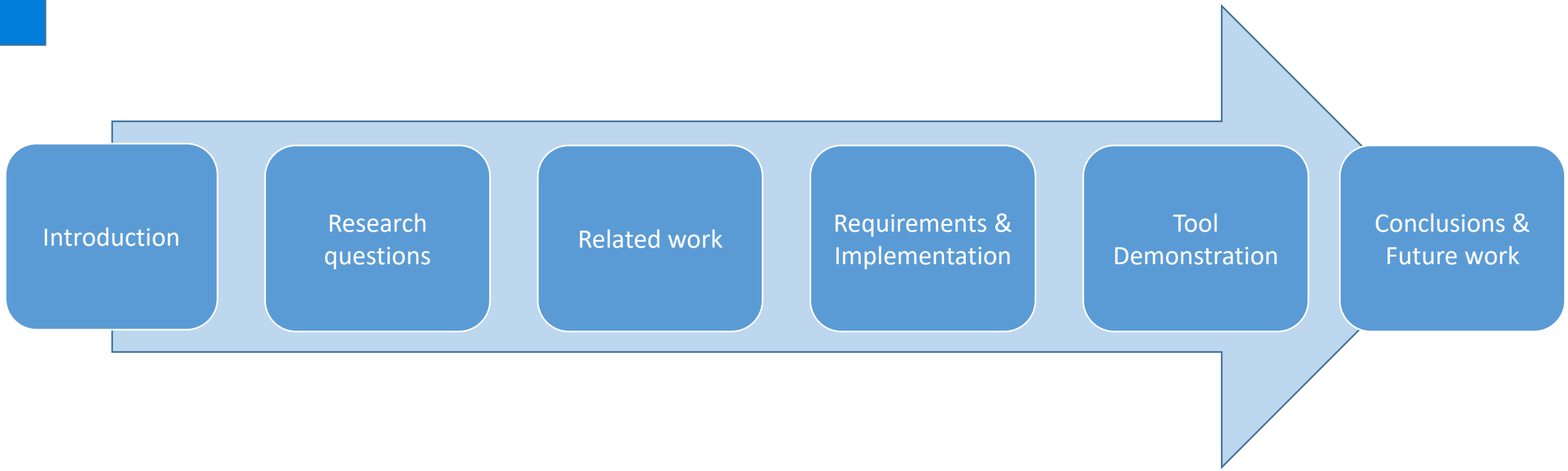
Design and Evaluate the OGC Web Services Architecture of a Geohazard analysis tool

Ioanna Micha MSc Geomatics

Mentor #1: Peter Van Oosterom
Mentor #2: Marianne de Vries
Co-reader #3 Pirouz Nourian
Deltares supervisors: Gerrit Hendriksen, Mike Woning

Presentation’s Structure



Spatial
Data formats, Services, Tools, Metadata

Interoperability- Share-Reuse

OGC Services implementations

More than 6000

Approved or not

Commercial or free & open source

GeoServer
ArcGIS Server
GeoNetwork
PyWPS
ZOO-Project
…

OGC®

Standards- Implementations- Organizations

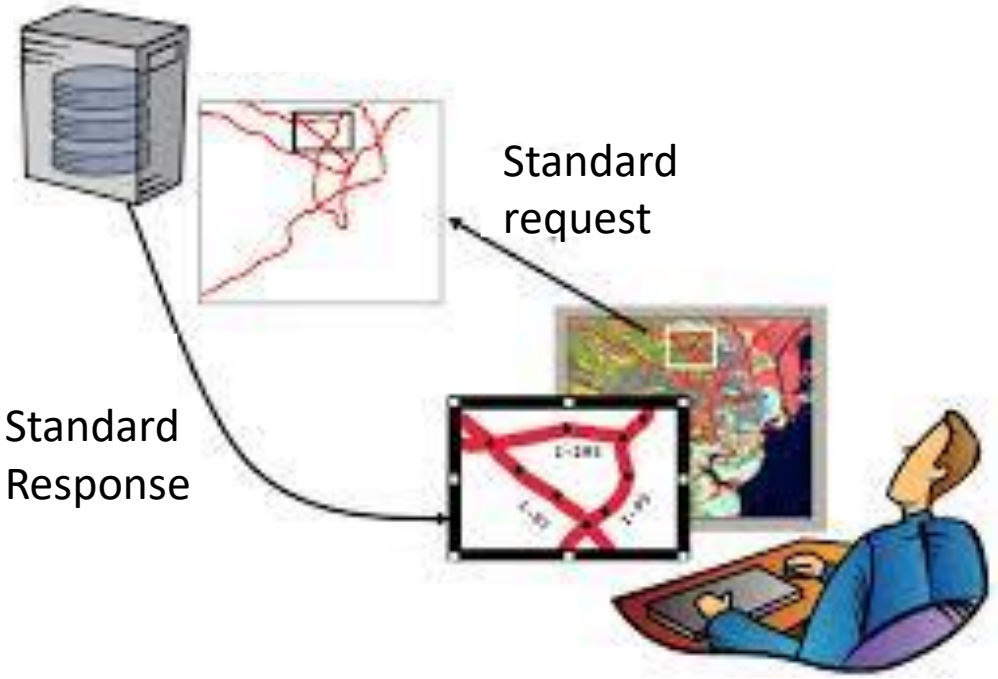
The Question is:

GIS Web applications can support:

Quality decision making

Complex geoprocesses

Up-to-date spatial datasets



Motivation

RI2DE GIS Web tool (Deltares, OpenEarth)

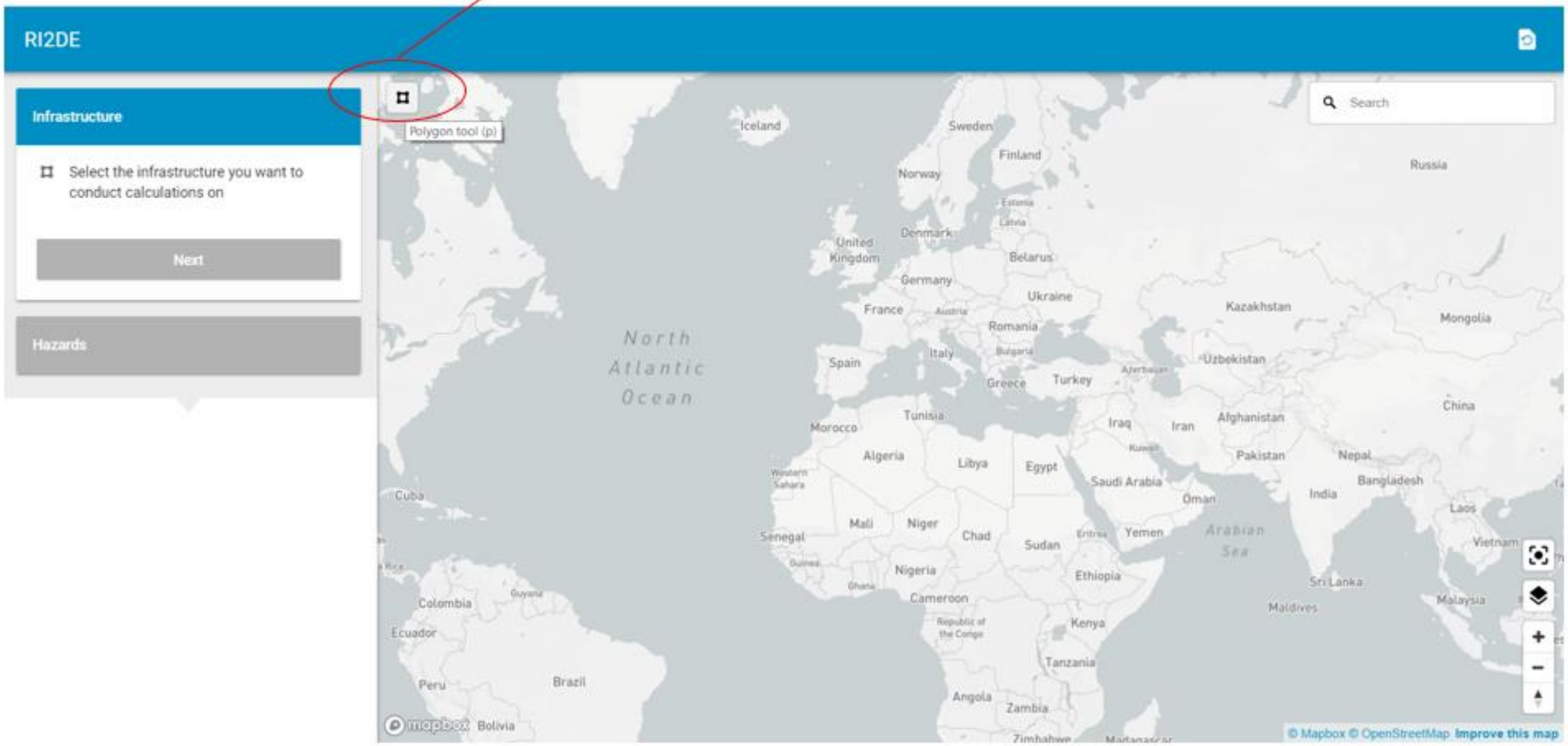
- Vulnerability assessment of areas around infrastructures
- Against climate related geohazards
- Based on ROADAPT VA (GIS Analysis)
- Use GIS datasets (e.g. DEM, Land use, Soil
- **OGC Web Services Architecture and standard data formats (E.g GeoJSON)**
- **Open Source technology components**

Landslides

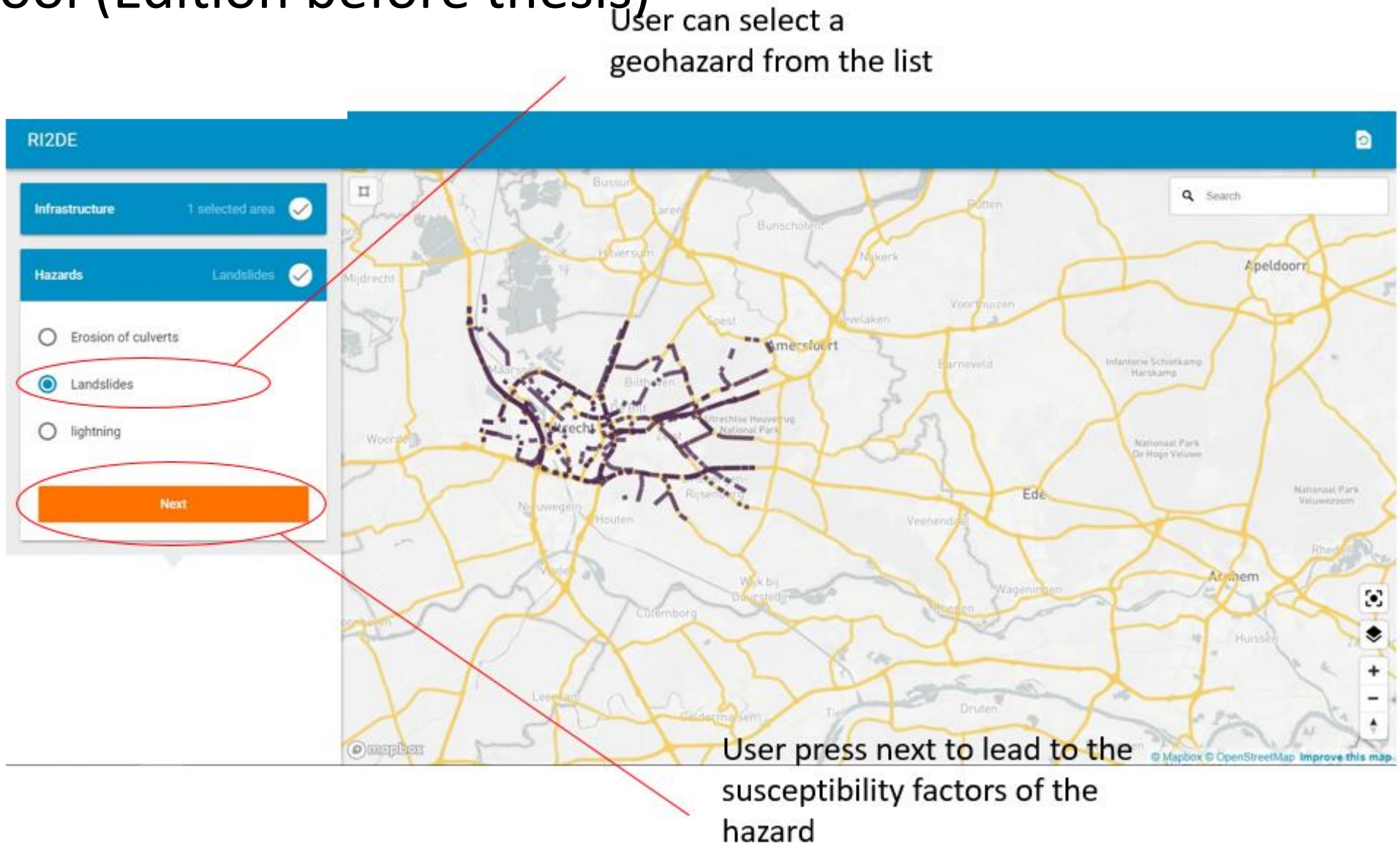
Erosion of culverts

RI2DE Tool (Edition before thesis) GUI(1)

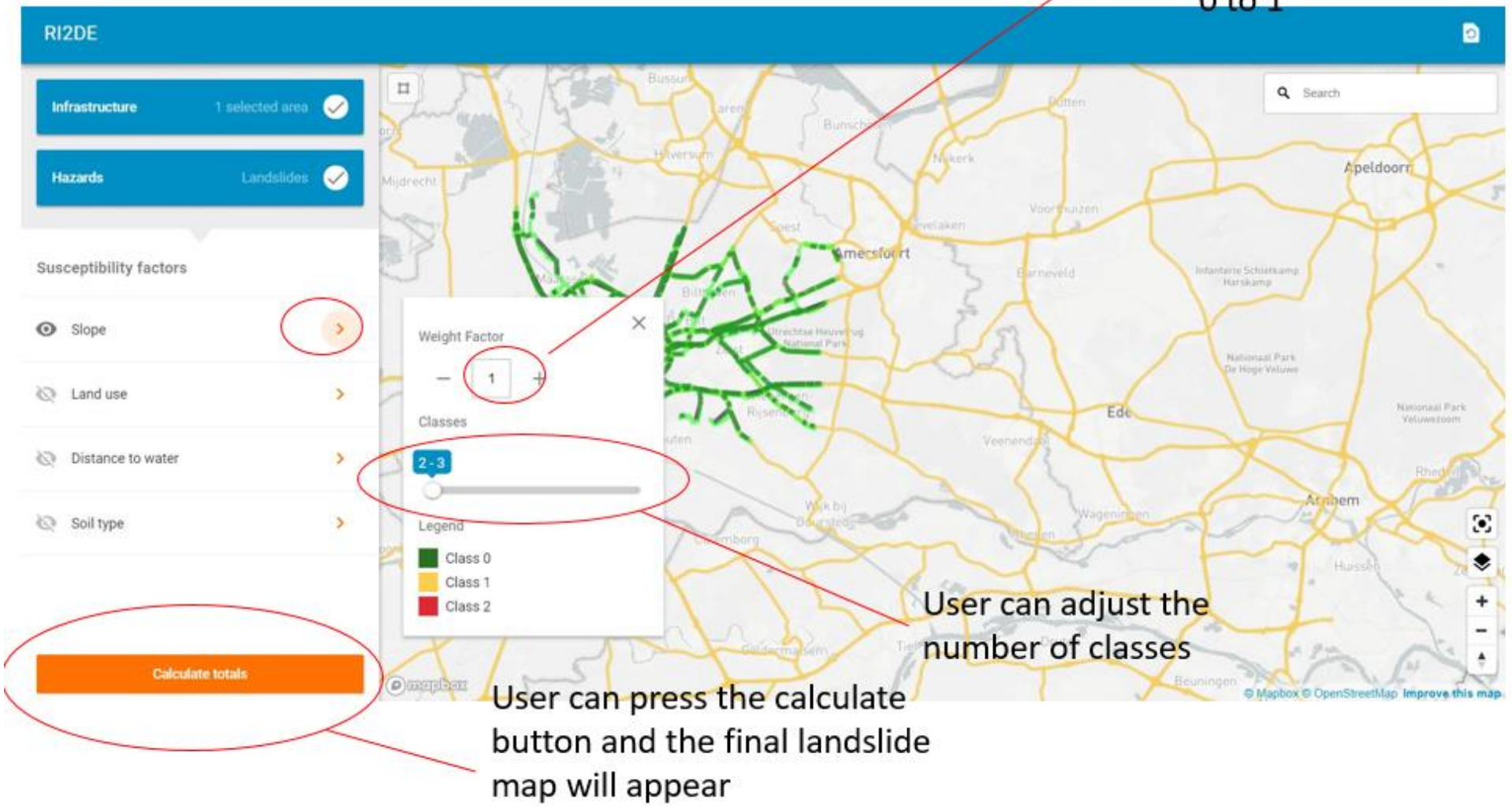
User can select the infrastructures with a polygon



RI2DE Tool (Edition before thesis) GUI(2)

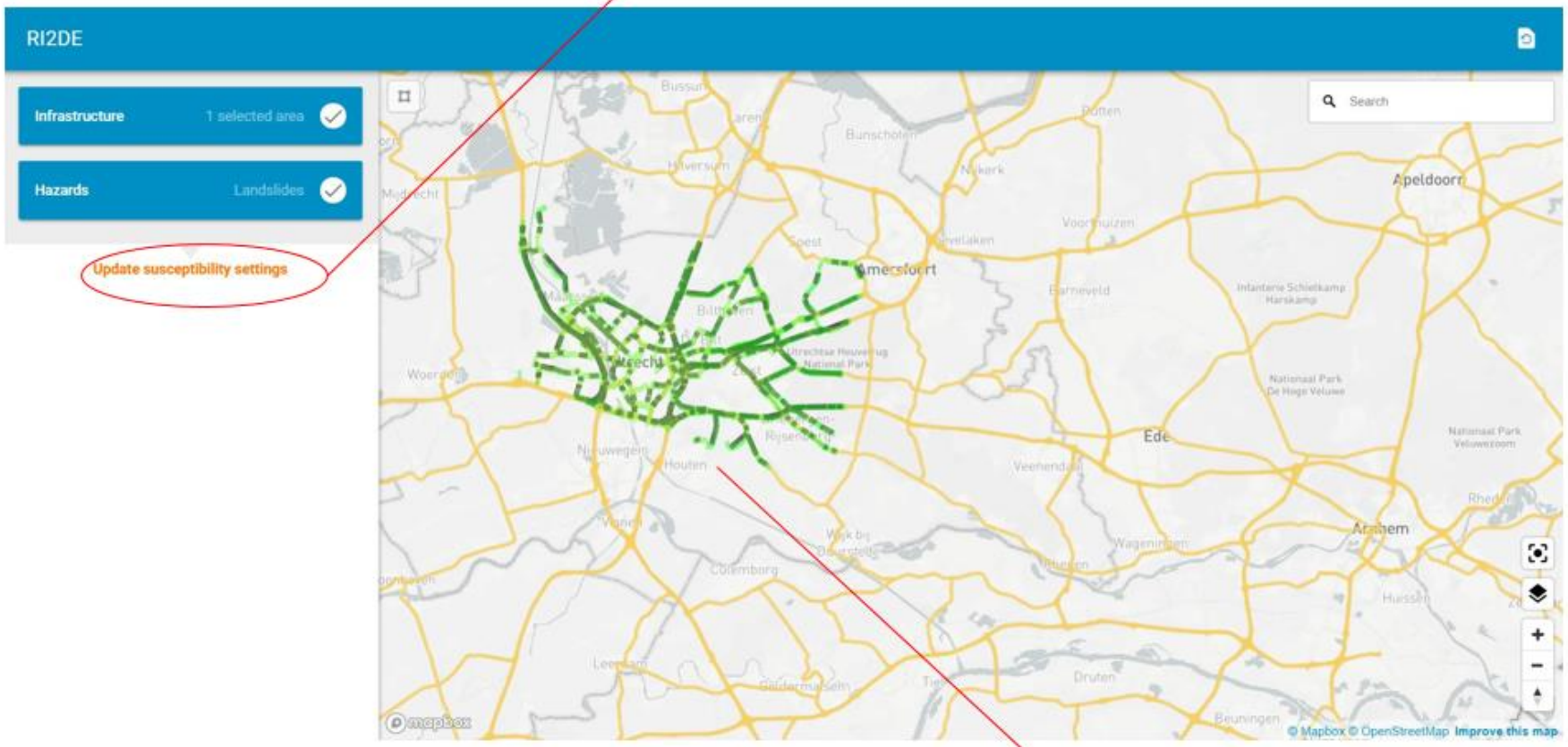


RI2DE Tool (Edition before thesis) GUI(3)



RI2DE Tool (Edition before thesis) GUI(4)

User can update susceptibility settings



Final susceptibility map of landslides

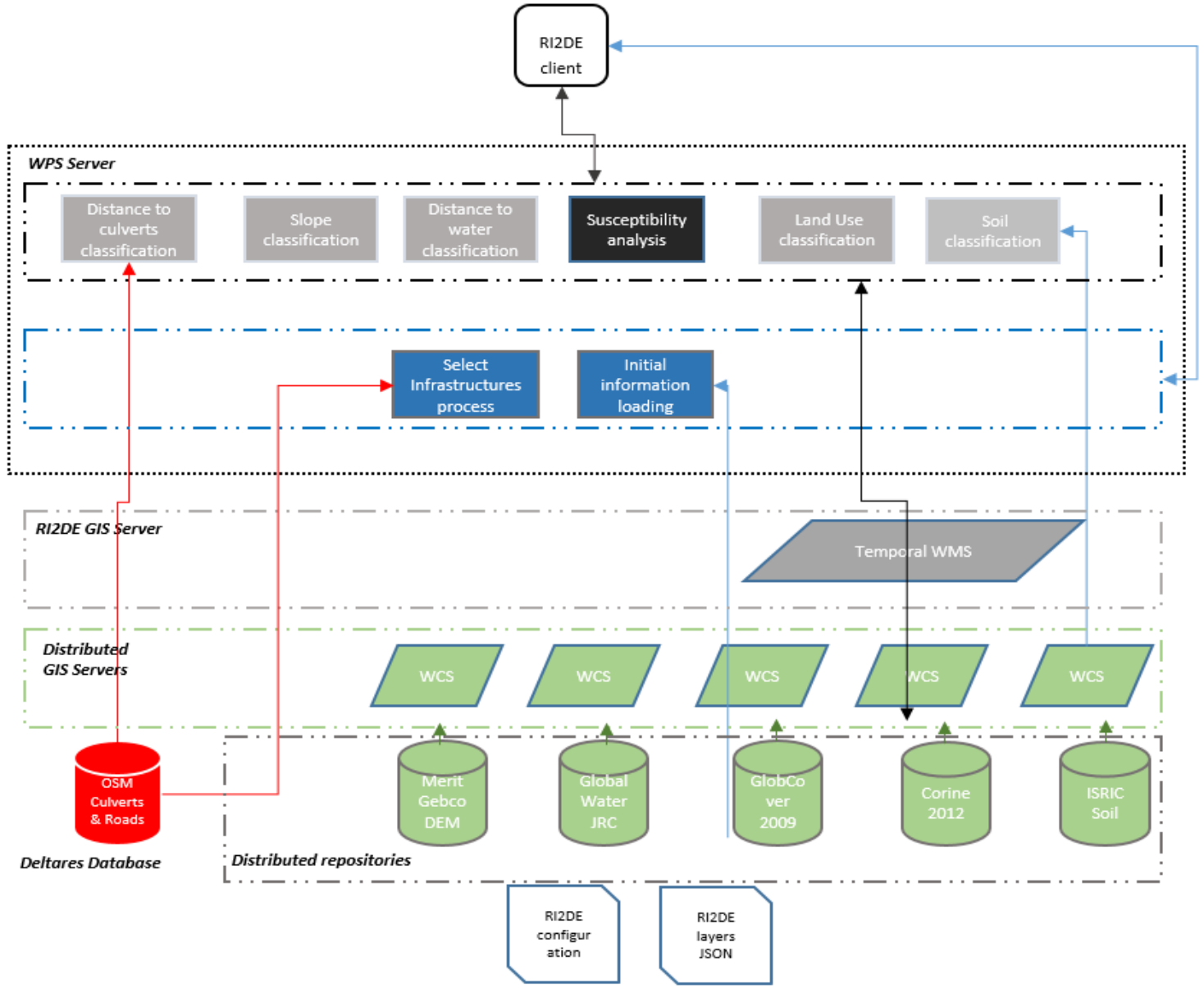
RI2DE OGC Web Services Architecture

Tool Processes as WPS

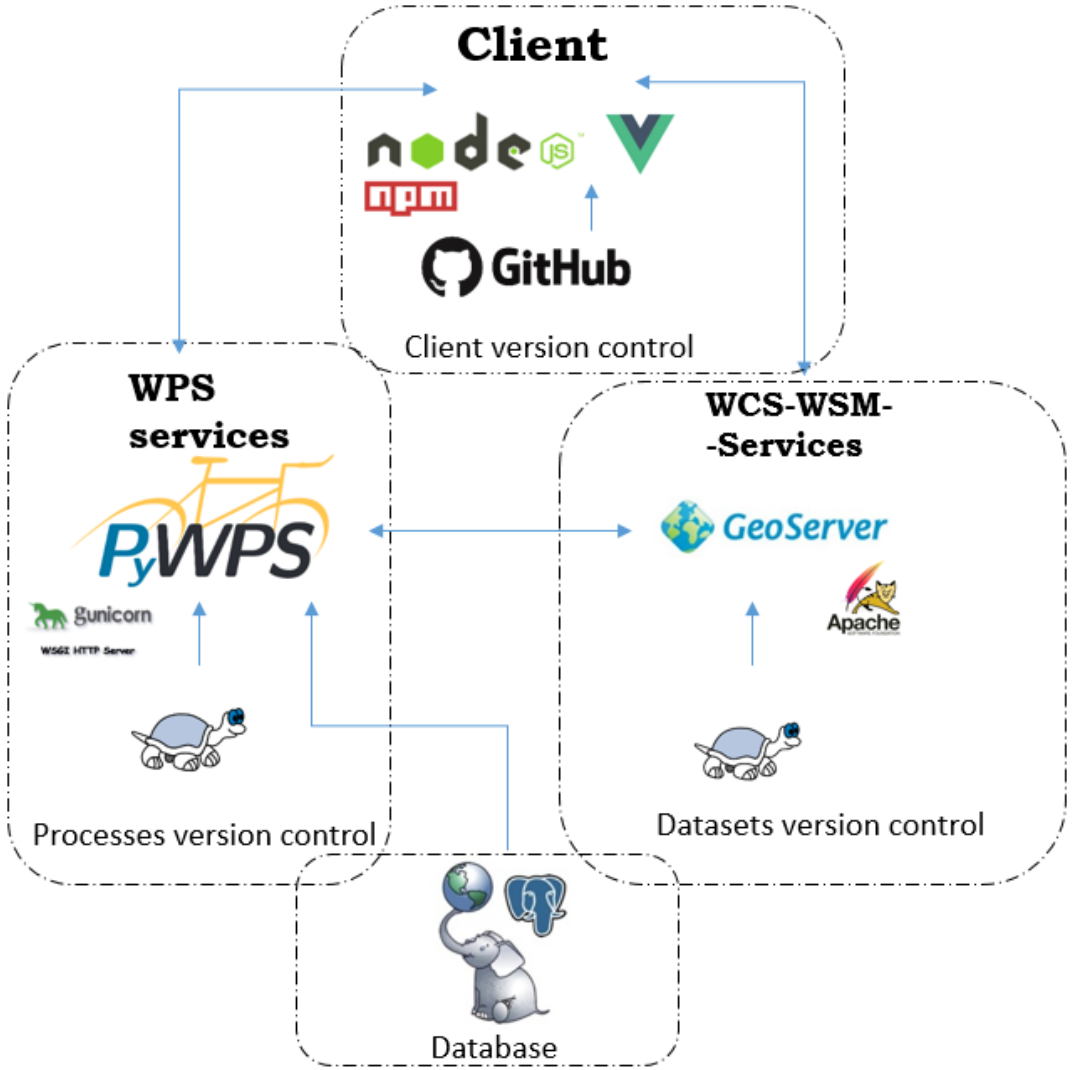
Results as WMS

Published as WCS

Global datasets



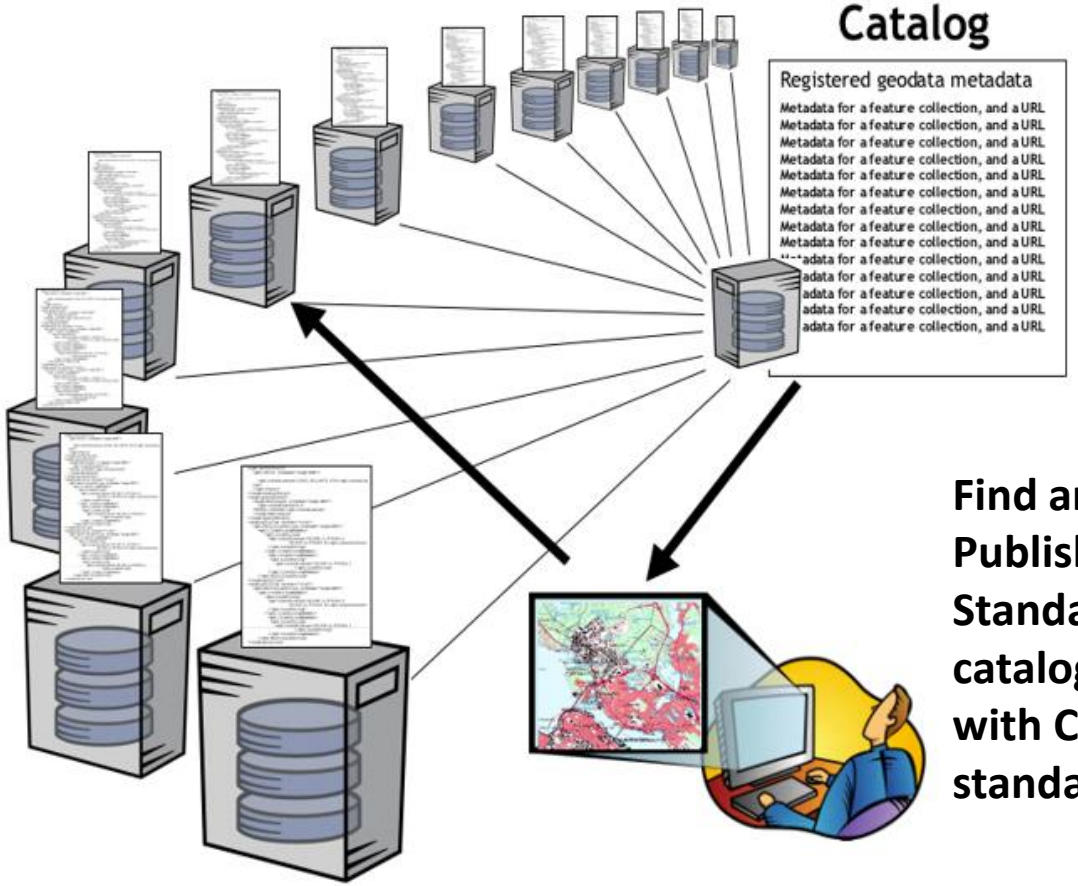
RI2DE technology components



RI2DE
New functionalities

Local, National SDIs, catalogues that implement the **OGC CSW** standard (e.g. INSPIRE)

DEMs, Land use, Water resource dataset with higher resolution, accuracy, detail



Find and Publish Standardized catalogues with CSW standard

RI2DE Improvements

Control of the process
&
Monitor of the process

WPS 2.0.2:

GetCapabilities
DesrcibeProcess
Execute
GetStatus
GetResult
Dismiss

Asynchronous execution

RI2DE WPS 

How much to complete

Status report messages (e.g 20%)

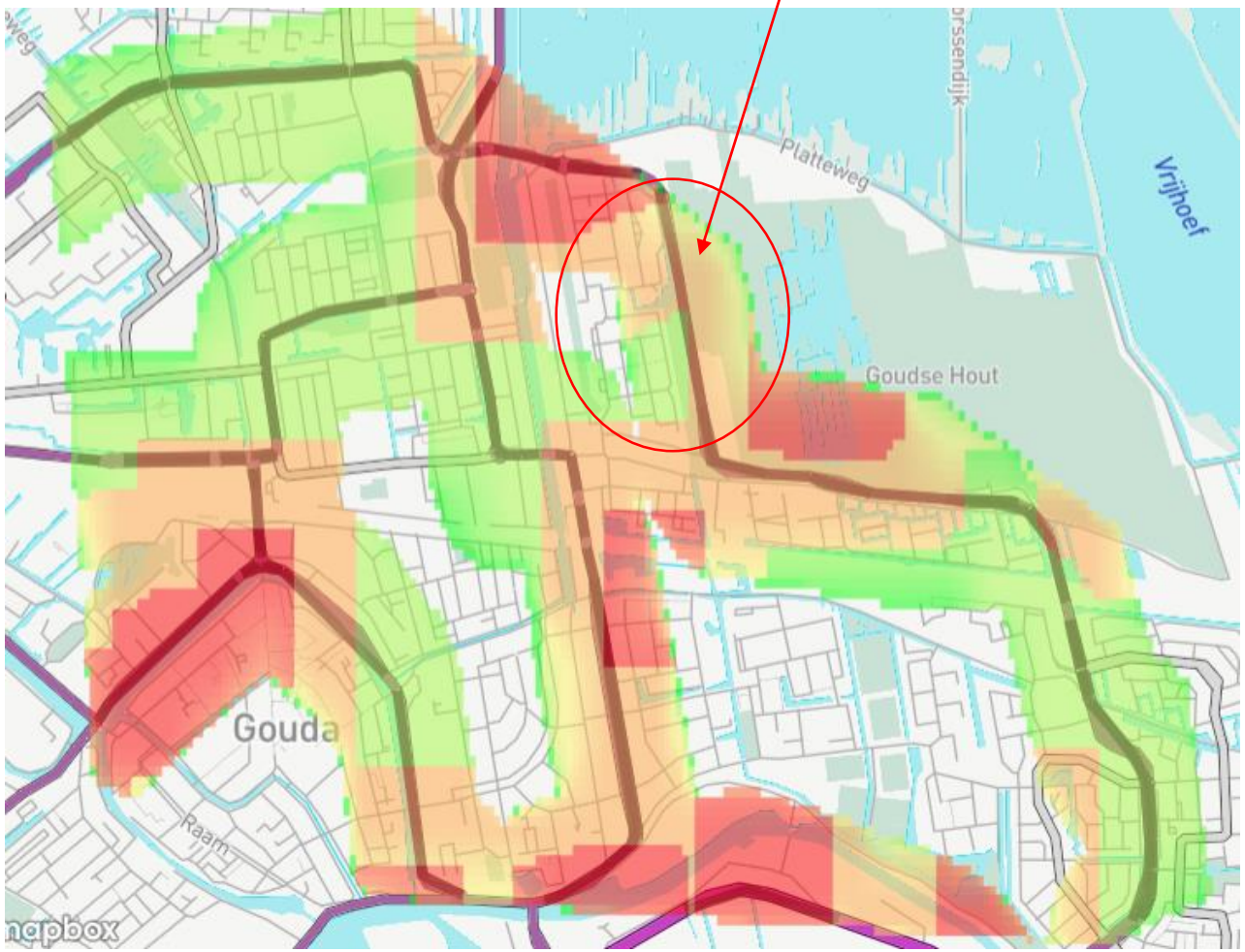
Too much time?

Stop it

RI2DE Improvements

New geo-process that
translate the risk on the **road segments**

Accuracy for cost benefit analysis



Research scope

- Research the WPS 2.0.2 and its free and open source PyWPS implementation (Try to implement it if the PyWPS supports it)
- Assess the CSW standard (how to find and access dataset through its metadata)
- Perform distributing search in the RI2DE web browser through CSW
- Make as much possible user friendly and self descriptive new GUI

Main research question:

"What will be the new OGC Web Services Architecture of the RI2DE tool?"

Sub questions:

- *“How flexible are the Web Processing Services of the tool”*
- *“How to get the metadata from the Catalogue Services?”*
- *“How to establish the connection between the CSW and WPS standard?”*
- *“What is the status of the PyWPS 4.0.0 with respect to the WPS 2.0.2 standard (test of asynchronous job control and job monitor)? “*
- *“What will be the new workflow of the user actions”?*

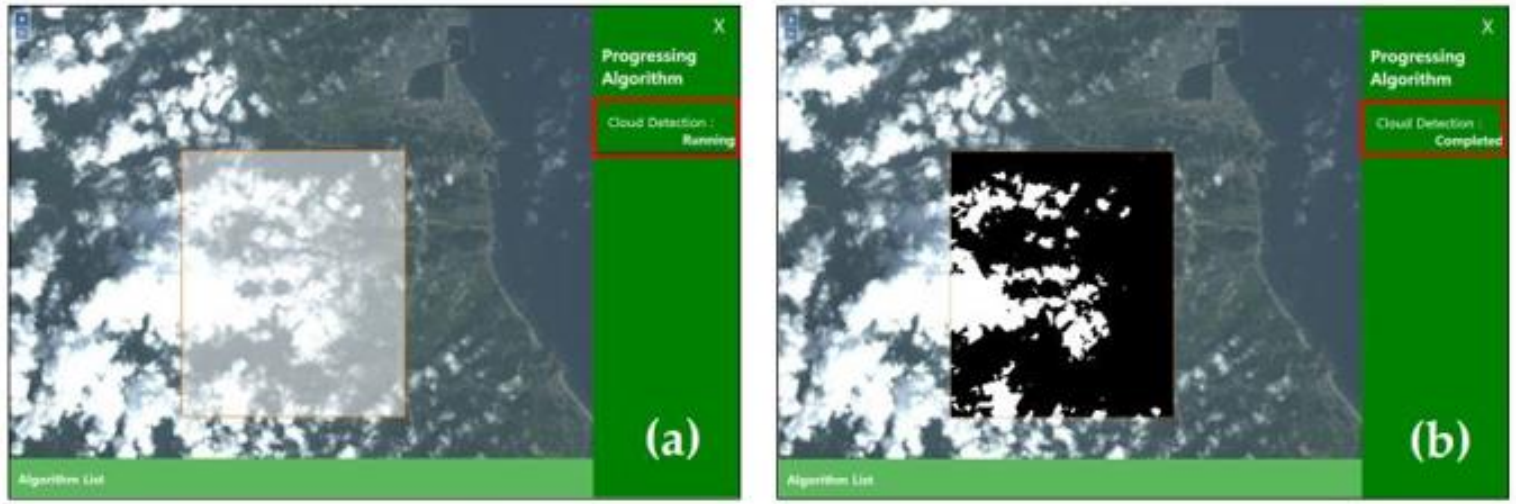
Other WPS implementations

- ArcGIS Server: Commercial solution, publish ArcGIS tools (e.g geoprocesses created with ArcPy library of ESRI), WPS 1.0.0
- Degree: Open source, written in Java, WPS 1.0.0
- 52 North: Free and open source, Java, no support for publishing as WPS 1.0.0, client side (support operations WPS 2.0.2)
- GeoServer: WPS 1.0.0, pseudo-operations (GetExecutionStatus and Dismiss)
- ZOO-Project: Written in C, Python and JavaScript, fully supports of WPS 2.0.0

Project with WPS 2.0.0 implementation

E-Government Standard framework of South Korea

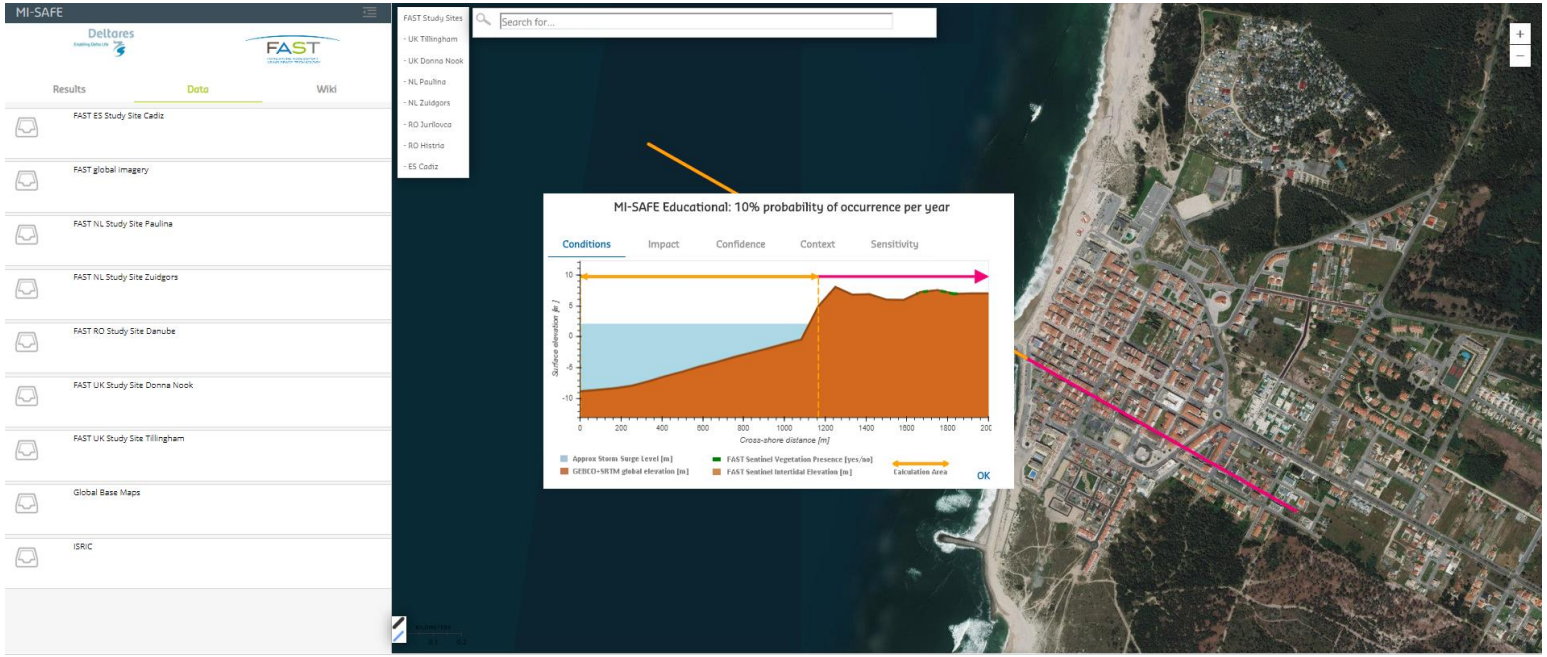
- Web platform based on OGC standards
- WPS that performs satellite image processing (WPS 2.0.2 with ZOO-Project)
- GetStatus (a)
- GetResult (b)



Another OpenEarth/Deltares GIS Web Project

MI-SAFE Viewer

- Based on OGC Standards
- Open source technology components
- Potential flood risk in the field sites
- **Fixed, local services** in the field sites



Requirements & Implementation

1. Discover-Retrieve-Display-Select-Use Source
2. Status reporting of the processes
3. Monitoring of the processes
4. Translate the risk on the road lines

Ideal Requirements

- **Discover & Retrieve** (Implement the *GetRecords* operation of the *CSW*)
- **Display, Select & Use** (Web Browser: Display the record services to the user in order to select and use one of them)
- **Reprojection** to WGS 84 in case of local datasets

Flexibility of the classification processes

In order to change the source the processes should be able to work with other datasets apart from default ones

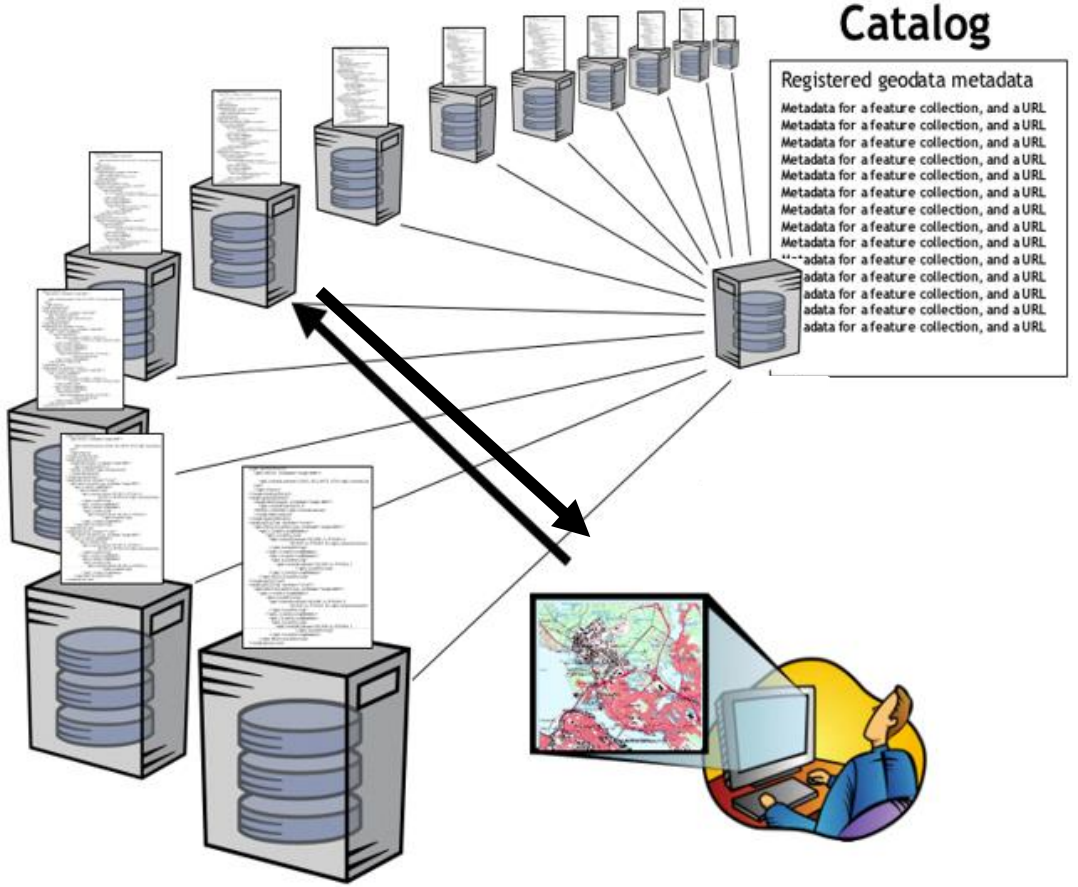
- **Land use** classification process: Developed based on the Corine 2012 and GlobCover 2009 WCS
- **Soil** classification process : Developed based on the SoilGrids250m of ISRIC WCS
- **Distance to culverts** classification process: Developed based on OSM culverts from database (No WCS or WFS service)
- **Distance to water** classification process: Flexible to accept other water surface datasets apart from the Globa water surface of JRC
- **Slope** classification process: Flexible to accept other DEM WCS

CSW Discovery of records

GetRecords response:

Core Returnables:

Title,
Publisher:
{layer name & OWS URL of the services,
Description {Abstract}



Metadata in XML encoding of **ISO** or Dublin

GetRecords request:

- Query: OGC Filter Encoding
- Keywords to query against *AnyText* (**Core Queryable**)
- BBOX to query against BoundingBox (Core Queryable)

Implementation (1)

- Discover Geoportals that:
  - Support the CSW
  - Have registered records of DEM and water surface WCS, with OWSurl, and layername that they are published.

Apparently not an easy task to find…

- Did not find Geoportals that have Water surface WCS
- Not In general have registered OWSurl and Layername

For that reason…

Implementation (2)

- The Catalogues that were used for testing purposes:

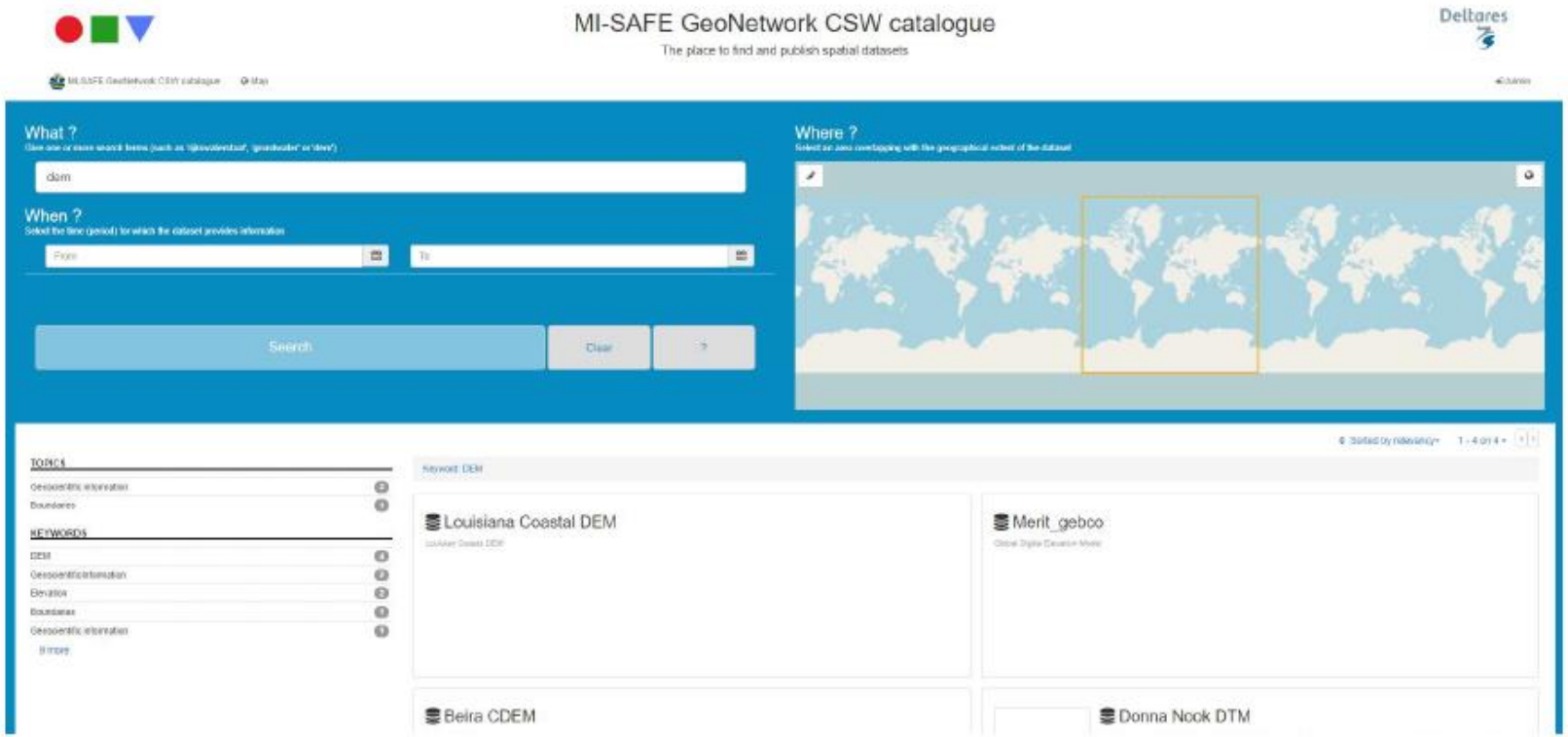
MI-SAFE Catalogue of Deltares & a Catalogue that was created with the GeoNetwork implementation of the CSW, for the purpose of this research.

Records were added and altered according to the needs of the research

- Title
- **Layer Name**
- **OWS Url** } **Added**
- Abstract

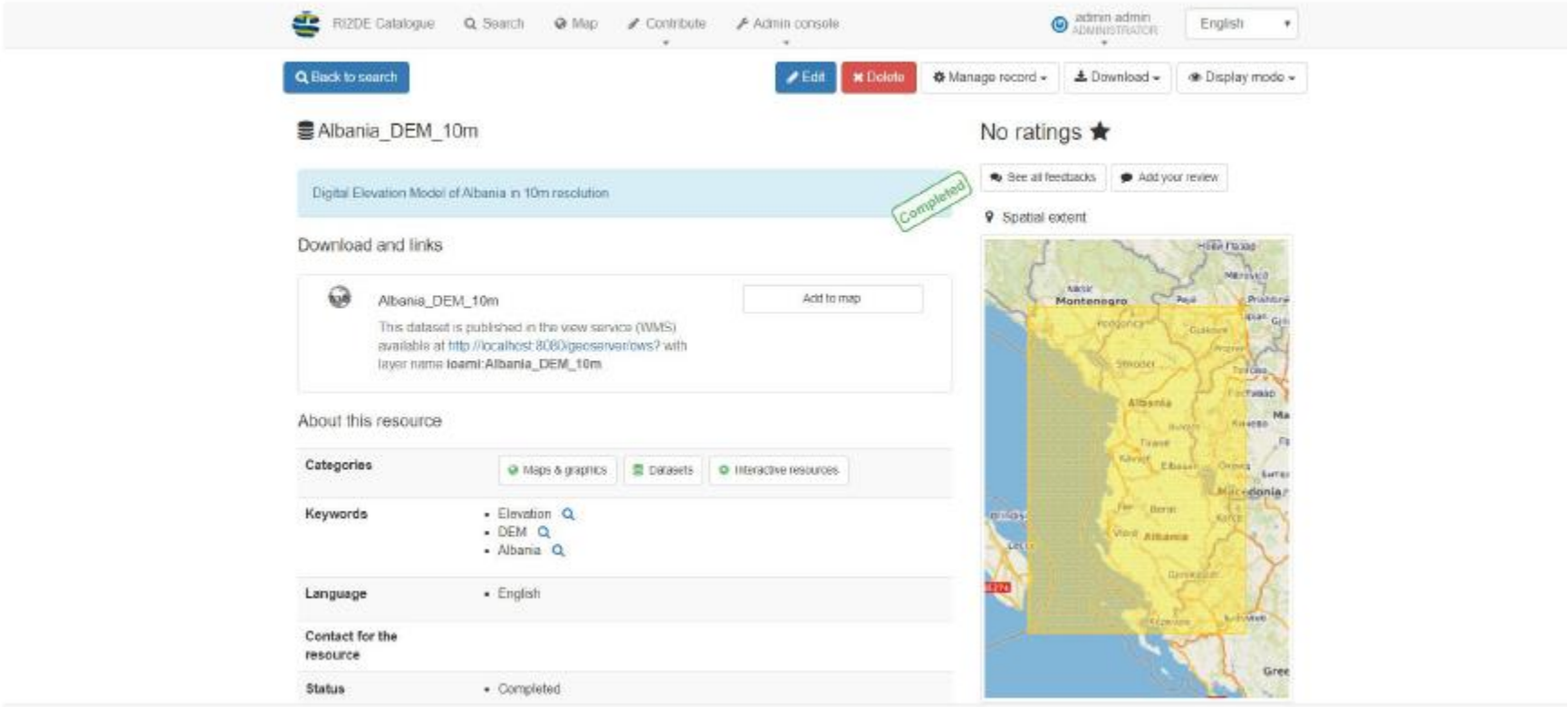
Implementation (3)

- Metadata Records for:
  - Louisiana WCS (*Deltares GeoServer)*
  - Beira WCS *(Deltares GeoServer)*
  - Merit Gebco WCS *(Deltares GeoServer)*
  - SRTM WCS *(Deltares GeoServer)*



Implementation (3)

- Metadata Records for:
  - Albania WCS (*Localhost GeoServer)*



Implementation (4)

Discover & Retrieve:

- Implement the GetRecords either:
  - In the RI2DE Web browser sends (XML request) and process the returned records
  - OWSlib python package in the back-end

More specifically..

Implementation (4)

Discover & Retrieve:

- OWSlib
  - Is client side implementation of the OGC Standards
  - Creates and sends requests of different operations
  - Receives and reads the responses
  - **Supports the OGC Filter Encoding**
  - **Supports the GetRecords operation of the CSW**

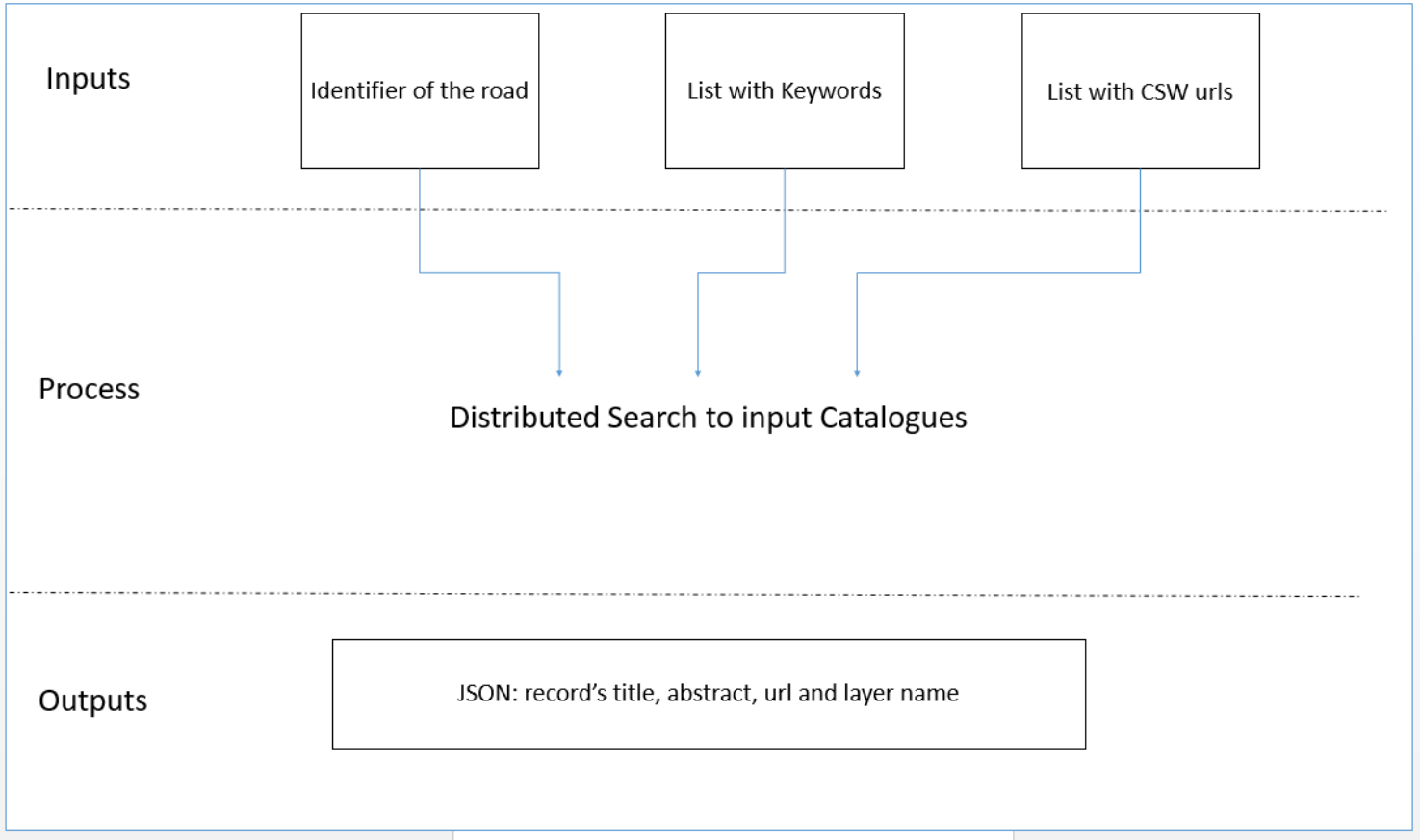
Implementation (4)
Discover & Retrieve:

- Created a process:
  - Connects to the catalogue server (with CSW endpoint):
    1. Creates OGC Filter object with OWSlib from (Bounding box and keywords)
    2. Creates and sends the GetRecords request with the OWSlib package
    3. Read response
    4. Check for duplicate records and if records have layer name and url, title and abstract (If not, pass them)
    5. Extract Title, Abstract, OWSurl, layer name
    6. Returns JSON with all records that were found

Implementation (4)

Discover & Retrieve:

- Published as WPS with the PyWPS
- Standard inputs with configurable values



Implementation (4)

Example Beira Africa:

Request (*road id of the roads*,
Keywords: DEM, Elevation
Catalogue Server:
MI-SAFE GeoNetwork,
Localhost GeoNetwork

BBox created from the id of the roads,

Beira Africa

Response of the WPS


```
[{"abstract": "Global Digital Elevation Model
 (DEM) derived from the SRTM (Shuttle Radar Topography Mission).
 The data is stored on the Deltares P:drive.
 The SRTM elevation data can also be downloaded
 from the website http://gdex.cr.usgs.gov/gdex/.
 This global dataset is freely available, you only
 need to register first. There is a demo with instructions on
 the download procedure (http://gdex.cr.usgs.gov/demo/demo.html).",
 "layername": "global:srtmplus15",
 "owsurl": "https://deltaresdata.openearth.eu/geoserver/ows?",
 "title": "global:srtmplus15"},
 {"abstract": "Digital Elevation Model Merit Gebco",
 "layername": "Global_Base_Maps:merit_gebco",
 "owsurl": "https://fast.openearth.eu/geoserver/ows?",
 "title": "Global_Base_Maps:merit_gebco"},
 {"abstract": "Beira CDEM",
 "layername": "Global_Base_Maps:Beira_CDEM",
 "owsurl": "https://fast.openearth.eu/geoserver/ows?",
 "title": "Global_Base_Maps:BeiraCDEM"},
 ]
```

Implementation (5)
Display & Select

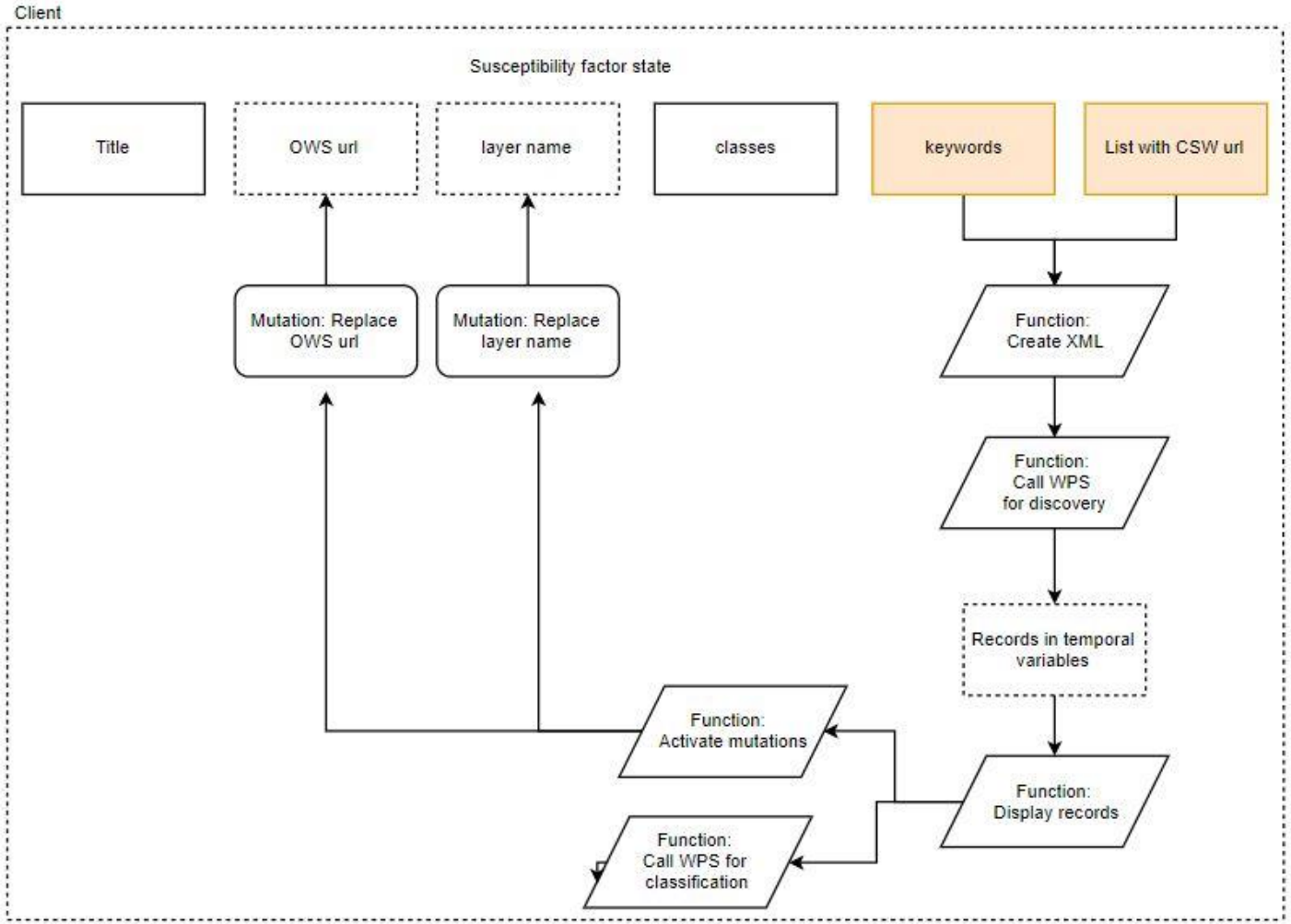
Add Keywords and CSW endpoints at the configuration file that setup geo-hazards and their susceptibility factors on the client:

- For every susceptibility factor is provided:
  - Title
  - WPS function id to call it
  - Classes boundaries (Only if configurable)
  - The layer name of the WCS in order to get the coverage (Only if configurable)
  - The OWSurl of the WCS in order to get the coverage (Only if configurable)
  - Keywords list for every factor
  - Catalogue Servers (CSW endpoints)

Implementation (5)
Display & Select

Client Side Processing:

- New states to store: Keywords & CSW URLs
- Function that creates & send Execute request to the WPS
- Function that reads the response
- Function that display the records
- In case of selection: Mutate the OWSurl and
- Layer name States
- Select & Resend the Execute request to the Slope classification process



Implementation (5)
Reprojecting

Wanted in the classification process to:

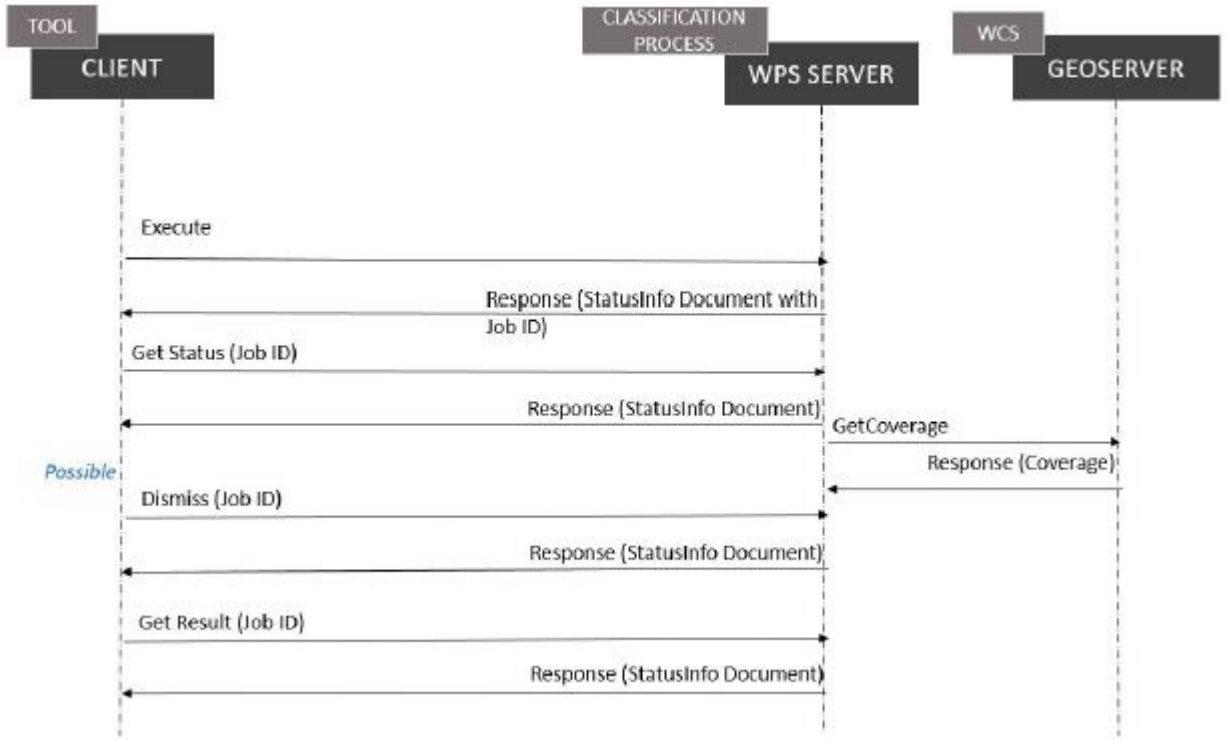
- Connect to Geoserver and extract projection information of the WCS
- Reproject bounding box to the local projection of the WCS
- Get the coverage from the GeoServer with this bounding box
- Reproject the coverage to the global projection (WGS84)

But..

Long time processing (So it was abandoned for now)

Ideal Requirements

- Asynchronous Execute request
- Implement the GetStatus (PyWPS does not support it yet)
- Implement the GetResult (PyWPS does not support it yet)



Asynchronous Execute Request of the RI2DE tool

Status Reporting
(PyWPS) But PyWPS offers a way of status reporting
How?

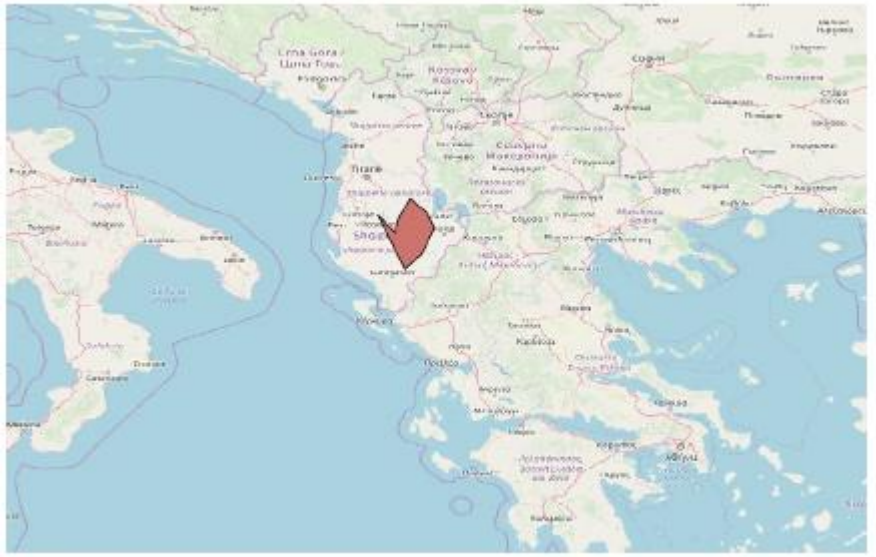
- Set output URL to store the status messages
- Set in the WPS class of the PyWPS the:
  - Status report to true
  - Store supported to true
- Set in the process when the status will be updated

Example with Albania WCS in Localhost…

Status Reporting (PyWPS)

```
def_handler(self,request,response):
...
    response.update_status('Read Input', 5%)
...
    response.update_status('Get Roads', 40%)
...
    response.update_status( 'Converted to slope', 70%)
...
    response.update_status('Classified', 80%)
...
    response.update_status('Applied Mask', 90%)
...
    response.update_status('Uploaded to GeoServer', 100%)
...
```

Expected status report messages



Selected big area in Albania, DEM resolution is 10m

Output was…

Status Reporting (PyWPS)

- First response: Process Accepted, status 0%
- status document: Process Started, status 40%
  (road lines have been retrieved from database)
- status document: Process Succeeded

Never got status for 70, 80, 90 per cent
either..

Wrong/not accurate estimation
Fast process (Needs higher resolution datasets more than 10m)

Not implemented

Ideal Requirements

- Implement the Dismiss Operation (PyWPS does not support it yet)

  Efforts for isolation of the processes (e.g. Docker containter) essential for control of the process

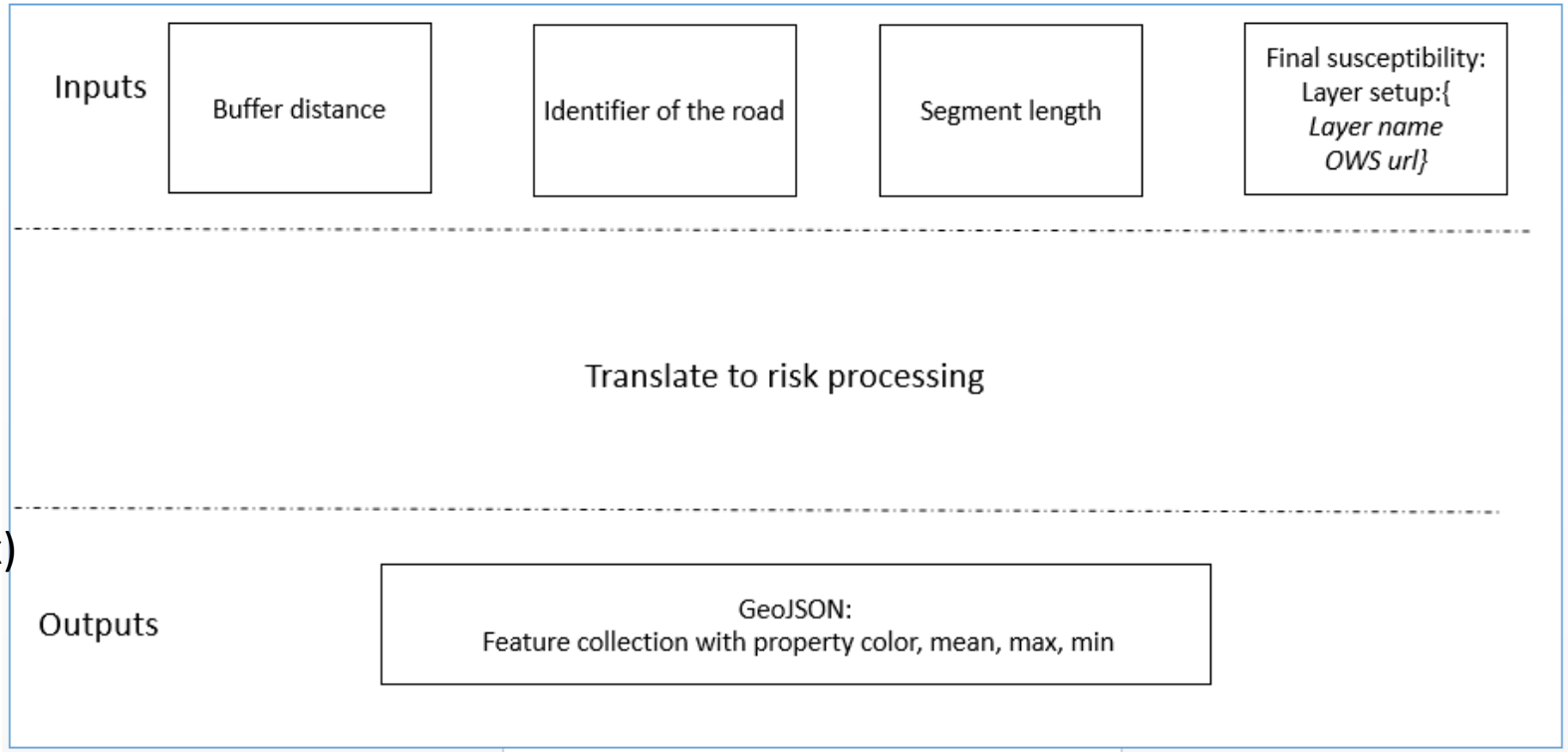
Ideal Requirements

4 Translate the risk on the road lines

- Back-end: Develop the process that transfer the mean vulnerability raster values to the road lines (Implemented by the back-end developer of the tool)
- **Front-end:**
  - **Develop the functions to send the request the WPS**
  - **On the fly reclassification of the road lines with the retrieve GeoJSON**
  - **Interface design and development**

Implementation (1)

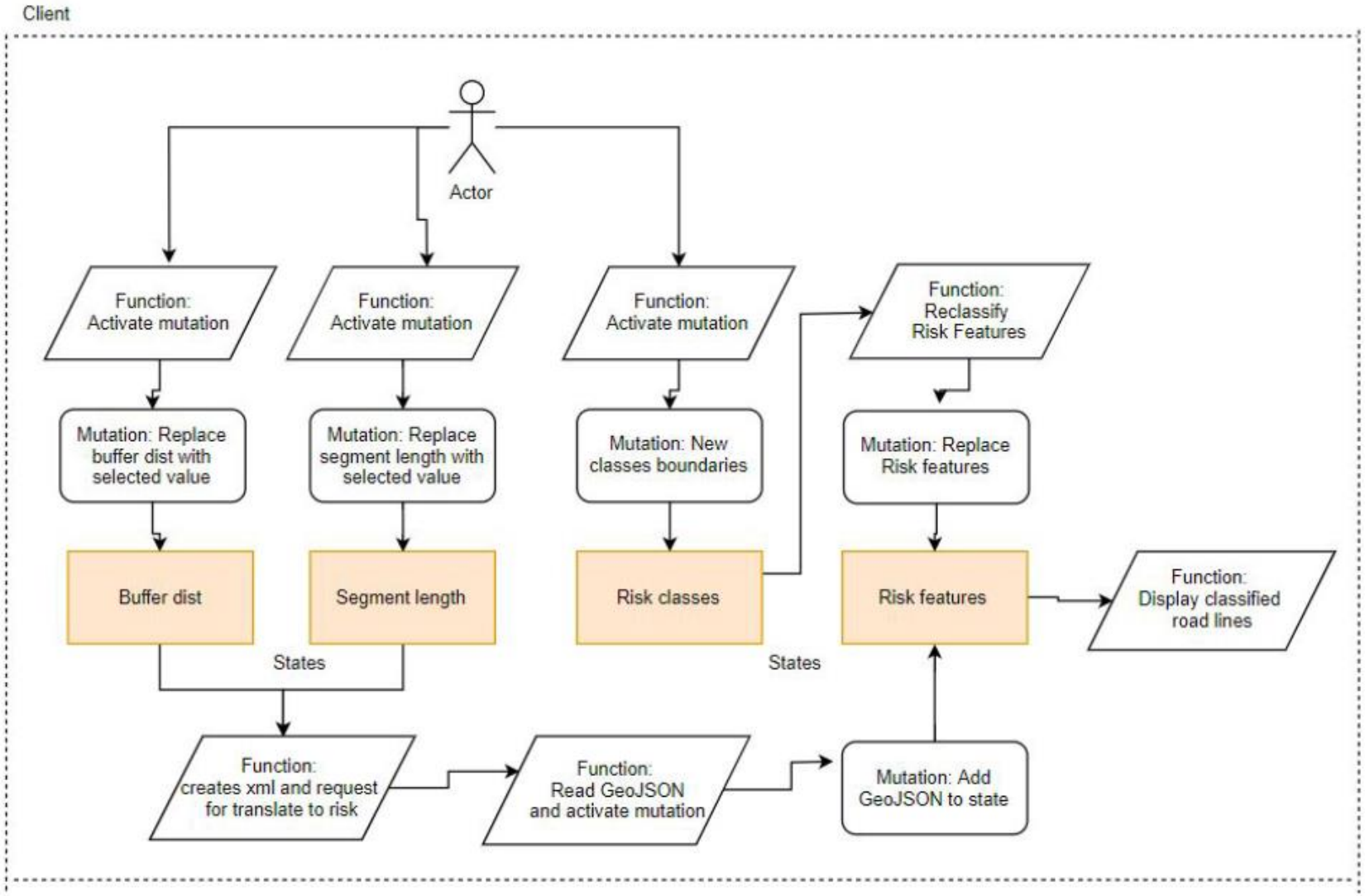
- WPS inputs:
  - Buffer distance (area that affects the road)
  - Segment length (e.g. to cut in smaller Pieces)
  - Identifier of the road (create the bbox)
  - Layer setup of the vulnerability map



- WPS outputs:
  - GeoJSON with feature collection of road lines (Property color, mean, max, min vulnerability value)

Implementation (1) Front end

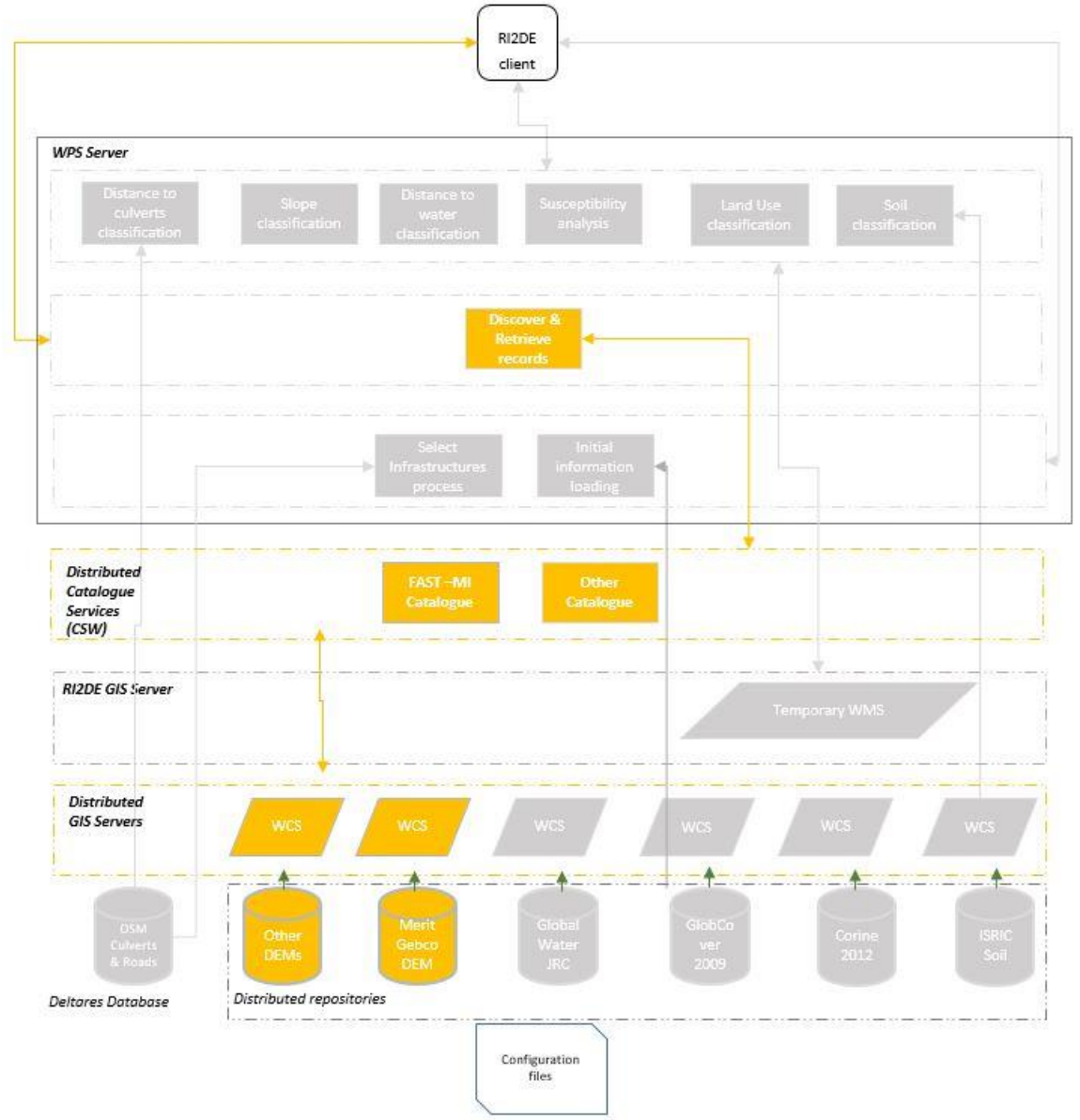
Buffer distance: 50 to 250
Segment length: 500 to 2000
Classes boundaries: 0 to 2



New Architecture

Added:

- WPS that discover & retrieves records
- Distributed Catalogue Services (CSW)
- Distributed WCS of Elevation datasets



Demonstration of the prototype

http://ri2de.openearth.eu

https://deploy-preview-51--ri2de.netlify.com/

Conclusions

Related to research questions

- The flexibility of the processes is a great problem for the tool (Only Slope and Water classification processes can accept other services. The Culverts does not accept even a services as input)
- The metadata were retrieved with the GetRecords operation
  - Not easy task to find well structured metadata records
  - Not easy task to find at all metadata records
- The VUEX framework is ideal for data applications (Store-State-Mutate)
- PyWPS 4.0.0 does not still support the GetStatus, GetResult and Dismiss operation of the WPS 2.0.0 and there are many issues until a stable a release (open conversation since 2016)
- Status supporting is not an easy task in general for these processes

Conclusions

General conclusions

- Greatest achievement was the distribute search and retrieve of records (Unique for a GIS Web decision support tool)
- Demands flexible process
  - Demands registered services with all the needed metadata
  - Demands services only in WGS84

  - Developing a WPS of the process means that it can be used by numerous of projects (OWSLib)
  - Being open source means that anyone can have access to the code and adapt it

- Standards, Implementations and organizations are not yet in line

Discussion

- Big amount of spatial data, no metadata, or no correct metadata
- WPS 2.0.2 has been released since 2015, but only ZOO-Project fully implements it. Arise question about the complexity and the state of the art technology
- When developing the XML Execute requests for the translate to risk
  - No need to know the process! Only the inputs that needs. GeoJSON data format simple and easy manipulation in the web browser

- Continue the work of others (Difficulty to understand their codes)
- Wished I have spend less time on understanding the tool, and learning the VUEX
- More time on solving the PyWPS issues

Future work…

- Usability testing
- Job monitor & Job control, e.g with ZOO-project, or with PyWPS in the future or after developing it (after all it is open source)
- Status report at every step of the processes
- WPS for CRS transformation
- GUI: Ling at every returned record that lead to the catalogue for more information
- More user friendly GUI: e.g add a manual for the tool, or a link to its wiki page. Configure everything from web browser instead of configuration files.

Questions?