Deliverable

D8.6 Report on Harmonized Platform for Operational Earthquake Forecast – demonstrator for ensemble models

[WP8 - *Demonstrator*]

Deliverable information	
Work package	D8.6 Harmonized platform for OEF forecasts and ensemble models [demonstrator]
Lead	Eth Zurich
Authors	Laura Sarson, ETH Zurich Nicolas Schmid, ETH Zurich Leila Mizrahi, ETH Zurich Irina Dallo, ETH Zurich
Reviewers	Laurentiu Danciu, ETH Zurich
Approval	[Management Board]
Status	[Draft]
Dissemination level	[Public/internal]
Delivery deadline	31.08.2023
Submission date	23.05.2023
Intranet path	DOCUMENTS/DELIVERABLES/ Deliverable 8.6

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821115

Table of Contents

Summary

This deliverable provides an overview of the harmonized platform for Operational Earthquake Forecasting (OEF), with application to natural earthquake forecasts and ensemble models.

The enhanced software RT-RAMSIS (Real Time Risk Assessment and Mitigation for Induced Seismicity) is the core of the system developed within the RISE project. Leveraging the diverse range of forecast models created in Work Package 3, with RT-RAMSIS allows to establish a modular and adaptable framework. This framework will facilitate the seamless integration of one or multiple OEF codes for a specific region or country, enabling the construction of dynamically weighted ensemble models (as outlined in WP3). The results will be effectively displayed and broadcasted, either in a web-platform or stand-alone solutions such as Jupyter notebooks.

In this report we summarize the main components of the software, platform as well as the technical aspects of the platforms. It shall be emphasized that the platform is a demonstrator and it can be setup to become operational given the availability of a seismic network. Overall, such tool can host the future earthquake forecasting models at regional, or local scale if future models will be available. Also, the earthquake forecasting models for Europe could be also linked with the statistical testing services i.e., CSEP community when available. Last but not least, these services will provide valuable insights and lessons for future advancements based on two-way communication as highlighted in the Deliverable D5.1.

1. RT-RAMSIS Overview

RT-RAMSIS software was originally designed to operationally serve the induced seismicity, and within the framework of RISE project it has been extended upon to work with natural seismicity forecast models. Many aspects of the software have been simplified to make setup and running of seismicity forecasts easier and with greater automation. RT-RAMSIS package is designed to run, configure, and orchestrate Seismicity Forecast Models (SFMs). A SFM is a model that takes input data such as an earthquake catalogue, other data (seismogenic sources, active faults) or information (i.e., completeness) that may be incorporated by a model, and produces forecast seismicity rates for selected periods of time. The earthquake rate forecast may be discretized into time and space bins and can optionally come as a stochastic catalogue or as parameters that describe the statistical likelihood of an event happening.

The need for a tool to handle the running of these SFMs came about as many scientists had designed models that had different inputs and result formatting, different needs for running their software and no clear way to gather results for easy comparison and analysis. Furthermore, running these models in an operational way, and in real time also needed to be considered.

The platform aims to solve the challenges encountered when running forecasts in a harmonized way. Below are the main challenges outlined and the way that RT-RAMSIS handles the problem.

Model related challenges:

- **Consolidating results from multiple models.** The data model captures data from multiple models in the same structure and access point. This allows users to easily download results, (as well as configuration and the data that was used in the forecast) and combine results on the same space and time scale.
- **Model access and standardization**. Often there are barriers to running different SFMs, such as thorough understanding of parameters and boundary conditions. With the inclusion of a SFM on the RT-RAMSIS platform, the parameters are standardized and documented

so that other users may more easily run them. The SFM is wrapped in a piece of code that transforms data and callable into what the SFM requires.

The two ways that a user can produce results:

- If a model is hosted operationally, users can run RT-RAMSIS themselves and access the model through its web service
- A user can install the model themselves. The addition of a single-user entry point means that the results will get saved to a local database. This is advantageous as the results will be saved in the same way as within RT-RAMSIS and are comparable.

Operational running challenges:

- **Data reproducibility**. In cases where results need to be verified by different parties or reproduced for analysis, RT-RAMSIS captures configuration at every level. This ensures the results are traceable and referable.
- **Data sharing**. As configuration for forecasts is kept in a version-controlled repository which allows for easy sharing of case studies. The results are stored within a database where the data can be accessed via a web service for public use. This web service can also be used to set up web viewers showing results for different projects.
- **Scheduling**. The platform enables a user to schedule forecasts for any time period with a predefined interval and handles overdue forecast runs. As the platform uses an advanced library for scheduling, this makes the system more dependable and improves system automation.
- **Dynamic model configuration**. Dynamic adding of a SFM configuration to existing forecast runs means that users may easily alter what is run on the pre-defined schedule. Configuration changes are tracked and controlled so that data is never deleted if it has been used, ensuring data persistence.
- **Parallelization**. Some tasks may be done sequentially, while others can be done at the same time. RT-RAMSIS runs forecasts and model runs in parallel so that the tasks can be completed in an optimized time frame.
- **Contingency**. If something goes wrong, it is easy to handle the logic of what should happen next, whether it is to retry later or fail with a certain error.
- **Scalability**. The engine that runs behind RT-RAMSIS in processing the tasks can be scaled up to suit the needs of different projects.

The solutions to these challenges will become clearer as RT-RAMSIS is used more widely.

As stated before, RT-RAMSIS was initially designed for induced seismicity models, however the transition from induced to natural seismicity is not a great leap. The software now handles both cases with a simple set of configuration parameters.

2. RT-RAMSIS Improvements

The main updates and improvements to the platform are discussed.

2.1 Time dependency

Improvements have been made to RT-RAMSIS in the scheduling of forecasts with an open-source solution (https://github.com/PrefectHQ/prefect). *Prefect* is a Python-based orchestrator that simplifies the management of data-intensive workflows. It enables the transformation of any Python function into a work unit that can be monitored and controlled. With capabilities such as automatic retries, distributed execution, and scheduling, *Prefect* makes it simple to create workflows that are resilient to changes and recover from errors. By upgrading the *Prefect* dependency within the software to the latest version, schedules can now be created without requiring an end date to be defined. This means that manual intervention to create more forecasts is not needed, and forecasts will now be created dynamically until the user decides to stop the schedule.

In addition, the ability to run overdue forecasts at a defined interval has also been added. This means that if there are scheduled forecasts for which the schedule data is in the past, they will not all be executed simultaneously. This is an improvement made due to the overloading of processing-intensive models when multiple model runs are initiated simultaneously.

2.2 Geo-Spatial dependency

As natural seismicity models typically cover large regions that may be required to be specific shapes, polygons are now used to define the forecast area. This moves away from a simple bounding box which may be insufficient to meet all the requirements of large-scale European models. The definition of a bounding box has been kept as an optional way of configuring the area, however this will be translated and stored as a polygon within RT-RAMSIS.

2.3 Results

Results from the seismicity forecast models can now be stored with a spatial geometry bounded by a polygon. The spatial area defined is the area of interest for which the results are valid, which may cover the entire region of the earthquake catalog if a single rate is determined, or if an earthquake catalog is generated for this area.

If a seismic model generates rates with a spatial resolution, this can be captured by describing each spatial bin with its own polygon. The spatial bin also optionally defines minimum and maximum depth which allows 2D and 3D results to be described.

The database column has the potential to accept any type of geometry described in the GeoAlchemy2 (i.e., a Python SQL Toolkit and Object Relational Mapper) geometry types list: https://geoalchemy-2.readthedocs.io/en/0.12.5/types.html.

This allows flexibility for future models which may have unexpected areas of interest, and where the region may be made of sub-regions that cannot be captured as a polygon.

The RT-RAMSIS data model has the option to store both rates and seismic catalogs as output from the seismicity forecast model. The data model also copes with the instance of a model running many simulations or returning probability density functions.

3. Seismicity Forecast Models (SFMs)

3.1 SFM Worker

The seismicity forecast model ***worker*** (sfm-worker) has been simplified and made more dynamic. Now there is less interaction required from the user to setup earthquake forecast models that can be run with this web service.

Rather than adding a new model entry to the *sfm database* on setup or adding a new model, the user may simply install the model code in the environment.

If the import module and Class of the new model code is defined in the RT-RAMSIS config, the software can dynamically import this code and run as usual.

The repository may be found here:
https://gitlab.seismo.ethz.ch/indu/ramsis.sfm.worker

3.2 SFM Wrapper

A new repository has been created for natural seismicity forecast models (nsfm models). This transforms the input parameters for ETAS (Epidemic-Type Aftershock Sequence) module (and is designed to more generally be used for other nsfm models).

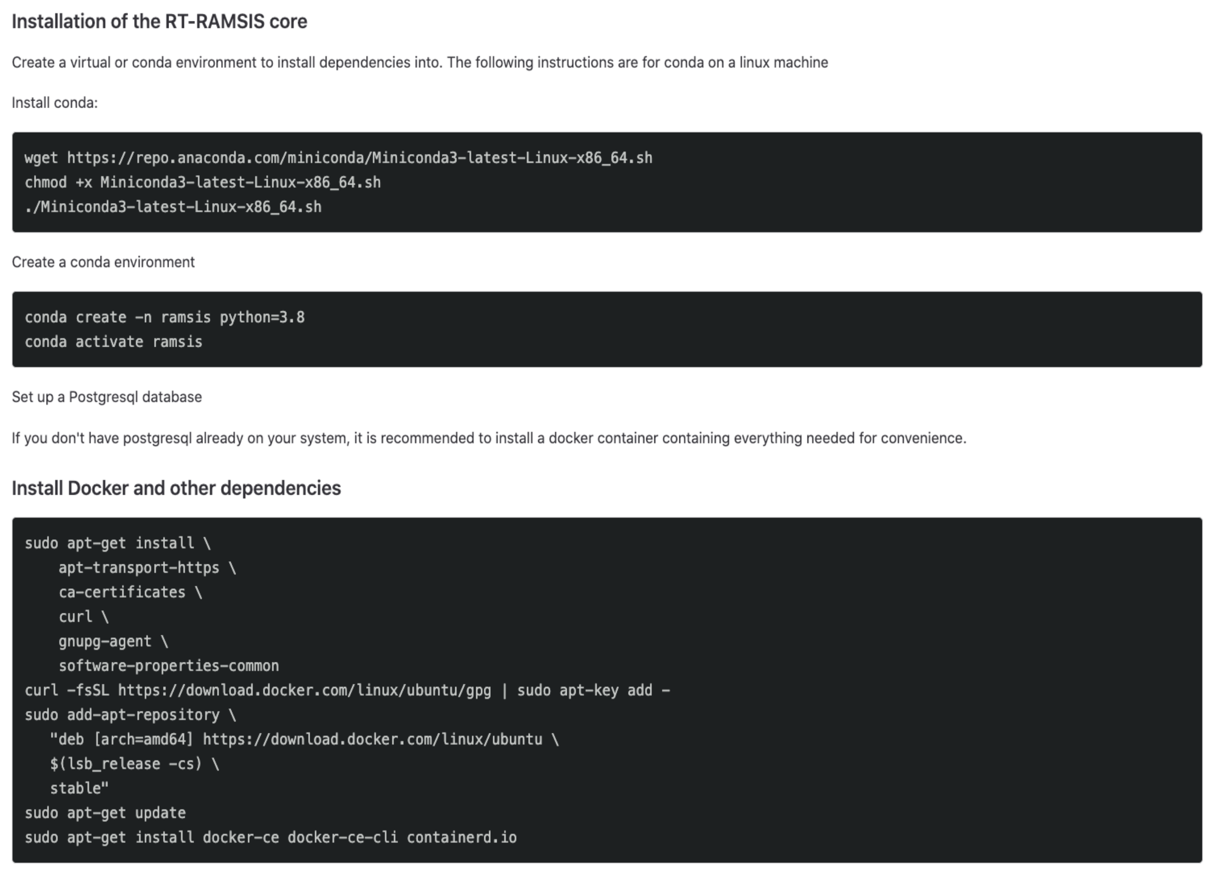
This wrapper to the core code of the ETAS module comes with base class so that new models can be incorporated into the RT-RAMSIS platform. This class specifies input parameters, result output, and methods for calibration and forecasting. This makes it easy for developers to know what specifications their code must have.

The repository can be found here:
https://gitlab.seismo.ethz.ch/indu/ramsis-nsfm

4. Seismicity Forecast Models (SFMs)

The readme of RT-RAMSIS is provided, where installation instructions are described, and an overview of the configuration and running is given:

https://gitlab.seismo.ethz.ch/indu/rt-ramsis/-/blob/feature/ramsis2-0



Installation of the RT-RAMSIS core

Create a virtual or conda environment to install dependencies into. The following instructions are for conda on a linux machine

Install conda:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

Create a conda environment

```
conda create -n ramsis python=3.8
conda activate ramsis
```

Set up a Postgresql database

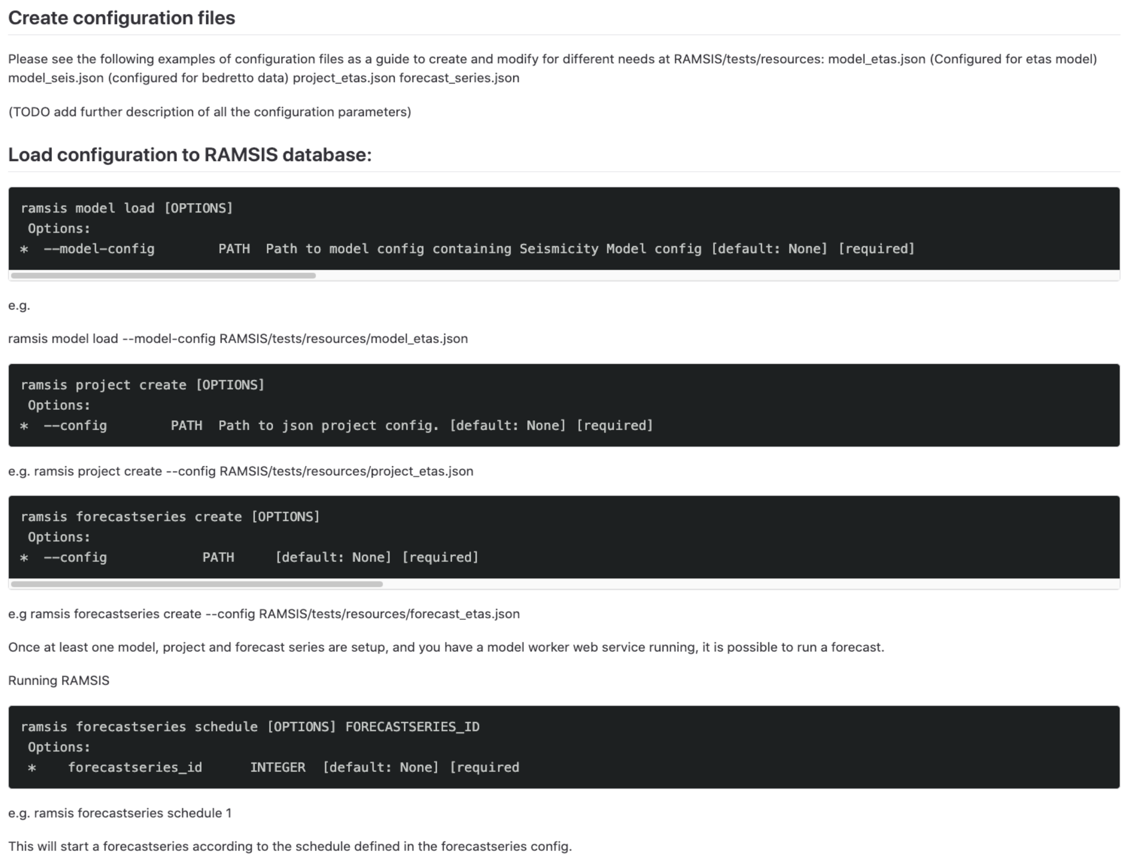
If you don't have postgresql already on your system, it is recommended to install a docker container containing everything needed for convenience.

Install Docker and other dependencies

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

The current version of the software is at the time of the writing of this report still under heavy development. Therefore, the latest branches of the respective packages and services are referenced. If those are not available anymore, it means that they got merged into the main branch. From that moment onwards the content of the main branch will hold the version of the software described here.

RT-Ramsis (core):	https://gitlab.seismo.ethz.ch/indu/rt-ramsis/-/blob/feature/ramsis2-0
Ramsis-worker:	https://gitlab.seismo.ethz.ch/indu/ramsis.sfm.worker/-/tree/feature/rise_updates
Ramsis-nsfm:	https://gitlab.seismo.ethz.ch/indu/ramsis-nsfm/-/tree/feature/rise_updates
Ramsis-ws:	https://gitlab.seismo.ethz.ch/indu/ramsis-ws/-/tree/feature/update_model



Create configuration files

Please see the following examples of configuration files as a guide to create and modify for different needs at RAMSIS/tests/resources: model_etas.json (Configured for etas model) model_seis.json (configured for bedretto data) project_etas.json forecast_series.json

(TODO add further description of all the configuration parameters)

Load configuration to RAMSIS database:

```
ramsis model load [OPTIONS]
 Options:
*  --model-config        PATH  Path to model config containing Seismicity Model config [default: None] [required]
```

e.g.

ramsis model load --model-config RAMSIS/tests/resources/model_etas.json

```
ramsis project create [OPTIONS]
 Options:
*  --config        PATH  Path to json project config. [default: None] [required]
```

e.g. ramsis project create --config RAMSIS/tests/resources/project_etas.json

```
ramsis forecastseries create [OPTIONS]
 Options:
*  --config            PATH     [default: None] [required]
```

e.g ramsis forecastseries create --config RAMSIS/tests/resources/forecast_etas.json

Once at least one model, project and forecast series are setup, and you have a model worker web service running, it is possible to run a forecast.

Running RAMSIS

```
ramsis forecastseries schedule [OPTIONS] FORECASTSERIES_ID
 Options:
 *    forecastseries_id      INTEGER  [default: None] [required
```

e.g. ramsis forecastseries schedule 1

This will start a forecastseries according to the schedule defined in the forecastseries config.

5. RT-RAMSIS - Core Concepts

5.1 Project

A project would normally be created for a work project, when planning to run models with the same data requirements.

As well as data requirements, it defines where to find the data via URLs (i.e., Uniform Resource Locator). Data can also be saved directly to the project on configuration, and in this case, the same data will be used for every forecast. A project's start time and end time defines the boundaries in time of input data if it is received via a web service.

5.2 Forecast Series

A forecast series is associated with a certain schedule and forecast region. Based on the start, end and interval time, forecasts are scheduled. 'tags' define which models should be run.

5.3 Forecast

A forecast has a defined start and end time, and contains model runs.

5.4 Model Run

A model run is associated with an individual model config. A task id that is returned from the sfm-worker will get stored, so that results can be retrieved. Results are stored here.

5.5 Schedule

A schedule is a regular occurrence according to a pattern. The only schedules that are enabled have a fixed time interval.

5.6 Model Config

Contains all the configurations for a SFM to be run. A single model existing on a web service can have many model configs, and a new model config must be created whenever a user wants to alter a parameter.

5.7 Configuration files

JSON (JavaScript Object Notation) formatted files containing all the required information to create database entries.

6. RT-RAMSIS - Workflow

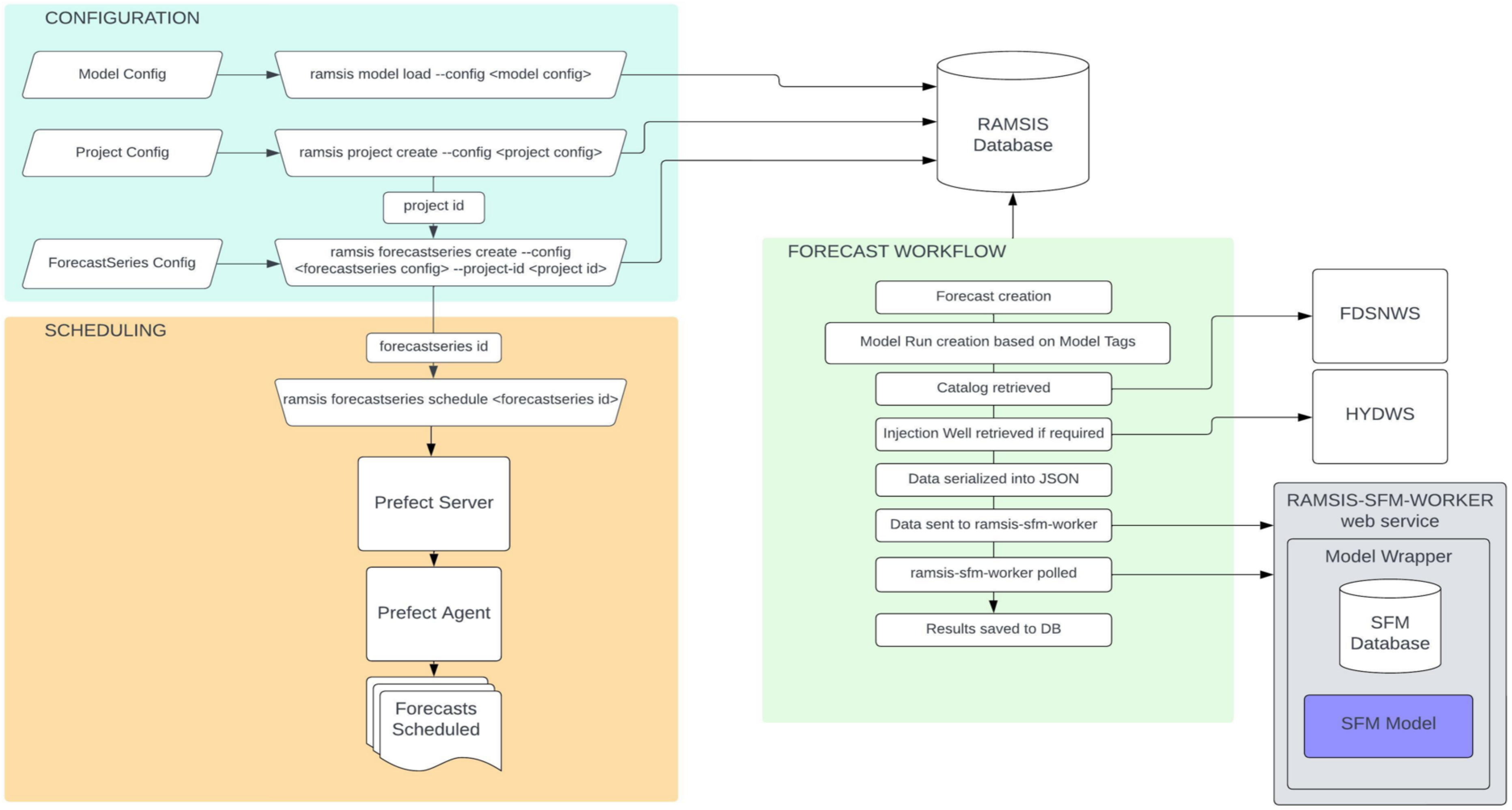
As shown in the workflow diagram, the user interacts with the software with a series of commands on a command line interface[1].

Once the user has configured models, projects and forecast series, a forecast series may now be scheduled. This schedule is picked up by a prefect server, which controls the scheduling of forecast workflows. When a forecast is due to be run, the prefect agent is called with instructions on where and how to run the forecast.

The forecast workflow creates database objects and collects data if required. The sfm-model-worker is then sent the data and RT-RAMSIS polls this web service.

[1]The most basic commands are shown to set up the minimum requirement for the configuration, but there are many options for the user to interact with the software. There are examples of configuration available either in the rt-ramsis/RAMSIS/tests/resources directory in the repository, or in the rt-ramsis/rise directory which contains the configuration for the operational setup of running daily forecasts for RISE.

RT-RAMSIS


CONFIGURATION
Model Config
ramsis model load --config <model config>
Project Config
ramsis project create --config <project config>
project id
ForecastSeries Config
ramsis forecastseries create --config <forecastseries config> --project-id <project id>
RAMSIS Database
SCHEDULING
forecastseries id
ramsis forecastseries schedule <forecastseries id>
Prefect Server
Prefect Agent
Forecasts Scheduled
FORECAST WORKFLOW
Forecast creation
Model Run creation based on Model Tags
Catalog retrieved
Injection Well retrieved if required
Data serialized into JSON
Data sent to ramsis-sfm-worker
ramsis-sfm-worker polled
Results saved to DB
FDSNWS
HYDWS
RAMSIS-SFM-WORKER web service
Model Wrapper
SFM Database
SFM Model

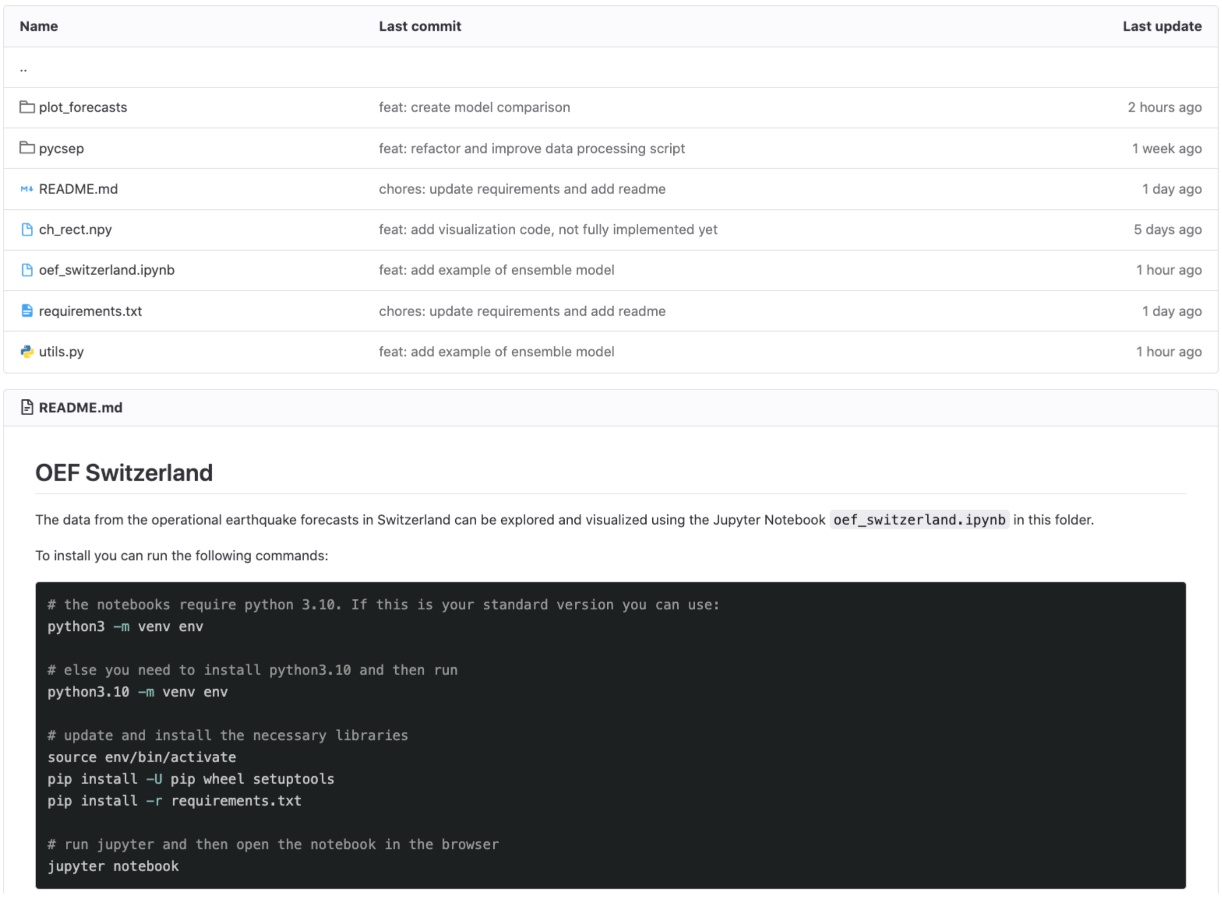
7. RT-RAMSIS - Results

The output from the model run is stored in the database in the following hierarchy:

→ Temporal Bins (starttime, endtime of the forecast time interval)
 → Spatial Bins (geometrytype, coordinates of the area of forecasting)
  → Catalogs (quakeml which describes predicted events. A list of catalogs describes results from multiple model simulations)
  or
  → Gutenberg-Richter parameters (a, b, alpha, magnitude of completeness. A list of results or pdfs may represent multiple model simulations)

7.1 Retrieving Results

A web service can be used to retrieve results from the database in a user-friendly way. A web service application must be running with access to the database. The web service can be used as the point of information access for web applications and individual users. The repository may be found here: https://gitlab.seismo.ethz.ch/indu/ramsis-ws.



Name	Last commit	Last update
..		
plot_forecasts	feat: create model comparison	2 hours ago
pycsep	feat: refactor and improve data processing script	1 week ago
README.md	chores: update requirements and add readme	1 day ago
ch_rect.npy	feat: add visualization code, not fully implemented yet	5 days ago
oef_switzerland.ipynb	feat: add example of ensemble model	1 hour ago
requirements.txt	chores: update requirements and add readme	1 day ago
utils.py	feat: add example of ensemble model	1 hour ago

README.md

OEF Switzerland

The data from the operational earthquake forecasts in Switzerland can be explored and visualized using the Jupyter Notebook `oef_switzerland.ipynb` in this folder.

To install you can run the following commands:

```
# the notebooks require python 3.10. If this is your standard version you can use:
python3 -m venv env

# else you need to install python3.10 and then run
python3.10 -m venv env

# update and install the necessary libraries
source env/bin/activate
pip install -U pip wheel setuptools
pip install -r requirements.txt

# run jupyter and then open the notebook in the browser
jupyter notebook
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821115

8. Ensemble Earthquake Forecast Models

Collaboration with the groups at INGV and University of Naples has provided an opportunity to post process and display model results using software that has been developed by them. A key feature of the software is that it can easily display outputs of ensemble models (see Deliverable 8.9).

Ensemble models are combinations of different ingredient models. Usually, they are multiplicative or linear combinations of their ingredients (Marzocchi et al., 2012), and the weight given to each ingredient model can be chosen to reflect different quantities (Herrmann et al., 2023).

To foster the integration of seismicity forecast models, the web service interface of seismicity forecast model is offered as an independent package i.e., seismicity forecast model worker available here: https://gitlab.seismo.ethz.ch/indu/ramsis.sfm.worker.

The standardized format in which RT-RAMSIS generates the forecasts enables modelers to easily access the data and create ensemble models. A simple example is done in the Jupyter Notebook which is used to visualize the output from Deliverable 8.9.

9. Display

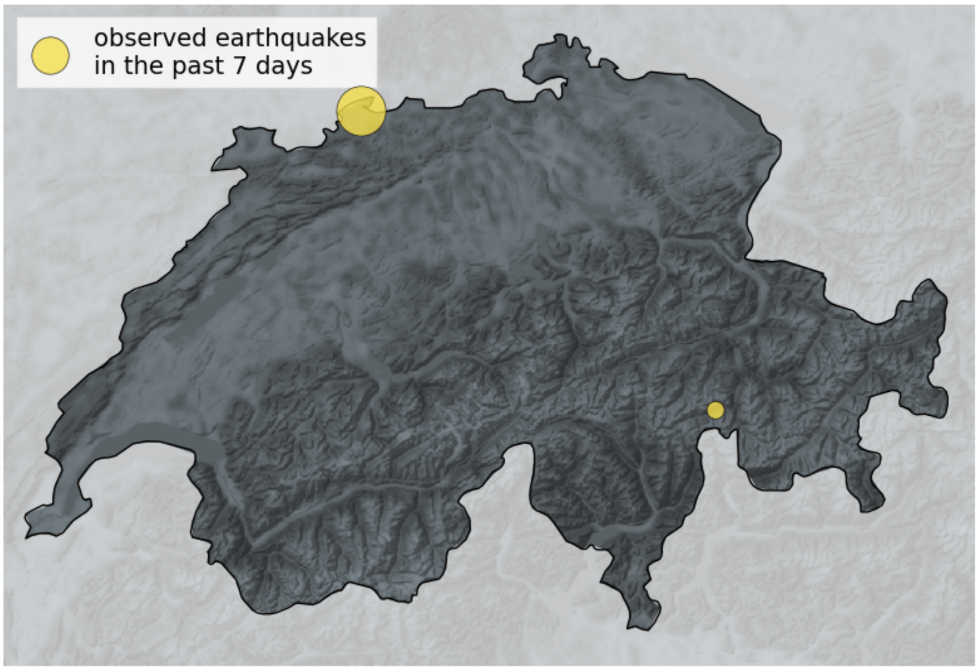
Research on how OEF forecasts should be communicated to societies has increased in the last years; however more efforts are needed. We here summarize the recommendations derived from interactions with societal stakeholders (Deliverable 5.5) as well as the recommendations from an expert elicitation with scientists working on OEF communication, modeling, and testing (see Deliverable 3.5).

For both Switzerland and Europe, OEF systems have been developed with the aim to communicate earthquake forecasts to different stakeholders.

In a first step, the forecasts may only be made available to employees of the Swiss Seismological Service internally, but the information to be communicated is the same that is intended to be communicated later to societal stakeholders.

Figure 1.1 shows four exemplary outputs which were generated based on the recommendations summarized above:

- a map of Switzerland with earthquakes of the past seven days;
- a scatter plot showing the observed events' times and magnitudes;
- a bar chart of observed event-counts for the past seven days. It transitions into a scatter plot, including confidence intervals, of expected future event rates; and
- temporal evolution of the probability that a felt or a damaging earthquake will take place in Switzerland within a day, for the next 30 days. These are only primary outputs which will be tested with end-users in the coming months and, afterwards, adjusted accordingly.



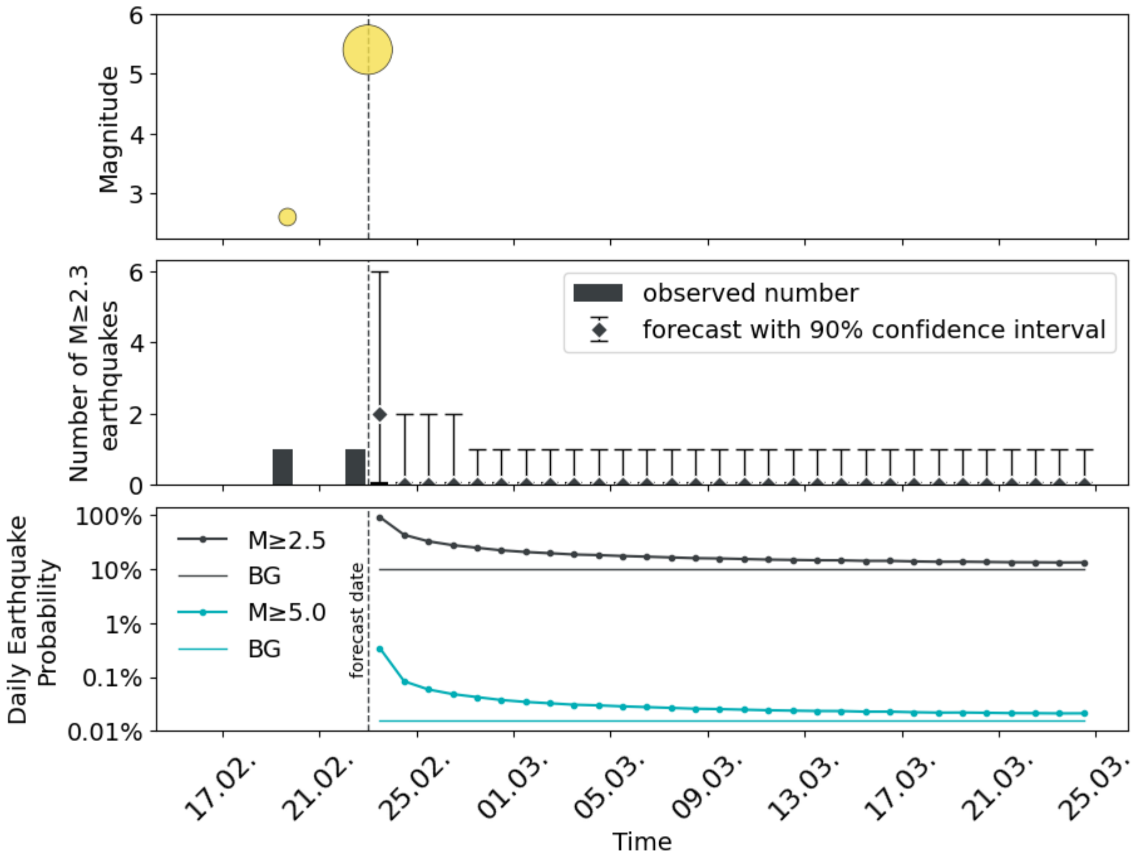


Figure 1.1: Draft of the information provided in a forecast for Switzerland, for the scenario of a M5.4 earthquake occurring close to Basel. Top shows the observed M≥2.3 earthquakes of the past seven days on a map. The second panel shows the same observed earthquakes on a time versus magnitude plot. The third panel shows the observed daily number of M≥2.3 earthquakes of the past seven days on the left, and expected seismicity on the right, separated by a dashed line indicating the time at which the forecast was issued. The forecast shows the expected daily number of M≥2.3 earthquakes, with 90% confidence intervals. Bottom panel shows the daily probability of the occurrence of one or more M≥2.5/M≥5.0 earthquakes anywhere in Switzerland, in the current sequence and during a quiet time.

10. References

Herrmann, M., & Marzocchi, W. (2023). Maximizing the forecasting skill of an ensemble model. Geophysical Journal International.
Marzocchi, W., Zechar, J. D., & Jordan, T. H. (2012). Bayesian forecast evaluation and ensemble earthquake forecasting. Bulletin of the Seismological Society of America, 102(6), 2574-2584.

Package requirements:

```
lxml>=3.3.3
marshmallow
numpy>=1.22.3
obspy>=1.3.0
PyOpenGL>=3.1.1a1
PyQt5
python-dateutil>=2.8.0
PyYAML>=5.1.1
ramsis.datamodel @ git+https://gitlab.seismo.ethz.ch/indu/ramsis.datamodel.git
ramsis.utils @ git+https://gitlab.seismo.ethz.ch/indu/ramsis.utils.git
requests>=2.18.4
transitions==0.6.9
prefect==0.15.10
pyproj>=3.2.1
psycopg2==2.8.3
jinja2
xmltodict
sqlalchemy>=1.4
typer
python-dotenv
```

RT-RAMSIS Copyright