Git versioning system, we recommend the book Pro Git<sup>10</sup>, available for free online. SINTEF will provide support in acquiring access to code.sintef.no for the project participants.

## 7. CONCLUSION

For the AQUAEXCEL2020 project we will use the FMI for Model Exchange and Co-Simulation standard, available at <https://www.fmi-standard.org/downloads>

The user interface and the FMI master will be developed as part of task 5.4 in the WP5 work-package, while the FMU's will be developed in tasks 5.1 to 5.3 (see Figure 10 and Figure 11).

In order to verify that each partner conforms to the standard we will use the FMU Compliance Checker 2.0.1, also available at <https://www.fmi-standard.org/downloads>

To ensure security, backup and versioning history, we will use SINTEF's code-collaboration server code.sintef.no which provides the Git-versioning system Bitbucket.

## References

- Jalali, L., Mehrotra S., Venkatasubramanian, N., *Multisimulations: Towards Next Generation Integrated Simulation Environments* in Formal Modeling: Actors, Open Systems, Biological Systems Volume 7000 of the series Lecture Notes in Computer Science, Agha, G. Danvy, O., Meseguer, J. eds, 352-367, 2011, Springer-Verlag Berlin Heidelberg
- Jain, S., McLean, C. R. *Integrated Simulation and Gaming Architecture for Incident Management Training* in Proceedings of the 2005 Winter Simulation Conference, 2005, 904-913
- Blochwitz, T., *Tutorial: Functional Mockup Interface 2.0 and HiL Applications*, presentation from the 10<sup>th</sup> International Modelica Conference 2014 available at <https://www.fmi-standard.org/literature>

<sup>6</sup> <https://www.atlassian.com/software/bitbucket>

<sup>7</sup> <https://www.atlassian.com/software/confluence>

<sup>8</sup> <https://www.atlassian.com/software/jira>

<sup>9</sup> <https://www.atlassian.com/software/bamboo>

<sup>10</sup> <https://git-scm.com/book/en/v2>



## Glossary

AQUAEXCEL<sup>2020</sup>: AQUAculture Infrastructures for EXCELlence in European Fish Research towards 2020

HLA: High Level Architecture

RTI: Run time infrastructure

OMT: Object Model Template

FMI: Functional Mock-up Interface

FMU: Functional Mock-up Unit



## 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

<b>EU Project N°</b>	652831	<b>Acronym</b>	AQUAEXCEL <sup>2020</sup>
<b>Full Title</b>	AQUAculture Infrastructures for EXCELlence in European Fish Research towards 2020		
<b>Project website</b>	<a href="http://www.aquaexcel.eu">www.aquaexcel.eu</a>		

<b>Deliverable</b>	<b>N°</b>	D5.1	<b>Title</b>	Model development guidelines
<b>Work Package</b>	<b>N°</b>	5	<b>Title</b>	Virtual laboratories and modelling tools for designing experiments in aquaculture facilities

<b>Date of delivery</b>	<b>Contractual</b>	30/06/2016 (Month 9)	<b>Actual</b>	30/06/2016 (Month 9)
<b>Dissemination level</b>	X	<b>PU Public, fully open, e.g. web</b>		
		<b>CO Confidential, restricted under conditions set out in Model Grant Agreement</b>		
		<b>CI Classified, information as referred to in Commission Decision 2001/844/EC.</b>		

<b>Authors (Partner)</b>				
<b>Responsible Author</b>	<b>Name</b>	Finn Olav Bjørnson	<b>Email</b>	Finnolav.bjornson@sintef.no

<b>Version log</b>			
<b>Issue Date</b>	<b>Revision N°</b>	<b>Author</b>	<b>Change</b>
20/05/2016	0	Finn Olav Bjørnson	First version
25/05/2016	1	Gunnar Senneset	First review by WP leader
17/06/2016	2	Finn Olav Bjørnson	Revision after review by assigned 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 <sup>2020</sup> deliverable template (title page, styles etc)	X	<i>Available in “Useful Documents” on the collaborative workspace</i>
<b>The draft is ready</b>			
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 <sup>nd</sup> Reviewer and to the Project coordinator (cc to the project manager) for approval	X	<i>Send the final draft to your WPLLeader, the 2<sup>nd</sup> Reviewer and the coordinator with cc to the project manager on the 1<sup>st</sup> 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>

## Annex 2: Survey


### HCMR

								
<b>WP 5 Survey</b>								
<p>The purpose of the survey is to get an overview of modelling tools developed by or used by the WPS partners. Only tools relevant for the WPS tasks should be included.</p> <p>The survey distinguishes between models developed inhouse and commercial/open source packages.</p> <p>Guidelines for some of the topics are embedded in the cell headlines.</p> <p>If available, please refer to web-pages for documentation or send as separate attachments.</p>								
<b>Commercial/Open source</b>								
System name	System description	Run-time/development environment	Type of software licence agreement	Source code available (yes/no/partial)	Data input interface/format	Data output interface/format	Documentation link/attachment	Contact person (name, e-mail)
DEBtool_M	A software package that can be used to illustrate some implications of the Dynamic Energy Budget (DEB) theory and to apply this theory in the analysis of eco-physiological data; it also allows to fit DEB models to data. Applications to aquaculture (e.g. fish growth, feed consumption, oxygen consumption, reproduction rates).	Windows, Linux, MacOS/Matlab	No licence for the DEBtool. Licence for Matlab is required.	yes	ASCII, CSV, XLS	ASCII, CSV, XLS	<a href="http://www.bio.vu.nl/th/b/deb/deblab/debtool/DEBtool_M/manual/index.html">http://www.bio.vu.nl/th/b/deb/deblab/debtool/DEBtool_M/manual/index.html</a>	Bas Kooijman (bas.kooijman@vu.nl)
DEBtool_R	Work in progress to convert (most) DEBtool_M using the R software environment.	Windows, Linux, MacOS/R	No licence is required for the DEBtool. R is a Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form.	yes	ASCII, CSV, XLS	ASCII, CSV, XLS	not available yet	Gonçalo Marques (goncalo.marques@tecnico.ulisboa.pt)

### JU

								
<b>WP 5 Survey</b>								
<b>Commercial</b>								
System name	System description	Run-time/development environment	Type of software licence agreement	Source code available (yes/no/partial)	Data input interface/format	Data output interface/format	Documentation link/attachment	Contact person (name, e-mail)
ANSYS CFD (Fluent)	Fluid flow simulation including heat and mass transfer, reactions, multiple phase systems, particle flows, etc.	C – language can be used to create user-defined functions and extend, for example, the turbulence models, ...	Proprietary licence (special licence for education and research)	no	ABAQUS, Enight, I-DEAS, NASTRAN, PATRAN, TECPLOT, ...	CSV, ABAQUS, Enight Case Gold, I-DEAS, NASTRAN, PATRAN, TECPLOT, ...	<a href="http://www.ansys.com">www.ansys.com</a>	Stepan Papacek, spapacek@frov.jcu.cz
<b>Commercial</b>								
System name	System description	Run-time/development environment	Type of software licence agreement	Source code available (yes/no/partial)	Data input interface/format	Data output interface/format	Documentation link/attachment	Contact person (name, e-mail)
COMSOL Multiphysics	Software package based on finite element method for corresponding PDEs integration. It is especially well suited for coupled phenomena (multiphysics) in various physics and engineering applications.	In addition to physics-based user interfaces (COMSOL Desktop, Application Builder), it allows entering coupled systems of partial differential equations (PDEs).	Proprietary licence (personal)	no	Matlab	Matlab	<a href="http://www.comsol.com">www.comsol.com</a>	Stepan Papacek, spapacek@frov.jcu.cz

## Nofima



## WP 5 Survey

The purpose of the survey is to get an overview of modelling tools developed by or used by the WPS partners. Only tools relevant for the WPS tasks should be included.

The survey distinguish between models developed inhouse and commercial/open source packages.


Guidelines for some of the topics are embedded in the cell headlines.

If available, please refer to web-pages for documentation or send as separate attachments.

### Commercial/Open source

System name	System description	Run-time/development environment	Type of software licence agreement	Source code available (yes/no/partial)	Data input interface/format	Data output interface/format	Documentation link/attachment	Contact person (name, e-mail)
STELLA from ISEE Systems	Object-based simulation environment. Simulate system development in time. From small to huge simulations. Used in Nofima for many purposes, e.g. RAS dimensioning or experimental outcome predictions	GUI. Finished models can be saved as standalone runtime	Commercial	No	GUI, database, Excel, csv	GUI, pdf, csv	<a href="http://www.iseesystems.com/software/Education/StellaSoftware.aspx">http://www.iseesystems.com/software/Education/StellaSoftware.aspx</a>	<a href="mailto:bendik.terjesen@nofima.no">bendik.terjesen@nofima.no</a>

## SINTEF



WP 5 Survey

FhSim simulation framework developed by SINTEF

In-house								
System name	System description	Run-time/development environment	Type of software licence agreement	Source code available (yes/no/partial)	Data input interface/format	Data output interface/format	Documentation link/attachment	Contact person (name, e-mail)
FhSim	Time-domain simulation of aquaculture systems and operations (e.g. flexible cages in currents and waves, fish growth and behaviour, feed distribution in sea cages). Systems are modelled using sub-models. 3D visualisation.	Windows, Linux C++ (support for other languages)	Proprietary licence (special licence for education and research)	Partial (subject to agreement)	XML GUI (for special applications)	CSV NetCDF	Reite, K.-J., et al (2014), «FHSIM – Time Domain Simulation of Marine Systems», Proceedings of the ASME 2014 33rd International Conference on Ocean, Offshore and Arctic Engineering, San Francisco, USA.	Martin Føre (martin.fore@sintef.no)