User Manual

HSY Combined Sewer
Real-Time Simulations
ÖVERI Setti

Fluidit Ltd
Mannilantie 44 A
04400 Järvenpää, Finland
+358 10 526 9780

www.fluidit.com

HSY

WATER PROTECTION PROGRAMME

CONTENTS

Version	Date	Author	Verification	Project Coordinator
2	3.2.2023	Pedro Almeida	Hannes Björninen	Doris Kalve, HSY
1	19.12.2022	Pedro Almeida, Hydroinformatics Engineer, Fluidit Ltd	Hannes Björninen Lead Engineer Fluidit Ltd Markus Sunela CTO Fluidit Ltd	Leena Sänkiaho, Development Engineer, HSY

APPENDICES

TABLES

FIGURES

1 INTRODUCTION

This report documents the technical aspects of HSY's ÖVERI "Setti" sub-project. The goal was to develop and pilot an online model that performs near real-time simulations of the HSY combined sewer system (also referred as *HSY-CS-model/HSY-SKV-Malli*) to report combined sewer overflows (CSO).

This document summarizes the technical aspects of the pilot stage, and presents suggestions for the next project phases for operational implementation and further development. The project was partially funded by the Ministry of the Environment's *Water Protection Programme*[1]. The work was undertaken in close collaboration between the three project team members:

- Helsinki Region Environmental Services Authority (HSY)
- Envera Oy (also referred as Neuroflux/Smartvatten in this report)
- Fluidit Oy

During the pilot phase the real-time simulation system was deployed to a test environment which was created in a server managed by HSY with VPN and remote desktop access provided to Fluidit. In the test environment it was possible to set up the data connections, run the simulations, review results, and identify the needs for the next phases of the project. The next phases will aim to address all the needs encountered during the pilot phase and deploy the real-time simulations to a production environment from where the model's results can be effectively considered in HSY's daily operations.

2 OVERVIEW OF REAL-TIME SYTEMS

A hydraulic model, such as the HSY-CS-Model, can be used for several tasks, such as:

- Plan new developments / design work
- Prepare operations by exploring different scenarios
- Investigate current issues in the network
- Estimate pollutant's discharge
- Identify critical locations

Since the model is a digital representation of the network, it can also be used to assist operational tasks when embedded in a real-time system. This application can also be referred as *online model, early warning system, forecasting system, digital twin, decision support system,* and more. The basic idea is that input data acquired in near real-time, such as precipitation, feeds the model, which runs simulations with a pre-defined frequency producing results (e.g. every 1h). All the process within the real-time system runs automatically instead of manually. Results can be generated for the past, using historical observational data, and for the future, using forecasted data. See Figure 1 for a visual interpretation of the time frame.

[1] Ministry of the Environment, Water Protection Programme, https://ym.fi/vedenvuoro

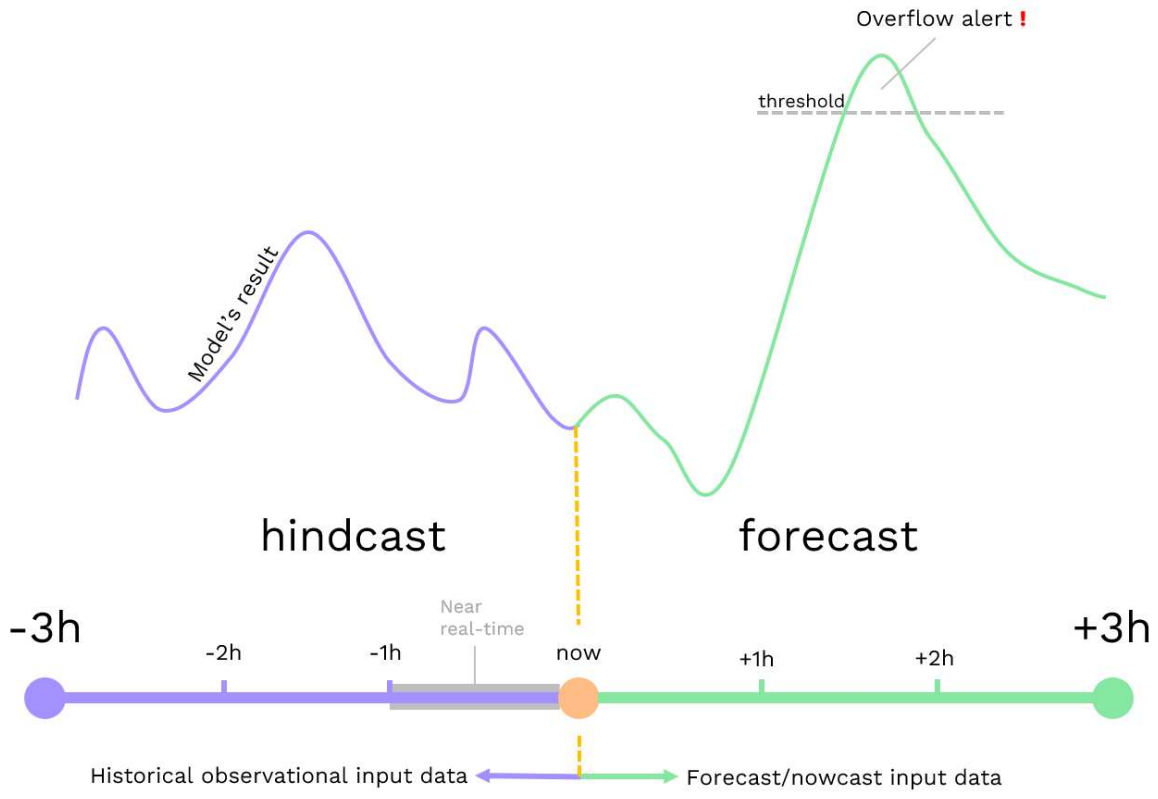


Figure 1. Simplified time frame scheme of a real-time system

The main differences between the hindcast and forecast models are outlined below:

Hindcast:

- <u>The model is fed with historical observational data</u>
  For instance, precipitation, temperature, and external inflows. External inflows refer to measured or estimated flow time series used to represent areas not included in the model. These areas are usually located at the borders of the network model, but the incoming flows are still relevant for the simulations. For example, the flows discharged from Suomenlinna island.

- <u>Model's results can be compared with measured data</u>
  For instance, flow or level measurements placed at the network in different locations. Measurements are the closest "estimation" of what truly happened. At these locations, results from the model can be compared with measurements to indicate, how well the model is reproducing measured values and how well the measurements are working.

- <u>Estimation of the past state in the entire network</u>
  Since measurements are available only at few locations, the model, when calibrated, gives the best estimation of what is happening in the entire network and where the measurements are not available or cannot be trusted.

Forecast:

- The model is fed with forecasted data
  For instance, forecasted weather data, such as precipitation and temperature, but also forecasted dry-weather inflows and direct inflows from the bordering areas.

- More uncertain results
  The model's results are directly impacted by the input data and its quality (e.g. precipitation). When the input data itself is a forecast (i.e. estimated values) the results of the model will also be more uncertain. The uncertainty can be also estimated in forms of probabilities. These probabilities can be distributed together with model's results similarly as it is used in weather forecasts (e.g. 80% chance of overflow in certain location).

- Estimation of the future state in the entire network
  The model uses the latest results, such as water depths, quality parameters, amount of snow and moisture in the soil, obtained from the hindcast as the initial condition. From there, the future state of hydraulic and hydrological variables is estimated for the entire model's area. [2]

There are several benefits of embedding the model into a real-time system, such as:

- Quickly visualize the current state of the network at any location
- Reduced or no manual work needed to generate results
- Measurements and model's results combined
- Effectively use all data produced in favour of the operations
- Results can be accessed easily by many different stakeholders
- Real-time system can form basis for an early-warning system able to produce alerts, such as overflow/flooding
- Improve communication with authorities in case of extreme events
- Enables other real-time projects, such as dynamic control of the network. For instance, aiming to improve energy efficiency as explored in Tampere or using real-time-control (RTC) systems to reduce overflows.

A real-time system, as described here, provides the ability to understand what happened in the near past and what is likely to happen in the near future. This information is especially critical for understanding the impacts of an extreme event a few hours in advance (forecast), but also quickly identify what were the problems that occurred during the event (hindcast). Adopting a real-time system to utility operations is an important step towards a more resilient infrastructure.

Only hindcast simulations were included and tested in the pilot phase of the real-time system. Forecasting capabilities are discussed in sections 5 and 6.

[2] Markus Sunela. *Real-Time Control Optimization of Water Distribution System with Storage*. 2017. ISBN 9789949831654. doi: 10.3917/lett.051.0109.

3 PILOT STAGE SET UP

3.1 General

It was decided to implement the real-time system step-by-step, starting with the nowcasting (also referred as hindcast model in this report). Nowcasting was the easiest milestone to achieve for this specific project since historical observational data was already familiar to HSY's workflows. Another reason to start with the hindcast is that the HSY-CS-model has been partially calibrated with historical observational data. Hindcasts, however, have their own challenges. For instance, the frequency of simulations is limited by the frequency of which input data is acquired.

Figure 2 contains a diagram of the hindcast from the perspective of an operator to whom the information would be relevant. During the pilot phase, the process to run a simulation happened every 1h beginning at exact clock hours, for instance, 09:00, 10:00, and 11:00 ● as depicted in the diagram. The first steps of the process involve fetching input data, setting up and running the simulation. In the diagram, the *time delay* Td represents the time it takes from the beginning of the process until the results are generated ▮. Deeper discussion about the time delay during the pilot phase is available in section 14.

As the process is repeated every 1h, and there is a delay until the results are available, the latest results are always delayed in relation to the clock time. Figure 2 also contains examples of delays that would be experienced by the operator when checking the results in different instants ▮. For instance, a 65min delay in the model's results would be observed when checking the results at 09:05 since the process to generate new results would be still running until ~ 09:10.

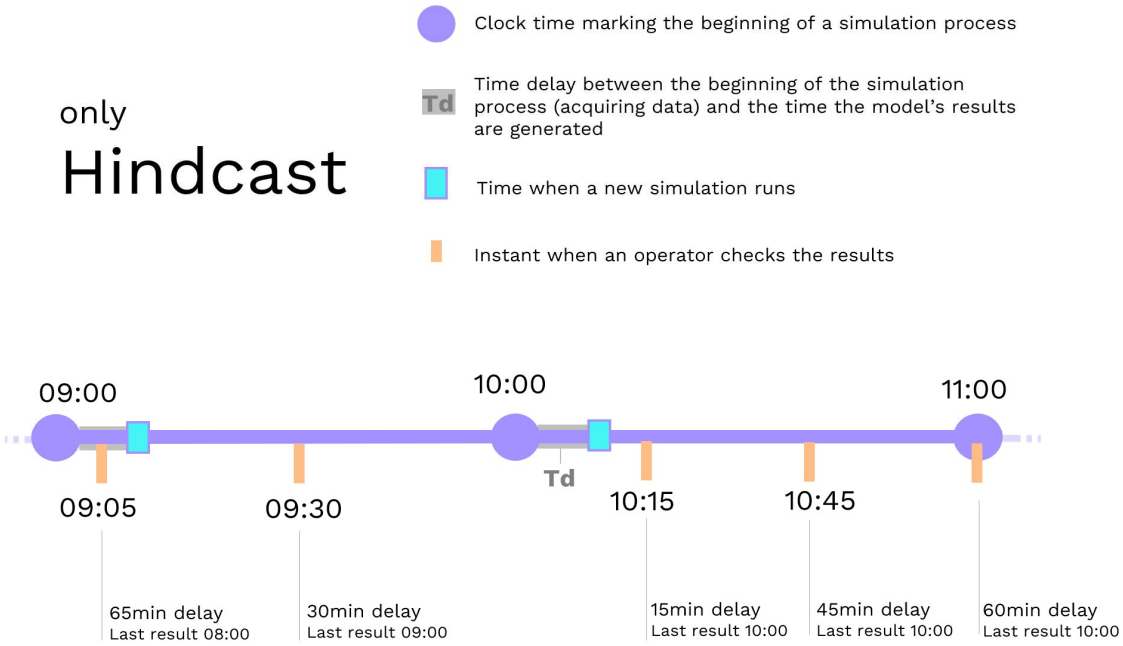


Figure 2. Diagram of a hindcast system. Results' delays from an operator's perspective.

More on result's delay from operator's perspective:

- Td = time to obtain input data + simulation's run time + time to transfer results
- Minimum possible result's delay = Td *(~ 10min in* Figure 2*)*
- Maximum possible result's delay = process frequency + Td *(~ 70min in* Figure 2*)*

Figure 16 illustrates how the delays from the operator's perspective could change once the process frequency is updated and forecasts are included.

3.2 Test environment

A server has been provisioned/ordered by HSY's IT department to this pilot project. The environment's requirements are driven by Fluidit Storm software, the HSY-CS-Model, and the nature of real-time continuous simulations.

- **Basic requirements for Fluidit Storm Online**
  - 64-bit operating system: Windows 10 or 11, Mac OS X 10.10 (or newer), Linux (e.g. Ubuntu 20.04, CentOS 9)
  - Min 2 cores (e.g., Intel i5 or AMD A10)
  - Min 4 GB RAM
  - Min 5 GB hard disk space, but in practice much more, depending on for how long period the historical simulation results are to be saved
  - Java 11 runtime, 64-bit, support for version 11–17 (64-bit). For Windows, Fluidit exe-installers include the latest Java 17 release from OpenJDK.
  - License and product key.
  - Refer to the software's installation guide or support@fluidit.com for more details.
- **Basic requirements for real-time continuous simulations**
  - Access to the input data (if the necessary data is not stored on the server)
  - On for 24/7
  - Software or framework to drive unattended processed with specified frequency (e.g. task scheduler in Windows or cron in Unix)
- **Basic details of the test environment in this project**
  - On for 24/7
  - Windows Server 2019 Standard
  - Processor: Intel® Xeon® Gold 6226 CPU @ 2.70GHz
  - Installed Memory: 16 GB
  - System type: 64-bit Operating system
  - Domain: hsy.local
  - Amazon Corretto 17 Java release installed
  - Internet access
  - Remote desktop access for Fluidit users in order to set up the software. Accessed via VPN.
  - Managed by HSY's IT department

3.3 Overview of the simulation process

There are three main operations for a real-time simulation. The operations are executed with a pre-defined frequency (1h for the pilot phase). The three main operations are:

1. Obtain input data
2. Simulate
3. Transfer results

The three main operations are repeated with certain frequency (e.g. every 1h). From the model's perspective, however, the simulations are continuous. Therefore, when a simulation ends the model's state is saved at the last time step to a SWMM's hot start file. When the next simulation starts, the hot start file is loaded, and the model continues from where the last simulation ended.

To summarize, there are two important elements that drive the process of the real-time system. Brief descriptions of them provided below and a the diagram of the process in Figure 3:

1. Fluidit Storm+ software
   The platform where the HSY-CS-model is managed. The Fluidit platform also has all infrastructure related to handling the model, setting up data connections (Dataport), data pre- and post-processing and reporting results. Hence, the model and the Fluidit platform are the central piece of the system. Fluidit Storm+ software was installed by Fluidit in the test environment.

2. Neuroflux and Finnish Meteorological Institute (FMI) API
   The simulation process requests input data from these application programming interfaces (APIs). In case of Neuroflux's API, it is also used for storing the simulation results at the end of a simulation. All the connection settings are stored in the model and can be modified using Fluidit Storm's user interface. More details on data connections are described in section 3.5.

From Figure 3, only FMI and Neuroflux API are not installed in the test environment (Windows Server). However, the settings of the connection between Fluidit Storm and the APIs are stored in the model.

3.4 Set up and tasks division

3.4.1 HSY tasks

- Manage the project
- Create the test environment
- Manage accesses to the test environment
- Review results
- Decision about next phases

3.4.2 Neuroflux tasks

- Make sure the API is working

- Manage the access to the API together with HSY ICT
- Inform Fluidit about the available data tags, their units etc.
- Receive simulation results
- Display simulation results as time series during the pilot phase.

3.4.3 Fluidit tasks

- Install the software in the test environment
- Create and manage license and software users
- Update the model for real-time requirements during the pilot phase
- Create online modelling solution
- Review results

It is important to remember that model used by the system (HSY-CS-model) is not static. Instead, the model is updated from time to time to reflect the current state of the network. To obtain the most realistic results, the real-time process should always use the latest stable version of the model. For that, the model itself could be managed in a more structured manner discussed in section 5.2.

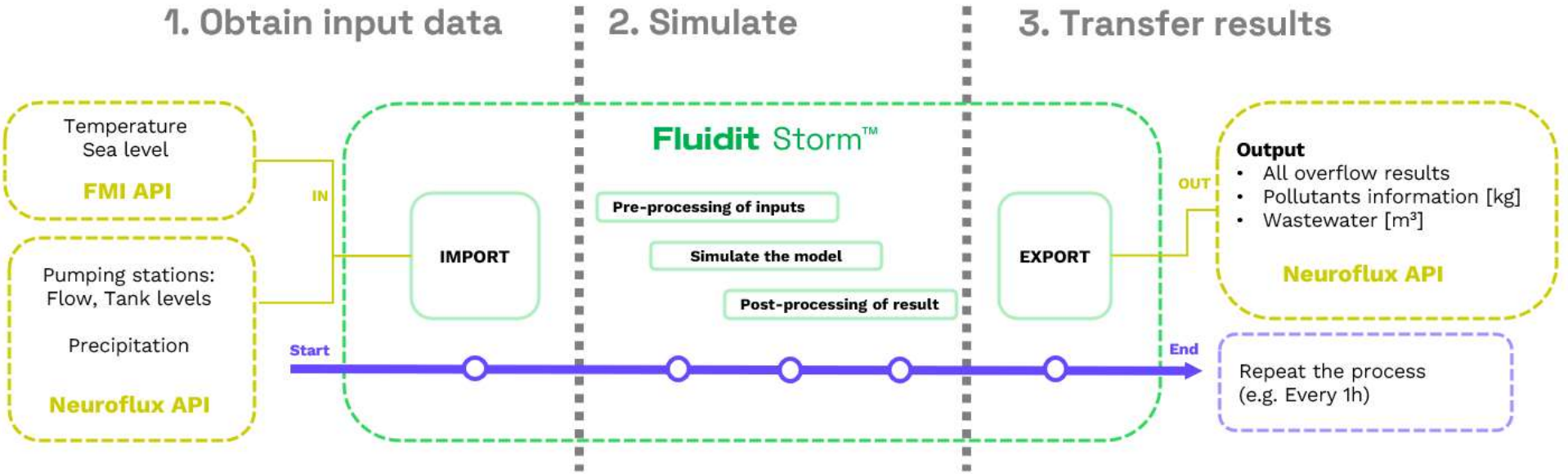


Figure 3. Process to run 1 simulation

3.5 Data sources

Data connections are set up using Fluidit's data source component. A data source component is created in Fluidit Storm for each source. During the pilot project a data source component was created for both the Finnish Meteorological Institute (FMI) API and for Neuroflux's API. It is possible to fetch data from the sources, but also send data to the sources. For instance, precipitation and measured flows are fetched from Neuroflux's API and used as input data for a simulation. Once the simulation is over, model's results are sent back to Neuroflux's API (more details of the process in 3.3). To access the data sources, open Fluidit software, then go to [Model] > [Data sources].

FMI data source

FMI is a weather data source. In Fluidit Storm, by the time of this project, two weather providers are included by default: FMI (coverage for Finland) and NOAA (global coverage). Creating such data source is simple. Go to [Model] > [Data sources] > New > Weather Data Source > select the provider (FMI in this case).

Neuroflux data source

Neuroflux data source can be accessed in the model in the same path as other sources. To gain access to this source the user must obtain the URL, key certificate, and private key from the data provider (Neuroflux).

Data sources inform to the model the location where it will fetch data. This is however not enough to specify what data will be fetched from the source. For that, data source tags components are created.

3.6 Data source tags

A data source tag is used to map data from the source to a variable that can be used in Fluidit software (e.g. as input to the model). For instance, the HSY-CS-Model uses sea level data as input (boundary conditions). During the pilot phase, the sea level data was fetched from FMI's data source. Since FMI provides data on several different weather parameters, there is the need to inform which "tag" refers to the sea level data. A data source tag component can be added/visualized in the software's UI via [Model] > [Data source tags]. Check Figure 4 for an example of the data source tag created for the sea level data. The main parameters are:

Source name / source name override:
This indicates the name of the parameter within the source. In other words, how Fluidit locates the given variables from the FMI's database. This code can be usually found from the data provider's documentations. In Fluidit Data Source component, right-click can also query the tags/codes that are available for the given source.

Time Offset, expression, and post-processing expression:
These are parameters that are used to process the data coming from the source. In the case of the sea level data, the value is divided by 1000 to convert the units from millimetres (unit provided by the source) to meters (unit needed by the model).

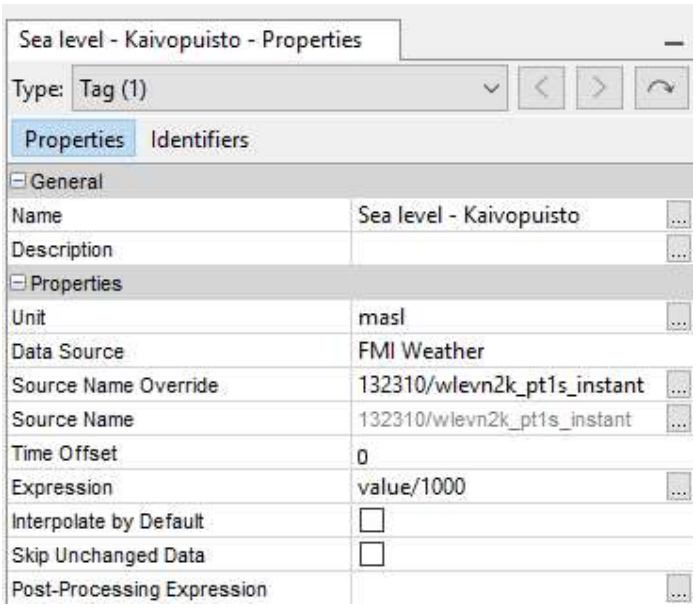


Figure 4. Example of data source tag

General tag statistics:

- 692 tags total
  - Of which 264 are for rain gages – Neuroflux
  - 27 for external inflows - Neuroflux
  - 2 from FMI sources – sea level and temperature
  - 399 tags for simulated CSO load results (description below)

During the pilot phase the model reported results for 57 locations. There were 7 different results for each location. Hence, 7 tags were created for each one of the 57 locations (57 x 7 = 399 tags). Only 3 of the 57 locations were from pumping stations (JVP1014, JVP1052, and JVP1053) and the remaining for CSO locations (YVK). An example of the seven results reported for a location is depicted in Figure 5. These results are sent to Neuroflux as indicated in the process diagram (Figure 3).

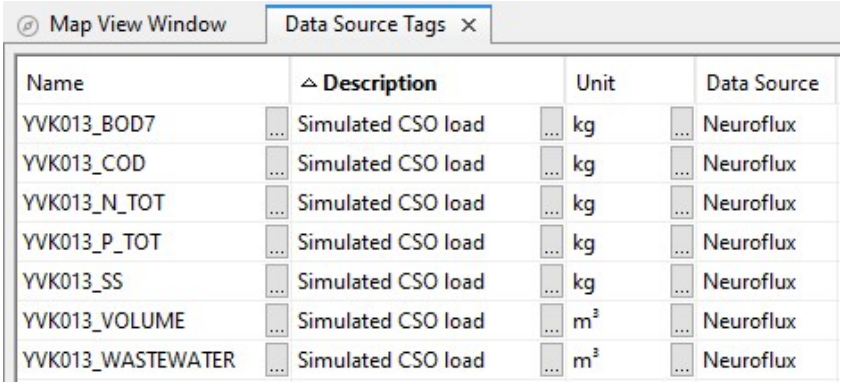


Figure 5. Results reported for each location

Note that the number of results reported (399) are solely determined by the needs of HSY. After a model's simulation is over, hundreds of results available for every location modelled. Thus, the number of results reported can be increased/decreased as per need.

Some of the data fetched from the sources need to be treated before being used as an input to the model. The data processing can range from a simple unit conversion to more complex treatments.

3.7 Data pre- and post-processing

All external inflows, which usually come from Pumping station data, have their unit converted from m³/h to l/s as indicated in Figure 6.

Map View Window | Data Source Tags

Name	Description	Unit	Data Source	Source Name Override	Source Name	Expression
Sea level - Kaivopuisto		masl	FMI Weather	132310/wlevn2k_pt1s_instant	132310/wlevn2k_pt1s_instant	value/1000
JVP1012.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1012.FI1_Q	value/3.6
JVP1023.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1023.FI1_Q	value/3.6
JVP1027.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1027.FI1_Q	value/3.6
JVP1037.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1037.FI1_Q	value/3.6
JVP1041.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1041.FI1_Q	value/3.6
JVP1045.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1045.FI1_Q	value/3.6
JVP1051.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1051.FI1_Q	value/3.6
JVP1057.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1057.FI1_Q	value/3.6
JVP1065.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1065.FI1_Q	value/3.6
JVP1127.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1127.FI1_Q	value/3.6
JVP1132.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1132.FI1_Q	value/3.6
JVP1137.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1137.FI1_Q	value/3.6
JVP1141.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1141.FI1_Q	value/3.6
JVP1148.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1148.FI1_Q	value/3.6
JVP1149.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1149.FI1_Q	value/3.6
JVP1160.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1160.FI1_Q	value/3.6
JVP1172.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1172.FI1_Q	value/3.6
JVP1175.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1175.FI1_Q	value/3.6
JVP1175.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1175.FI1_Q	value/3.6
JVP1176.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1176.FI1_Q	value/3.6
JVP1205.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	JVP1205.FI1_Q	value/3.6
MA1210.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	MA1210.FI1_Q	value/3.6
MA1226.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	MA1226.FI1_Q	value/3.6
MA3104.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	MA3104.FI1_Q	value/3.6
MA3105.FI1_Q	Incoming unit is m³/h	l/s	Neuroflux	<null value>	MA3105.FI1_Q	value/3.6
PITKAKOSKI1	One of the Pitkäkoski treatment plant measurements. Incoming in m³/h.	l/s	Neuroflux	7FI4058:av-1H	7FI4058:av-1H	value/3.6
PITKAKOSKI2	One of the Pitkäkoski treatment plant measurements. Incoming in m³/h.	l/s	Neuroflux	7FI4059:av-1H	7FI4059:av-1H	value/3.6

Figure 6. Data source tags with unit conversion

Finally, the data source tags are assigned to time series components, which can then be used by components in the model such as inflows or rain gages. However, before being used as an input to the model, some data are post-processed as an attempt to avoid issues caused by the faulty data. Time series post processing is done using expressions that are written to the series' property *Data Post Processing* as shown in Figure 7. To access the properties of a given time series through Fluidit's software UI, go to Model > Time Series > select a time series.

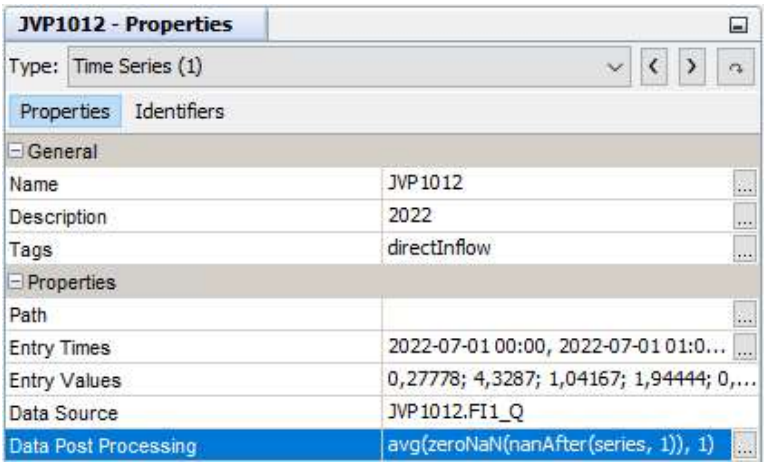


Figure 7. Time series with data source and post processing expression

3.8 Detailed description of the post-processing expressions used

Post-processing used for precipitation data

Precipitation data received a post processing expression to ensure that all steps have data. If the following step has missing data, it repeats the last available value. In case the next data-pairs are also missing, then the values are assumed as zero. See below a detailed description of each function used and a visual representation in Figure 8.

*capMin(lag(zeroNaN(nanAfter(**series**, 10 / 60),), 5 / 60), 0)*

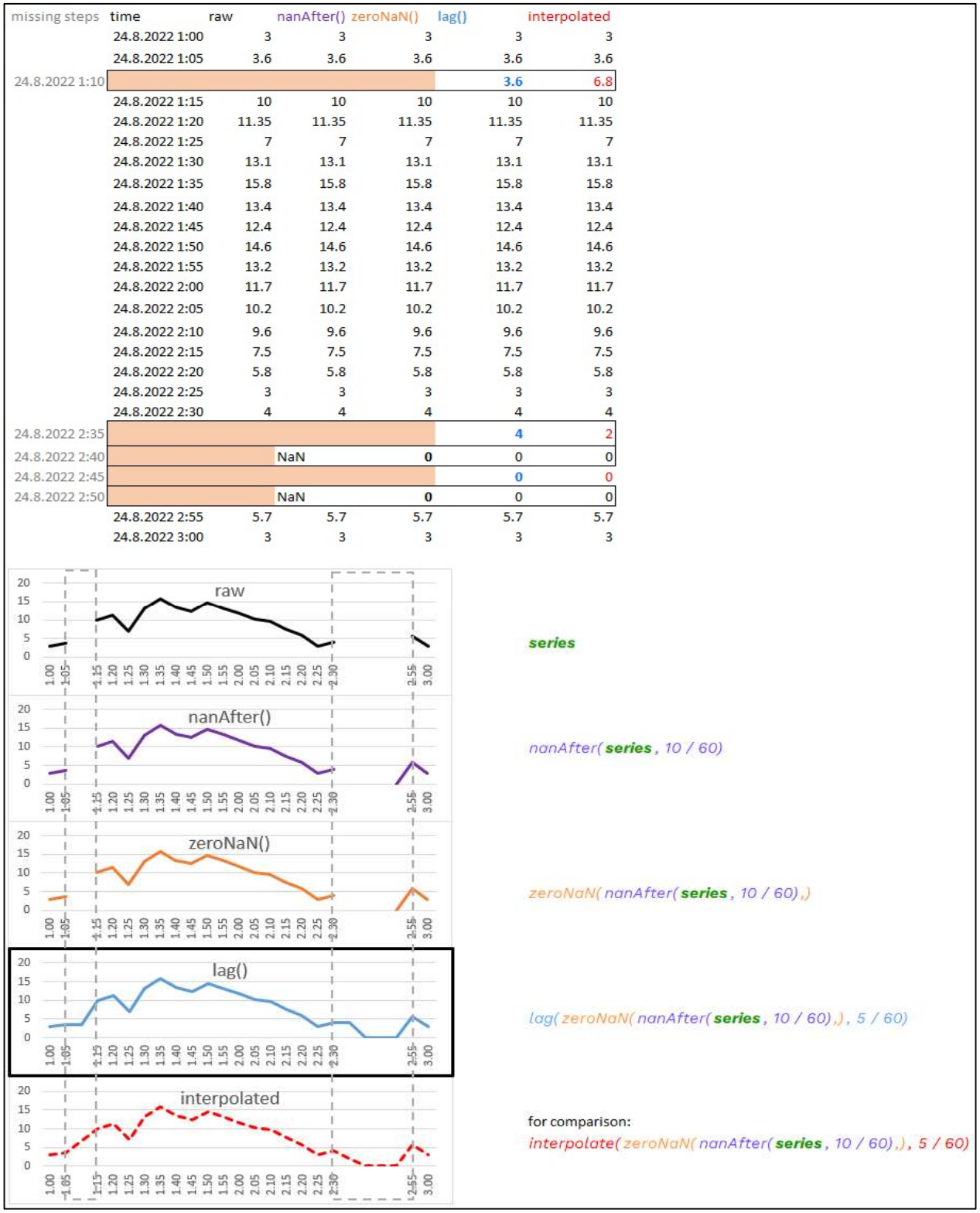
series
Refers to the original raw time series being processed by the expression

nanAfter() *nanAfter(timeseries, 12fterhours)*
Keeps returning NaN for the series, as long as the gap between entries is larger than *afterhours*. In our case this function simply “adds” a NaN (not a number) when there is a gap in the source time series larger than 10min (i.e. sixth of an hour -> 10/60 as this function uses hours as time units). The 10min was chosen because the precipitation data had a 5min step. Hence this assures that there will be a value (NaN) for at least every 10min.

zeroNaN() *zeroNaN(timeseries)*
Replaces all NaN values from the series with zero. In here, this simply replaces all the NaN values that were existing in the source data or generated by the nanAfter() function to zero.

lag() *lag(timeseries)*
Returns lagged values from the raw time series. This function guarantees there will be a data entry every 5min. Also, it copies (lags) the last existing value for the next missing step.

capMin() *capMin(timeseries, min)*
Limits the value to min. In this case, it sets the minimum value possible for the precipitation data to be zero so that negative values (if any) coming from the data source are replaced by zero.



missing steps	time	raw	nanAfter()	zeroNaN()	lag()	interpolated
	24.8.2022 1:00	3	3	3	3	3
	24.8.2022 1:05	3.6	3.6	3.6	3.6	3.6
24.8.2022 1:10					3.6	6.8
	24.8.2022 1:15	10	10	10	10	10
	24.8.2022 1:20	11.35	11.35	11.35	11.35	11.35
	24.8.2022 1:25	7	7	7	7	7
	24.8.2022 1:30	13.1	13.1	13.1	13.1	13.1
	24.8.2022 1:35	15.8	15.8	15.8	15.8	15.8
	24.8.2022 1:40	13.4	13.4	13.4	13.4	13.4
	24.8.2022 1:45	12.4	12.4	12.4	12.4	12.4
	24.8.2022 1:50	14.6	14.6	14.6	14.6	14.6
	24.8.2022 1:55	13.2	13.2	13.2	13.2	13.2
	24.8.2022 2:00	11.7	11.7	11.7	11.7	11.7
	24.8.2022 2:05	10.2	10.2	10.2	10.2	10.2
	24.8.2022 2:10	9.6	9.6	9.6	9.6	9.6
	24.8.2022 2:15	7.5	7.5	7.5	7.5	7.5
	24.8.2022 2:20	5.8	5.8	5.8	5.8	5.8
	24.8.2022 2:25	3	3	3	3	3
	24.8.2022 2:30	4	4	4	4	4
24.8.2022 2:35					4	2
24.8.2022 2:40			NaN	0	0	0
24.8.2022 2:45					0	0
24.8.2022 2:50			NaN	0	0	0
	24.8.2022 2:55	5.7	5.7	5.7	5.7	5.7
	24.8.2022 3:00	3	3	3	3	3

Figure 8. Visual representation of a post processing expression

Post-processing for pumping station data

Pumping station data also received a post processing expression. In this case, it guarantees there will be data after a 1h gap and the value will be the average between the 1h step. As it uses almost the same functions as the expression used for precipitation data, refer to its documentation above for more details.

*avg(zeroNaN(nanAfter(**series**, 1)), 1)*

There are several possibilities of data processing available in Fluidit software. The examples above were utilized during the pilot phase. More processing can be done in future stages of the project per need. Contact support@fluidit.com for more details.

4 TESTS AND RESULTS

This section describes tests carried out during the pilot phase.

4.1 Real-time vs manually generated results

The goal of this test was to assess whether the results generated by the real-time process (every 1h) are similar to results obtained when running the simulation manually. Larger discrepancies here could indicate issues in the data or data transfer. Smaller discrepancies can occur since both methods are not identical as, for instance, pumps' statuses are not retained in the hot start file. However, even smaller discrepancies can be further investigated in this test.

The initial tests showed large discrepancies in overflow quantities with the real-time application overestimating by about 4 times the amount reported by simulations manually generated. After investigations it was observed that the precipitation data of the two methods was the main cause of discrepancies. The two main differences noted were:

1. No-data values: precipitation data input to the real-time model (from Neuroflux) had much larger overall volume as no-data values of 2.55 were kept in the data. This was corrected during the pilot stage. The new results after correction showed a much better agreement in overflow volumes reported when comparing the two methods.

2. Differences observed: there were still differences observed when checking the precipitation data on the rain gage's level. Even when the overall volume of precipitation falling on the entire model's domain is similar, the local variations observed still yield different overflow results since catchments upstream the overflow points will receive slightly different precipitation data. See an example for rain gage 254954036675064 in Figure 9. Investigations regarding the differences are still ongoing with the latest hypothesis being related to how precipitation data is treated by the two distinct processes (Manual vs Neuroflux). Details of the discussions can be found in the Appendix B.

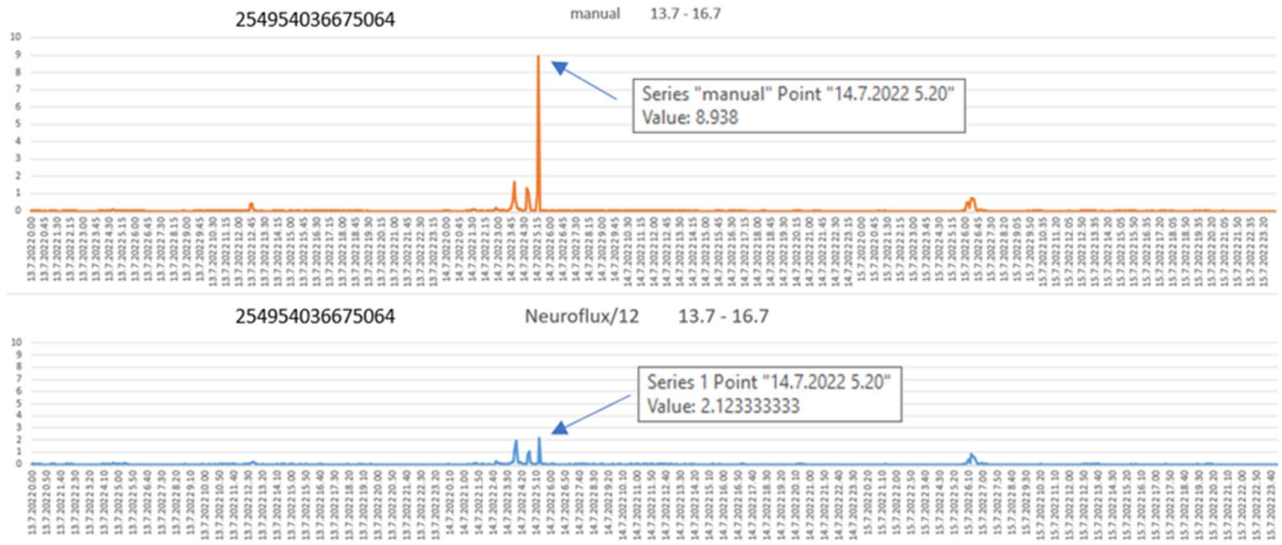


Figure 9. Manual vs Real-time: differences in precipitation input data

4.2 Simulation parameters vs accuracy

The period between 14.7.2022 and 17.7.2022 was chosen for the evaluation as three events occurred within this period. The events are classified here as A, B and C. Four measurements were available (Figure 10), but only YVK008 was evaluated for simplicity.

Overflows during the two first events (A and B) were likely to have occurred as the measured depth at the manhole/tank was above the overflow threshold (see Figure 12). No overflow was observed from the measurements in the third event (C).

Three sets of simulation parameters were used for the comparison. The initial set of parameters are referred as *Optimized* since they represent the set which focus on run time speeds rather than accuracy. The other two sets: *Accurate* and *Extra* used refined parameters to increase the model's results' accuracy. The set of parameters are depicted in Figure 11.

The results of each set are depicted in Figure 12. The *Accurate* set did not present any relevant improvement in relation to the previous set (*Optimized*). The *Extra* set simulated the initial peak of event A with more agreement with the measurements and presented a better match for event B. All sets overestimated the measurements for event C. All simulations seem to overestimate the volume in the system at this location during the three events.

Figure 10. Depth measurements available within the model's domain

Accurate Settings changed to increase accuracy. Min. Step, head tolerance, and inertial terms

Extra Further increase as lengthening step is removed, inertial terms kept, min step and head tolerance reduced

General	
Name	HSY
Description	Combined sewer model
Coordinate Reference System (C	ETRS89 / GK25FIN (EPSG:387...
Time	
Simulation Start Time	1.1.2022 0:00
Simulation End Time	2.1.2022 1:00
Report Results Start	1.1.2022 0:00
Report Step	01:00:00
Dry Step	02:00:00
Wet Step	00:05:00
Lengthening Step	5 → 0
Minimum Step	0,5 → 0,1 → 0,001
Routing Step	15
Variable Step	0,75
Report Averages	☑
Time Zone	Europe/Helsinki
Properties	
Background Color	☐ [255,255,255]
Units	l/s
Headloss Formula	Darcy–Weisbach
Dry Days	0
Sweep Start	Jan 1
Sweep End	Dec 31
Link Offsets	Elevation
Minimum Surface Area	0,503

Solver	
Flow Routing	Dynamic wave
Head Tolerance	0,01 → 0,0015 → 0,001
Allow Ponding	☑
Surcharge Model	Preissmann Slot Method
Infiltration Model	Green ampt
Ignore Groundwater	☐
Ignore Quality	☐
Ignore Rainfall	☐
Ignore RDII	☐
Ignore Routing	☐
Ignore Snow Melt	☐
Inertial Terms	Ignore → dampen → keep
Maximum Trials	16
Minimum Slope	0,001
Skip Steady State	☐
Lateral Flow Tolerance	5
System Flow Tolerance	5
Supercritical Flow	Slope + Froude number
Threads	12
Energy	
Default Pump Efficiency	70
Default Motor Efficiency	85
Default VSD Efficiency	95
Fluid	
Relative Specific Gravity	1

Figure 11. Previous optimized model's properties. Also used during most of the pilot phase. New settings (accurate and extra) used to evaluate model's performance.

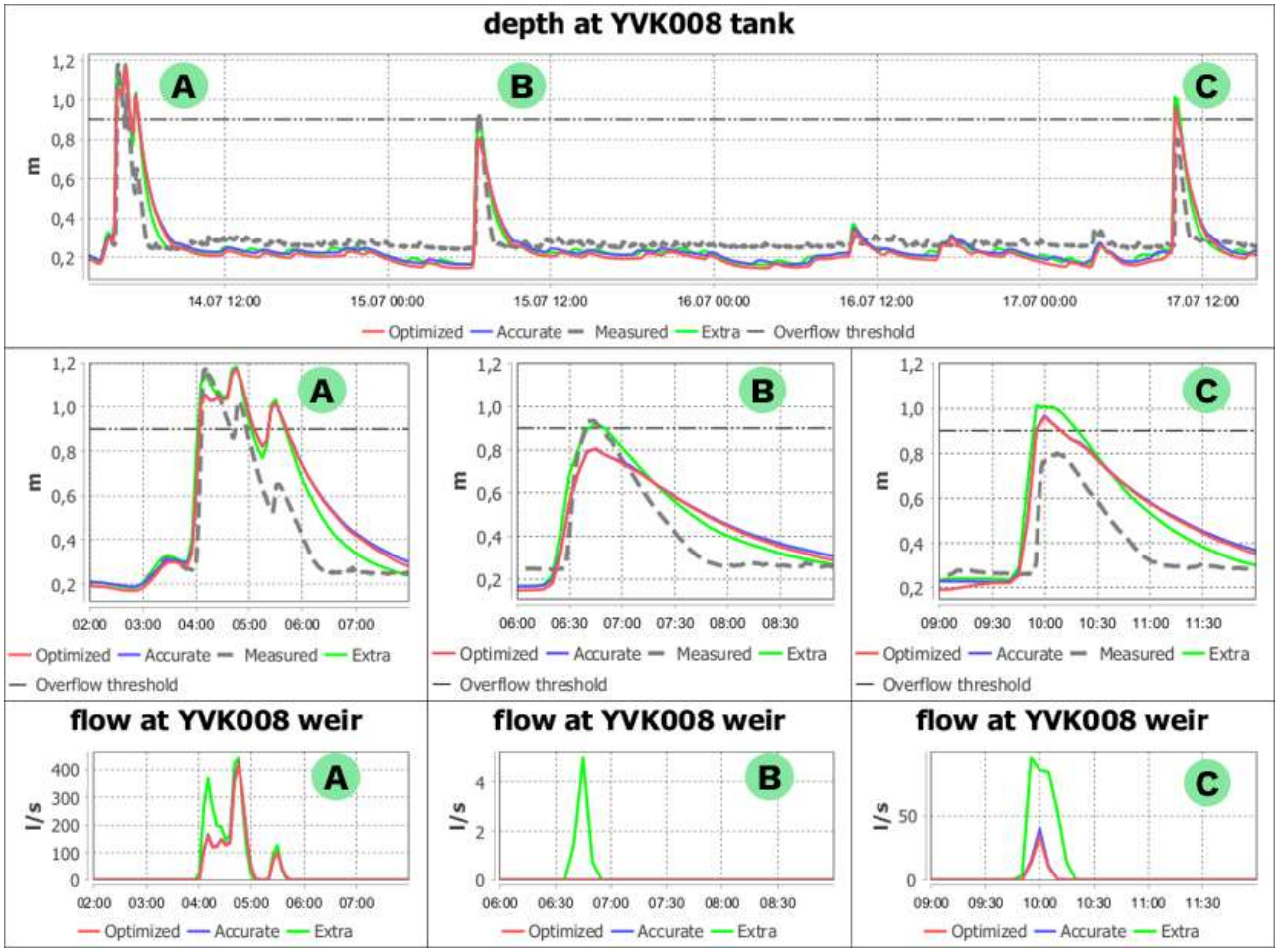


Figure 12. Model results for YVK008 with different model settings: Optimized, Accurate, and Extra

4.3 Run times

Goal of this test was to measure the time it takes for the model to run using the different simulation settings as described in the previous section and document the time delays to acquire input data for a simulation.

HSY-CS-model has been used to simulate 3-month long simulations among other applications. Hence, the simulation's settings were optimized to ensure reasonable run times considering the size of the model and long period of simulations required. However, for the real-time framework, the settings could be adjusted to focus on accuracy, provided that the run times will not increase over the limit required for the application. After testing it was observed that the total run time (acquire the data + simulate) with more accurate settings (*Extra*) is about 3min for 1h long simulation period. Hence, the most accurate setting tested can be used for the application with the current model's version. Run times for other periods are available in Table 1. smaller simulation periods would also be possible (30min, 10min, and 5min) with the current version of the model.

Table 1. Run times with different settings

Period's length	**Delay to** fetch and load **input data** (Neuroflux)	**Model's run time optimized settings**	**Model's run time with extra accurate settings**
5min	00:00:47	00:00:05	00:00:14
10min	00:00:47	00:00:08	00:00:21
30min	00:00:47	00:00:14	00:00:52
1h	00:00:48	00:00:23	00:02:00
3h	00:00:50	00:01:23	00:06:00 *
24h	00:00:54	00:07:06	00:48:00 *
7d	00:01:40	01:10:30	05:36:00 *
31 days (1m)	00:04:21	04:45:12 *	24:48:00 *
92 days (3m)	00:10:05	13:24:12	73:36:00 *

** Estimated based on 1h simulation period of the same settings*
Notes:

- ***tests were carried operating the model via Fluidit's software** user interface. When the software runs via command line the processes are faster since no UI processing is required (the case for the real-time application). Hence, the times above can be considered conservative.*
- *Tests were carried in the test environment. Specification of the test environment can be found in section 3.2.*
- *Simulations had initial conditions set by a hotstart file.*
- *Extra accurate settings are defined* **Figure 11**.
- *The model version used had only 1D components. No 2D flood simulations were evaluated.*

5 CHALLENGES AND PROPOSED TIMELINE

5.1 Other challenges and enhancements

Challenges and enhancements during the pilot stage are listed here:

- Flow oscillations in long-term simulations. Details in appendix C
- Fluidit's SWMM improved how pollutants are calculated at links after discrepancies pointed out by Leena Sänkiaho (HSY).
- Fluidit implemented hotstart files to better account for initial conditions.
- Fluidit implemented Neuroflux data source drive. Described in section 3.5.
- Input data required some processing, which is done directly in Fluidit software. Details already described in section 3.7. Further processing is suggested in 6.
- Neuroflux/HSY also has depth data available for various locations within the model's domain which could be used as initial condition (overriding, for these locations, values obtained from the hotstart file). However, these can cause instabilities as the values may not agree with initial conditions of the neighbouring elements in the network. Thus, no initial level/data was used during the pilot state to override those coming from the hotstart file. If necessary, this can be implemented/reassessed during the next stages of the project.
- Flow oscillations in long-term simulations: details in appendix C
- Different versions of the model. This is discussed in the following section.

5.2 Suggested version control framework

It is essential to store the model more consistently to avoid having multiple versions of the model stored separately with little metadata and to make it possible to apply updates to the model. This section proposes a framework for storing the model leveraging Fluidit's version control capabilities (a built-in GIT implementation). The framework is illustrated in Figure 13. The need for such framework was observed also during this project.

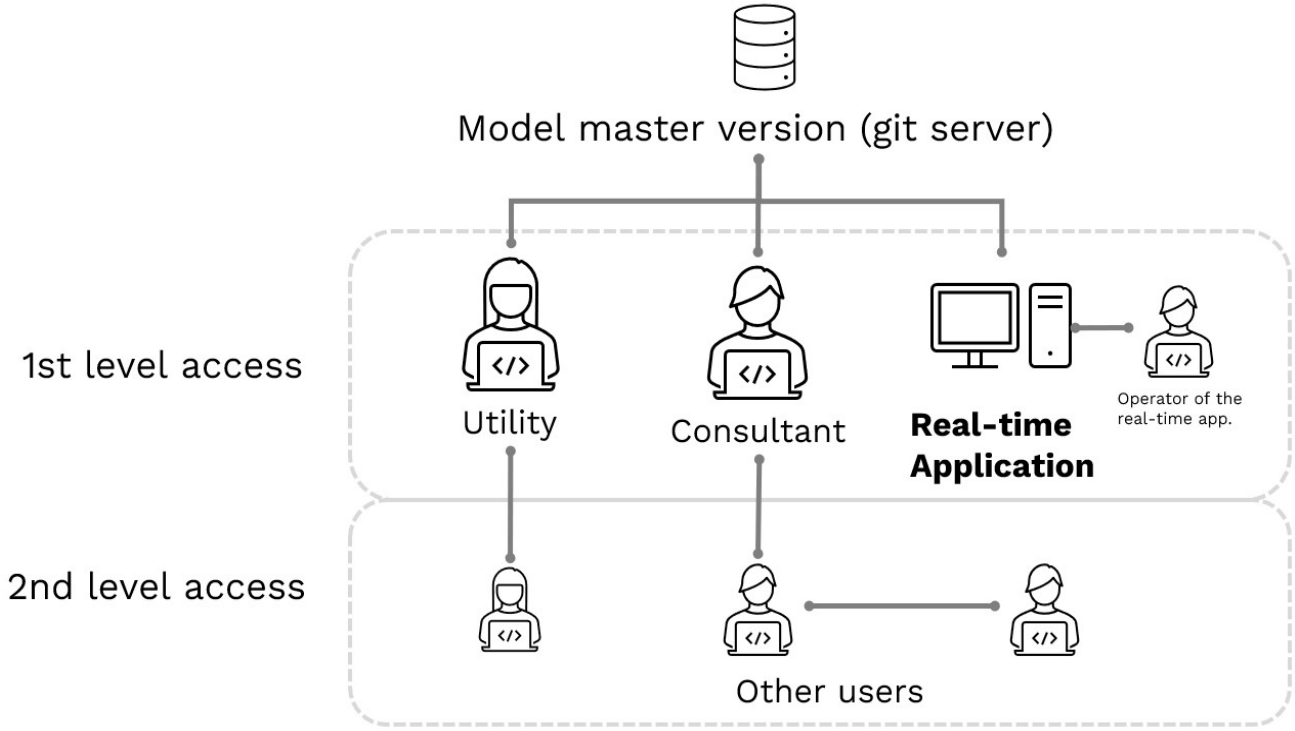


Figure 13. Version control framework

Model master version
Main version of the model. This version should be a working stable version of the model that all the users can rely on. In other words, it is the version of the model containing the most updated or reliable information of the real network/system. To leverage the existing feature of working collaboratively in Fluidit Storm, the model should be stored on a GIT server. The server can be a local server of the utility (i.e. not in the cloud) and can be managed fully by the utility's IT department in order to apply all the security protocols desired, such as management of access. Since GIT version-control system is widely used, it is likely that utilities (or their IT departments) are already familiar with it and even have already their own GIT server(s). If more assistance is required, please contact support@fluidit.com.

1st level access
These are the users/applications with access to the model master version. By default in Fluidit Storm's user interface, the latest version of the master model is downloaded to the user's computer whenever the model is opened, if the user provides his/her credentials for the server. Once permanent updates are done in the model by the user (e.g. deleting a pipe that has been discontinued) and the model is saved, these updates are sent to the master version of the model which will be available for other users with 1st level access - including the real-time

application discussed in this report. Examples of tasks that would be carried by a user with 1st level access are:

- Updating a pipe in the model
- Adding network of new neighbourhood
- Updating the inflows (e.g. adding new wastewater consumers)
- Updating parameters of a catchment
- Updating settings of pumping station to better match measurements
- Creating new visualizations that can be reused by other users or recurrent processes (drawing states)
- Creating new reports that can be reused by other users or recurrent processes (schematics)
- Calibrating groundwater infiltration with new available measurement data

Every time a user updates the model, the user may write a brief description (commit) describing the alterations. The user's credentials, date and time of the updates, and model version is stored. These versions can be viewed and managed through Storm's user interface (Figure 14). It's possible also to visually inspect previous versions (check out) and, if necessary, revert to a previous version. Reverting, for instance, could happen when the user realizes the latest updates (commits) contain errors and a previous version should be the latest (head).

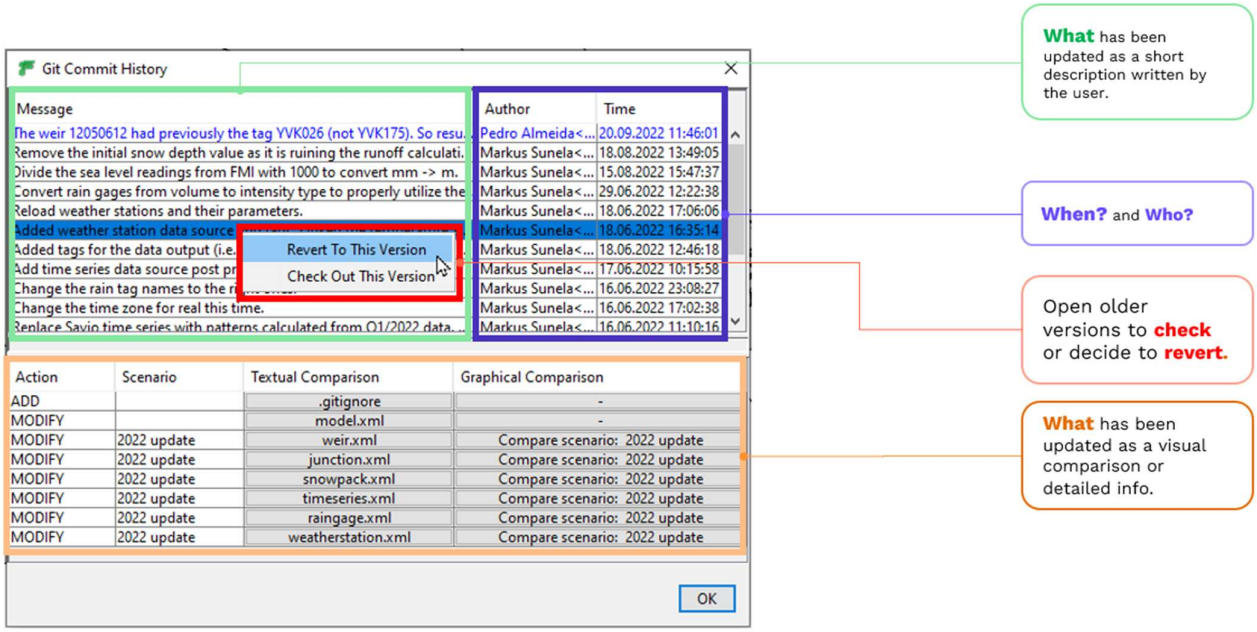


Figure 14. Git history tool in Fluidit

2nd level access

Users with 2nd level access receive the model only from users that have 1st level access. Thus, there is no direct connection between a user with second level access and the master version of the model. Users with 1st level access can share a snapshot version of the model to users with 2nd level access in case they have to perform some tasks using the model. The model can be shared via a simple file. External consultants are an example of users that could have 2nd level access. Examples of tasks that would be carried by a user with 2nd level access are:

- Temporary analysis using a snapshot version of the model
- New modellers or trainees getting familiar with the model
- Running simulations for a specific report that do not require any update to the model

For more on Fluidit's version-control, check our support page:
https://support.fluidit.com/projects/simulators/wiki/Saving_and_opening_the_model

5.3 Proposed timeline to achieve early-warning system

The hindcast/nowcast in real-time and the infrastructure created to support such system allows for a wide range of applications. Many of these applications could greatly benefit the utility's operation and the society. Figure 15 below shows a suggestion for a timeline to achieve an early-warning system. It is important to highlight that the next steps can be achieved with less effort since they leverage the same/similar infrastructure created and tested during this pilot stage.

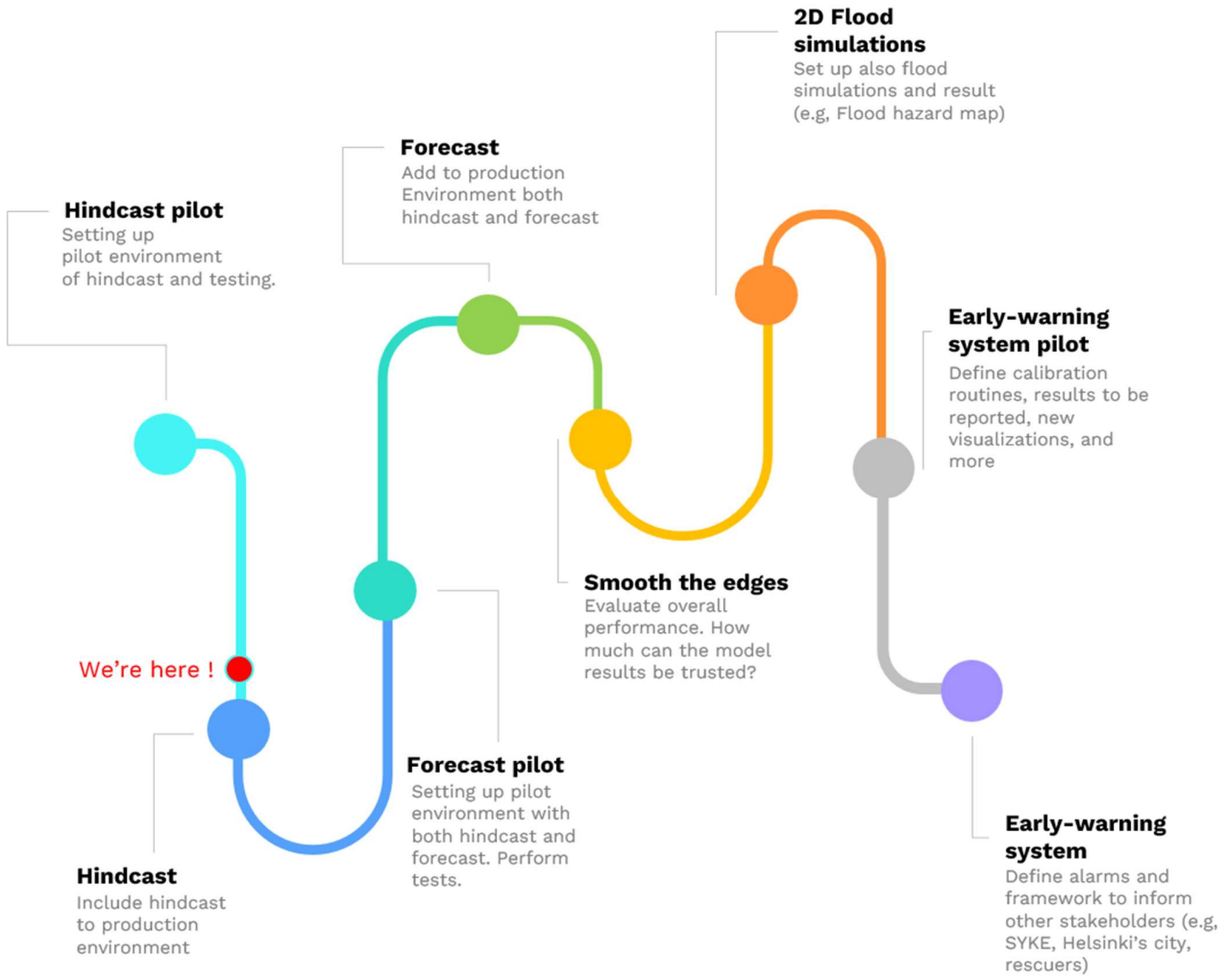


Figure 15. Suggested timeline to achieve an early-warning system

6 CONCLUSIONS AND LIST OF TASKS

This section lists the tasks suggested for the next stages of the real-time application. The list is sorted by the suggested sequence the tasks should be carried when moving forward in the timeline proposed in Figure 15 above. It is recommended that the 12 topics discussed in section 6.1 are implemented already while setting up the production environment.

6.1 Tasks to achieve production environment

6.1.1 Reducing the manual work for Q3 simulations and report

Once the differences in precipitation are sorted out (described in Section 5.1) the accumulated 3 months long results can be compared between the previous manual method and Neuroflux's accumulated results. After this check, the amount of manual work to prepare the model for a Q3 simulations can be immediately reduced since all input data can be fetched with a few clicks [Tools] > [Update Time Series Data from Source].

6.1.2 Updating strategies to produce automated reports

The reports that are obtained from the manual method (via the Python code *generateReport.py* which is included in the model) and the reports shown in Neuroflux are likely different in content and layout. HSY and the other parties should decide if the reports will be still generated by the Python plugin available in the model or if the reports will be compiled by Neuroflux. Once this decision about the reports is made, the manual work can be almost completely deprecated as all the process can be automated leveraging the real-time system. Additionally, the amount/location of model's results transmitted to Neuroflux (section 3.6) can be easily increased/decreased if necessary. The model produces several results for every location of the modelled network.

6.1.3 Creating data tags for measurement data

HSY has measurement data collected in different parts of the network. A quick access to this data would likely benefit all users of the model in all its applications since data would be easily fetched directly from Fluidit Storm's user interface with few clicks. It also supports further calibration or performance evaluation strategies for the real-time system.

6.1.4 Implement a version control strategy

Implementation of the version control framework is proposed in 5.2.

6.1.5 Metadata for simulation results

It was discussed during this pilot the need to connect the model's version to the results generated by the simulation. This information informs the users which results were generated by which version of the model to assist in future reviews should any error be encountered in the model. It was suggested during this project that the simulations will never be rerun to avoid confusion (e.g. old reports lose their value as the past results would

be modified). If an update is required in the model to ensure more realistic results, the update is carried and the next results from that moment will be generated with the correct version of the model. The version control framework proposed in 5.2 precedes this implementation. The metadata indicating the version that generated the results can be easily stored (and sent) to other systems, for example stored as a result in Neuroflux.

6.1.6 Updating the model version used in the real-time system

The model version used by the real-time system can be updated by users with first level access and access to the environment. Updates are likely to be done manually (or semi manually) to ensure that the version chosen is suitable to the purpose of the system. The utility can decide when updates are necessary based on changes done to the master version (e.g. when it is known that the version used by the system is outdated).

6.1.7 Cleaning up the files

With the current framework (Figure 3) new model and a hotstart file are created for every run (two files every 1h) which were about 17 MB together (8MB for results, 8 MB for the model itself, and 1 MB hotstart file). During the pilot stage these files were kept in order for quick debugging. However, keeping these files is important as they work as input for the next simulations and can also be used to investigate recent interesting events without having to resimulate. It should be decided, during the next stage, how many files can/should be stored based on the production's environment storage space limits. For instance, if only the files generated in the latest week were kept, and the model generates results for 2h10min period (10min hindcast and 2h forecast) with 10min report step and simulation frequency, each simulation would generate ~ 37 MB. For a week, this would mean ~ 37 GB worth of files. The old files are currently automatically cleaned up by a separate Python script run once a day.

6.1.8 Server's technical specifications for production environment

The main driving factors defining the recommended specifications are the model and its data, Fluidit Storm and peripheral software, and the desired level of availability for the server. Cyber security is discussed instead in section 6.1.9 below.

Table 2. Server's technical specifications for production environment

Operating system	Preferably Linux (e.g. Ubuntu 20.04, CentOS 9) or Windows Server 2019 or 2022. 64-bit operating system is required
Processor	Minimum a dual-core CPU (e.g., Intel i5 or AMD A10); at least four cores (eight recommended) for running Fluidit Viewer server or 2D flood simulations
RAM	8GB RAM; 16 GB or more, if running Fluidit Viewer server or 2D flood simulations
Disk space	At least 256 GB
GPU	For 2D flood simulations, a powerful OpenCL >=2.0 capable, discrete GPU with at least 4 GB of dedicated memory along with the proper drivers is required. Preferably a Nvidia GPU

	(e.g. GeForce 16xx or 20xx series). AMD GPU's from Radeon RX 500 and RX 5000 series work too.
Outbound connections	The software can function without Internet connection, but there likely is need for at least limited connections. Currently the temperature and sea level data are fetched from FMI servers (https://opendata.fmi.fi/), and HSY measurement data is fetched from and simulation results are stored in Neuroflux via (https://hsy.nfapp.fi:5000), and if no changes are to be done, these connections must be enabled. Ideally there would also be possibility for communicating with the Fluidit license server at (https://license.fluidit.fi). If Fluidit Viewer server is installed, then Internet connection is also required to any background map sources, such as https://kartat.kapsi.fi for the Finnish national land survey maps and OpenStreetMap https://[abc].tile.openstreetmap.org/.
Inbound connections	If Fluidit Viewer server is installed, then the server machine and related firewalls etc. must accept HTTP(S)-connections from the HSY intranet (office network).
Software required on server	Docker (Linux) for running Fluidit Storm Online Docker (Linux) for running Fluidit Viewer, if enabled
Availability	24/7 The system will be producing data that is likely going to be used in some extent by HSY's operational team on daily basis. Some details are also discussed in section 6.1.9.
License	The service requires a specific license for Fluidit Storm Online software which will be granted by Fluidit Ltd. The license file and product key are stored on the server. If Internet connection won't be available, a perpetual license tied to the specific machine will be used. The license will grant right to run the online simulation (and Fluidit Viewer) on the server.
Permissions to users outside HSY's domain	True remote connection (SSH, RDP) is required to the server. Access to the related services, such as Neuroflux and Fluidit Viewer, via VPN is required, too. Ideally the Fluidit support team would have root-level access to the server.
Software distribution	The real-time simulations require no software on the end user workstations. The results are accessible via web-browser via Neuroflux and the optional Fluidit Viewer web interfaces. Desktop version Fluidit Storm can be installed on interested users' workstations. On Windows, the software should be distributed using Endpoint Management (Software Center), via the installer provided by Fluidit Ltd. The cryptographically signed installer bundles Java with the software and supports silent installs.

6.1.9 Server resilience and cyber security

As the real-time application progresses and starts to be used by the operational team of the utility, it is recommended that at least some level of resilience for the production environment (server) is implemented. The resilience to be discussed may include backup in case of power outage, alerts of communication failure or speed, security, and other overall status alerts.

Funding opportunity is available from the Finnish Transport and Communications Agency - Traficom (Liikenne- ja viestintävirasto). The grant is aimed to support projects that increase the cyber security of companies critical to society's functioning. It is very likely that the cyber security around the model's data and all its applications are eligible for the grant. A project to improve the cyber security can run in parallel with other activities/projects done

with the model. Fluidit's IT experts can also work together with HSY's IT department to define security specifications for the model's applications. The application is open since 1st of December 2022. Traficom will organize a webinars to instruct how to apply for the grant. First webinar happens on January 11, 2023 from 9 to 11 a.m. More information (in Finnish) available here:
https://www.traficom.fi/fi/ajankohtaista/tietoturvasetelin-haku-aukeaa-pian-tutustu-tietoturvan-kehittamisen-tuen-ehtoihin-ja

6.1.10 New simulation's frequency for the production environment

As discussed in 4.2 and 4.3, the simulations can run more frequently and use more accurate settings. This can be already implemented when the hindcast goes for the production environment, but the frequency adopted will also likely continue even for further steps of the system. For instance, for a 2h10min simulation period (10min hindcast and simulation frequency + 2h forecasted results) as idealized in Figure 16, the entire process is likely to take ~ 8min. So, even if longer periods are simulated, a 10min frequency would be possible or 20min for a more conservative approach. The more accurate properties can be set to the model using the python script at the beginning of each simulation to ensure that the master version of the model keeps the more optimized settings.

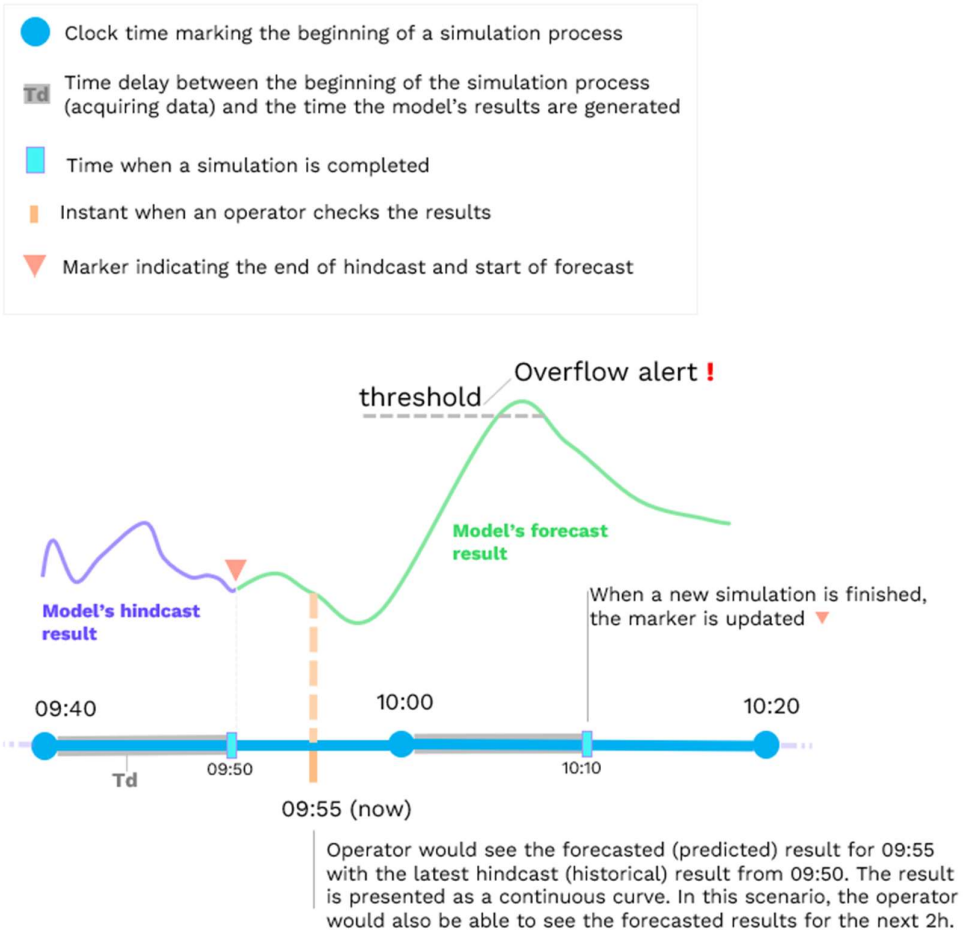


Figure 16. Operator's perspective when visualizing results from the model once the forecast is introduced. The scenario assumes simulations every 10min with 2h forecasted values.

6.1.11 More input data processing

Simple data processing was introduced during this project as described in 3.7 and 3.8. More opportunities for further processing were identified during this project. The processing of the data used as input for the model can be done in Neuroflux so that the data fetched by the model will already be processed or done directly in the model via Fluidit Storm's user interface as described here in this document. The parties should decide during the next stages.

- Cap values for all input data to remove extreme/negative values when applied
- Replacement strategy when data is not available for some rain gage or pumping station external inflows
- Identification and replacement of outliers which indicate erroneous measurement
- Identification of discontinuity in direct inflows that can cause instabilities in the model and replacement strategy when applicable.
- Logging erroneous data to be reported to the data providers.
- Discuss and identify other data treatment necessary and add to this list.

6.1.12 Continuous automated checks and status report

The checks listed below can be implemented to ensure that the model is simulating reasonably, and system's responsible and operational team are alerted when the simulation results are not in the acceptable range. First it is necessary to decide how alerts/status reports should be delivered (e.g. by e-mail or logged to a file available for all users).

Proposed alerts include:

- Errors acquiring input data
- Expected time delay to complete a simulation is exceeded
- Issues with the server (HSY's IT department)

Proposed Information to be logged:

- SWMM's simulation reports (also related to section 6.1.7)
- List/GIS data for locations with less accurate results (e.g. list of nodes with high instability index).
- Instabilities in the main model's results (such as flow oscillations discussed in the appendix C)
- Unrealistic results, such as overflows during dry periods.
- Recent and past model performance (discussed in more details in section 6.2.3)

6.2 Discussions for the future of the system

6.2.1 Forecast input data

When simulated forecasts are to be implemented, the forecast input data should be chosen. The most likely provider is the Finnish Meteorological Institute (FMI). All the parties involved in this project can work together with FMI to decide on the best data available.

Since Finland is an EU member country, it is also likely that the forecasts produced by the ECMWF are available for free (which can likely also be distributed via FMI's APIs).

6.2.2 Test how 2D simulations can be applied to the real-time system

Flood maps can be generated using Fluidit Storm so that the real-time system would also be able to provide such maps and other details related to flood risks. For that, it is necessary to access how the 2D simulations would impact the simulation time and how the optimum strategy for HSY's model.

6.2.3 Continuous and automated performance evaluation

Measurement data is compared with simulated results to evaluate the model's performance. This task is usually carried out during calibration projects. The real-time system can be leveraged to reduce the manual work and increase seasonality and knowledge about shortcomings in the model. This can be achieved via automated performance evaluation (comparison measured vs. simulated values) with defined frequency. Such framework allows the location of uncertainties in the model that helps the operational team to take more informed decisions based on the model's results. One of the frameworks of performance evaluation and user-friendly visualization is discussed in this study:
https://www.researchgate.net/publication/358634722_Using_multi-event_hydrologic_and_hydraulic_signatures_from_water_level_sensors_to_diagnose_locations_of_uncertainty_in_integrated_urban_drainage_models_used_in_living_digital_twins

6.2.4 Including probabilities to the results

Probabilities added to results can lead to much better estimations, especially when setting up warnings (e.g. overflow alarms). There is an interesting game prepared by research carried by the University of Newcastle (UK) to illustrate, in very practical way, how probabilistic results are usually much better than deterministic (current approach in the real-time process). The game is in a spreadsheet. Both the game and reference can be found here: https://research.ncl.ac.uk/vcurbanflood/exploreourproject/virtualflood/