

# Proposed Solution to Use Case

Mike Li

## 1. Overview

The solution for this use case aims to enhance the flood and wildfire risk assessment for large industrial sites. These sites, covering extensive areas, face varying flood risks owing to factors such as proximity to rivers and elevation differences. Additionally, they are susceptible to wildfire risks due to their potential location near forests or other high-risk zones, and the presence of flammable materials on-site. However, each record in the client's data only encompasses a single geocoded coordinate. My proposed solution is to utilise advanced geospatial analysis techniques, enabling us to extrapolate both flood and wildfire risk data across the entire industrial site from this single point. This approach will provide a more comprehensive and detailed understanding of both flood and wildfire risk across these expansive industrial areas.

The workflow and interface between different parties are detailed in Diagram 1.

- ***Flood Risk Processing*** – This workflow illustrates the processing of various layers of data, such as parcel, building, and flood, to assess flood exposure risk.
- ***Wildfire Risk Processing*** - Similarly, this workflow presents the processing of parcel, building, and wildfire data layers to gauge wildfire exposure risk.
- ***Application*** – This layer describes where the processed data is stored and how it is used by the application. The specifics include employing a cloud-based GIS database to store the processed data. The application layer interfaces with the database, offering accessibility and real-time updates on both flood and wildfire risks.
- ***User*** – This layer explains how users can obtain the necessary exposure information through a web browser or the desktop ArcGIS Pro application.

All processed results are stored in new tables titled "**RiskExposureInParcel**" and "**RiskExposureInBuilding**". These tables have a one-to-one relationship with client data and parcel tables. The development of the API and web application will directly utilize these tables, promoting efficient querying for risk assessments.

The strength of this design lies in its ability to add new risk exposures by using a similar workflow as those used for flood and wildfire risks. For example, by incorporating additional geospatial data like seismic activity from earthquakes, the workflow can be easily expanded to calculate related risk exposures. This flexibility enables the system to continually adapt and provide increasingly comprehensive risk assessments.

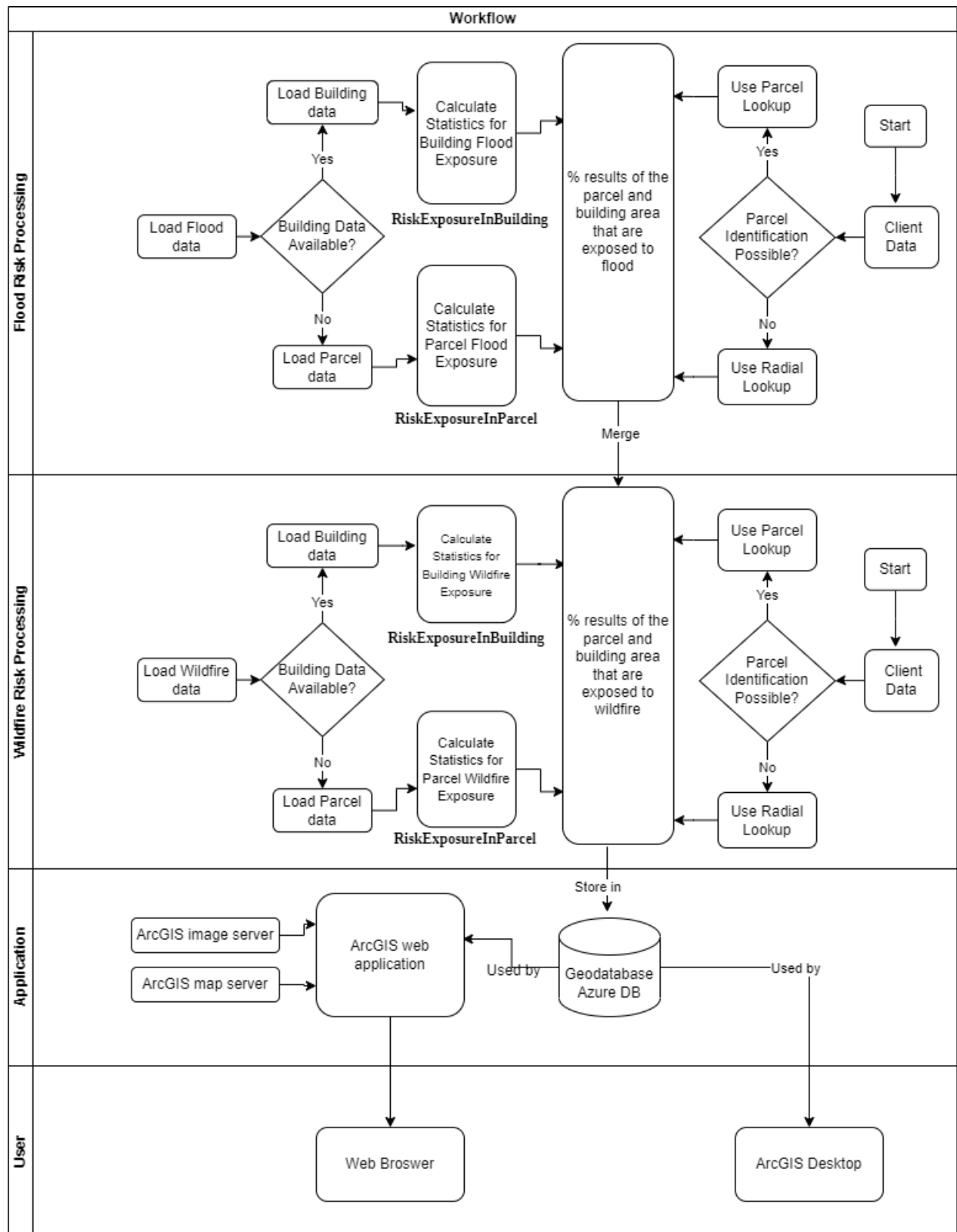


Diagram 1

## 2. Data Gathering

The first step involves collecting all relevant data. This includes geocoded client data, flood and wildfire risk data, as well as building data and parcel identification if available. From the information provided, we can identify the following types of data:

**a. Client Data:** This data, typically represented in vector format as points, primarily includes address data and associated asset values for each location. The address data is geocoded into coordinates for spatial analysis, while the asset values likely indicate the monetary value of structures and their contents at each location.

**b. Parcel Data:** This data, often represented in vector format as polygons, includes information about property boundaries, sizes, and potentially ownership details. Nationwide parcel datasets are available for the USA and France.

**c. Building Data:** This data, also usually represented in vector format as polygons, can include attributes such as building height, usage (e.g., residential, commercial), construction materials, and year of construction.

**d. Flood Return Period Data:** Presented in vector format as polygons, this data quantifies the probability and potential severity of a flood event, typically expressed in years. The data is differentiated between river floods and surface water floods, and it may yield multiple results per pixel. In this use case, return periods of 1-in-50 years, 1-in-100 years, and 1-in-500 years are considered. Consequently, each pixel could potentially receive six results — three for river floods and three for surface water floods.

**e. Wildfire Data:** Wildfire data is ingested through an API, which returns hazard level attributes.

### 3. Risk Assessment

The goal is to create a service that calculates the risk of floods and wildfires specific to each client's location. The assessment procedure involves determining the part of a property area vulnerable to these risk. When building data is available, the service should also evaluate the potential impact of these hazards on building assets.

#### Area Exposure Calculation

- Diagram 1 presents a high-level workflow for assessing flood and wildfire risk exposure.
- The workflow for flood risk exposure begins with client data. For each data point with a geocoded location, a spatial-joined parcel is obtained using a buffering method.
- In instances where parcel identification isn't possible, a radial lookup is employed. This necessitates the development of an algorithm to adjust the radial size based on the asset value. Higher asset values will correspond to larger radial radii.
- If building data is available, the process is repeated for the respective buildings.
- The workflow for wildfire risk exposure mirrors that of the flood risk, beginning with client data. For each data point with a geocoded location, a spatial-joined parcel is identified using a buffering method. The hazard level attribute within the parcel is determined via API, which then calculates the percentage risk exposure.
- If building data is available, this process is repeated for the respective buildings.

- Once the flood and wildfire hazard data have been processed, the results are stored in the '**RiskExposureInParcel**' and '**RiskExposureInBuilding**' tables (please refer to Diagram 2). These tables connect the risk exposure results with client data, parcels, or buildings (when available), creating a comprehensive dataset for future utilization, such as API and web application development.

### **Software, Libraries, Services used**

- PostgreSQL with PostGIS extension
- pgAdmin 4
- ArcGIS Pro
- ArcGIS Data Management Toolbox
- Python and ArcPy
- Geodatabase - <https://desktop.arcgis.com/en/arcmap/latest/manage-data/geodatabases/what-is-a-geodatabase.htm>
- Intersect (Analysis) - <https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/intersect.htm>
- Buffer (Analysis) - <https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/buffer.htm>
- Spatial Join (Analysis) - <https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/spatial-join.htm>
- Add Geometry Attributes - <https://desktop.arcgis.com/en/arcmap/latest/tools/data-management-toolbox/add-geometry-attributes.htm>

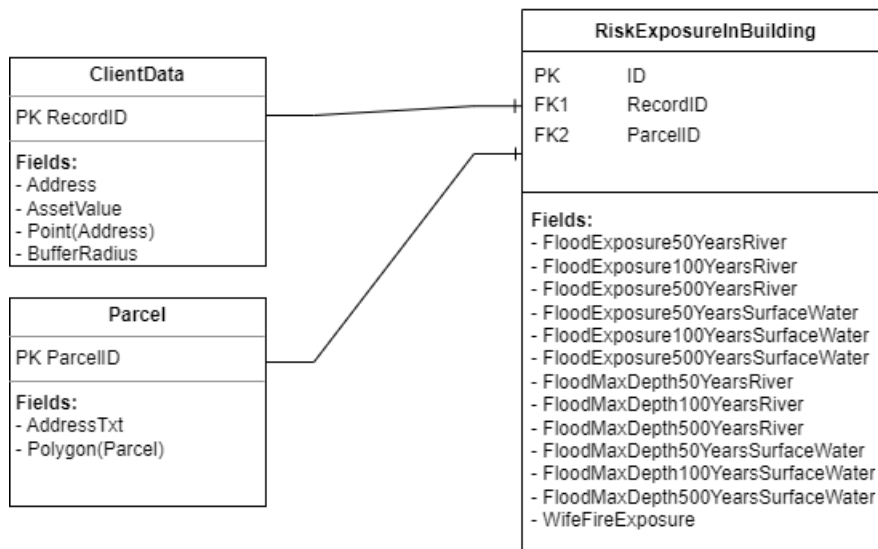
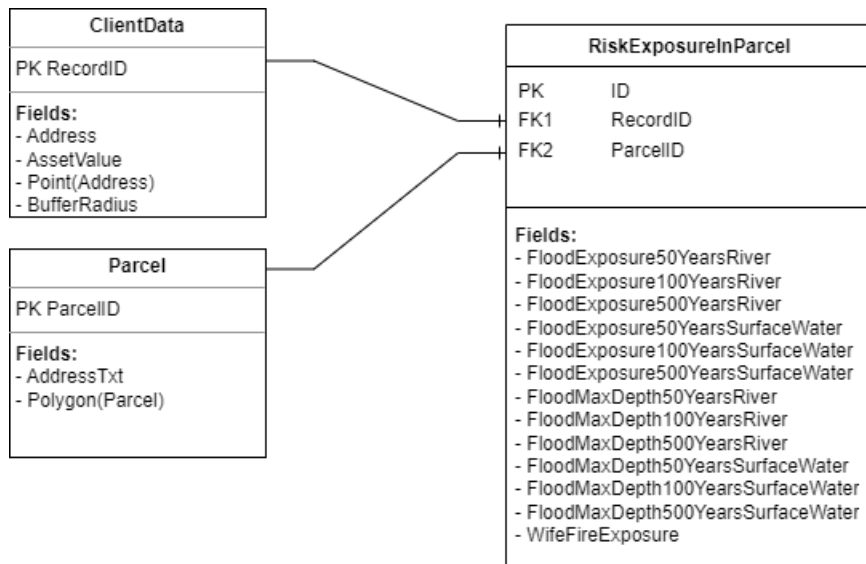


Diagram 2

Table description for **RiskExposureInParcel**.

1	ID	record unique key
2	RecordID	id in record_data
3	ParcelID	parcel id in selected_parcel_zone
4	FloodExposure50YearsRiver	percentage of overlay for periods 1: 50 years and river type
5	FloodExposure100YearsRiver	percentage of overlay for periods 1: 100 years and river type
6	FloodExposure500YearsRiver	percentage of overlay for periods 1: 500 years and river type
7	FloodExposure50YearsSurfaceWater	percentage of overlay for periods 1: 50 years and surface_water type
8	FloodExposure100YearsSurfaceWater	percentage of overlay for periods 1: 100 years and surface_water type
9	FloodExposure500YearsSurfaceWater	percentage of overlay for periods 1: 500 years and surface_water type
10	FloodMaxDepth50YearsRiver	max depth value for periods 1: 50 years and river type

11	FloodMaxDepth100YearsRiver	max depth value for periods 1: 100 years and river type
12	FloodMaxDepth500YearsRiver	max depth value for periods 1: 500 years and river type
13	FloodMaxDepth50YearsSurfaceWater	max depth value for periods 1: 50 years and surface_water type
14	FloodMaxDepth100YearsSurfaceWater	max depth value for periods 1: 100 years and surface_water type
15	FloodMaxDepth500YearsSurfaceWater	max depth value for periods 1: 500 years and surface_water type
16	WildfireExposure	percentage of overlay for risk of wildfire

Table 1

Table description for **RiskExposureInBuilding**

1	ID	record unique key
2	RecordID	id in record_data
3	BuildingID	building id in selected_parcel_zone
3	ParcelID	parcel id in selected_parcel_zone
4	FloodExposure50YearsRiver	percentage of overlay for periods 1: 50 years and river type
5	FloodExposure100YearsRiver	percentage of overlay for periods 1: 100 years and river type
6	FloodExposure500YearsRiver	percentage of overlay for periods 1: 500 years and river type
7	FloodExposure50YearsSurfaceWater	percentage of overlay for periods 1: 50 years and surface_water type
8	FloodExposure100YearsSurfaceWater	percentage of overlay for periods 1: 100 years and surface_water type
9	FloodExposure500YearsSurfaceWater	percentage of overlay for periods 1: 500 years and surface_water type
10	FloodMaxDepth50YearsRiver	max depth value for periods 1: 50 years and river type
11	FloodMaxDepth100YearsRiver	max depth value for periods 1: 100 years and river type
12	FloodMaxDepth500YearsRiver	max depth value for periods 1: 500 years and river type
13	FloodMaxDepth50YearsSurfaceWater	max depth value for periods 1: 50 years and surface_water type
14	FloodMaxDepth100YearsSurfaceWater	max depth value for periods 1: 100 years and surface_water type
15	FloodMaxDepth500YearsSurfaceWater	max depth value for periods 1: 500 years and surface_water type
16	WildfireExposure	percentage of overlay for risk of wildfire

Table 2

To demonstrate the viability of this solution, I simulated two client's records, two parcels, and several buildings within those parcels using artificial data. The flood area is represented by many of 5m square grids. The geocoded point location of one simulated client record is situated within a parcel, while the other is outside. Portions of these two parcels overlap with the simulated flood area. You can explore this scenario by downloading the provided ArcGIS project package and opening it in ArcGIS Pro. The package includes all necessary ArcPy scripts and map layer assets for processing.

To download the ArcGIS project package, please use this link  
[https://drive.google.com/file/d/1Bj1kWwZXSbKFD6Ia3-43R2icX52sx1c3/view?usp=drive\\_link](https://drive.google.com/file/d/1Bj1kWwZXSbKFD6Ia3-43R2icX52sx1c3/view?usp=drive_link)

To view the code for python scripts only, please visit this link <https://github.com/webtrackerxy/gis-check-parcel-in-flooding>

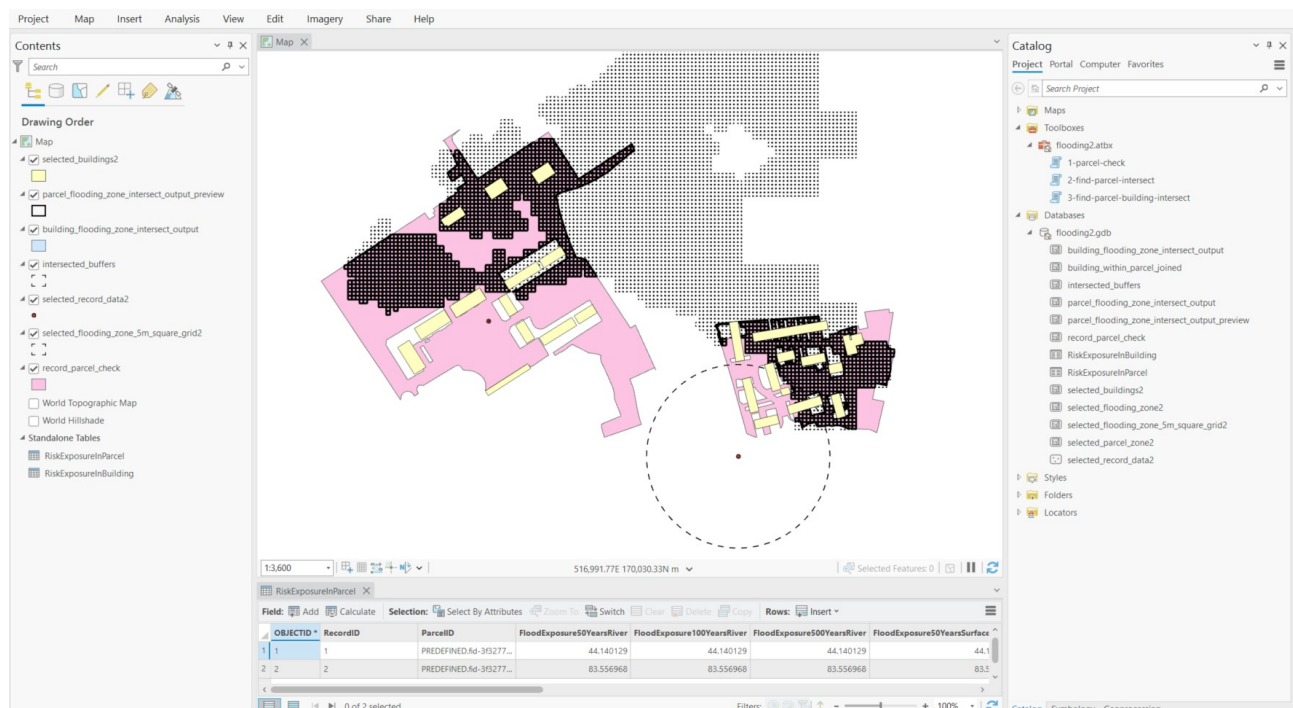


Diagram 3

Diagram 3 illustrates the project view in ArcGIS Pro. Here, each point signifies a client record, with their respective buffer zones denoted by dashed circles. The pink-colored regions represent two parcels. The buildings are represented in yellow colour. The grid represents the flood-prone area, where each block equates to a 5m x 5m square. The darker segments within each parcel indicate the portions susceptible to flooding. I used three steps involving ArcPy scripts to accomplish these tasks. You may trigger the processing in the toolboxes (refer to Diagram 4).

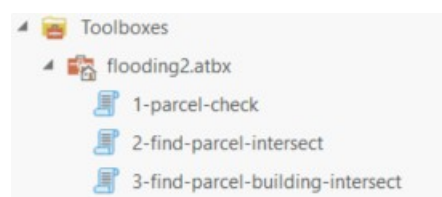
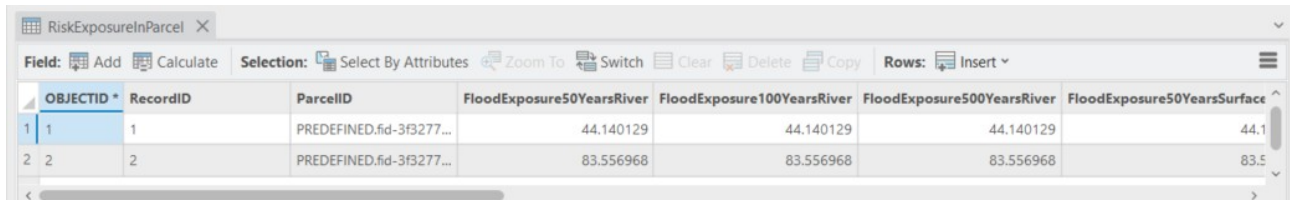


Diagram 4

**Step 1:** Run “1-parcel-check”. This process will create a new polygon layer which linked the parcel and record layers together. This polygon layer will be further used by step 2.

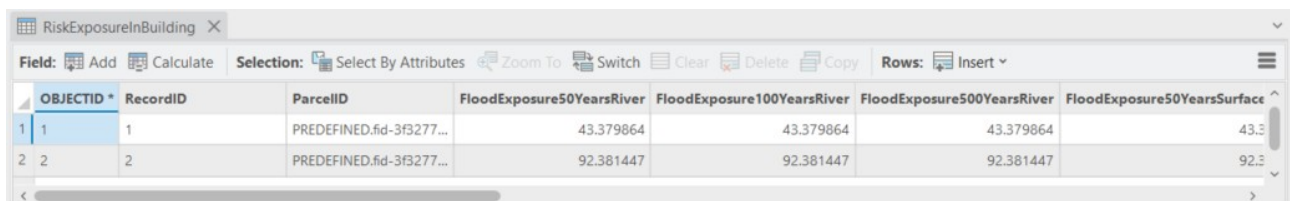
**Step 2:** Run “2-find-parcel-intersect”. This step will process the flood data and the new polygon layer generated from step 1 to get % results of the parcel area that is exposed to flood. The result will be saved to table **RiskExposureInParcel**. Open the table in ArcGIS Pro, you would see all the values of different exposure (refer to Diagram 5).



OBJECTID *	RecordID	ParcelID	FloodExposure50YearsRiver	FloodExposure100YearsRiver	FloodExposure500YearsRiver	FloodExposure50YearsSurface
1	1	PREDEFINED.fid-3f3277...	44.140129	44.140129	44.140129	44.1
2	2	PREDEFINED.fid-3f3277...	83.556968	83.556968	83.556968	83.5

Diagram 5

**Step 3:** Run “3-find-parcel-building-intersect”. This step will process the flood data and the new polygon layer generated from step 1 to get % results of the building area that is exposed to flood. The result will be saved to table **RiskExposureInBuilding**. Open the table in ArcGIS Pro, you would see all the values of different exposure (refer to Diagram 6).



OBJECTID *	RecordID	ParcelID	FloodExposure50YearsRiver	FloodExposure100YearsRiver	FloodExposure500YearsRiver	FloodExposure50YearsSurface
1	1	PREDEFINED.fid-3f3277...	43.379864	43.379864	43.379864	43.3
2	2	PREDEFINED.fid-3f3277...	92.381447	92.381447	92.381447	92.3

Diagram 6

Please note that this ArcGIS project package currently only implements the risk exposure calculation for flooding related to river and surface water, with return periods of 1 in 50 years, 1 in 100 years, and 1 in 500 years, based on client data. It does not currently account for the risk exposure related to wildfires. Furthermore, the data used in this project is simulated and does not reflect real-world scenarios. Adjustments would be necessary to accommodate real data and accurately reflect actual conditions.



## Step-by-step guide

To conduct the Area Exposure Calculation, here's a general step-by-step guide using the ArcGIS geo processing tool and language.

(i) *For those locations where parcel identification is not possible together with other Countries to use a radial lookup. The size of the radial is scaled based on the asset value at the site as we are assuming value as a rough proxy of site size.*

The following steps refer to the **Step 1 ArcPy processing script [1-parcel-check](refer to Diagram 4)**

This code checks whether geocoded address points (in the selected\_record\_data layer) fall within a given parcel polygon (in the selected\_parcel\_zone2 layer). It uses a buffering approach to account for potential inaccuracies in the geocoded addresses. If an address point initially doesn't intersect a parcel polygon, the buffer around the point is increased and the intersection checked again. The results of successful intersections are stored in the record\_parcel\_check layer.

Code link: [https://github.com/webtrackerxy/gis-check-parcel-in-flooding/blob/main/radial\\_check.py](https://github.com/webtrackerxy/gis-check-parcel-in-flooding/blob/main/radial_check.py)

- It specifies the layers to work with, which are record\_data (a point layer containing geocoded addresses) and selected\_parcel\_zone2 (a polygon layer representing parcels).
- The script creates an empty polygon feature class named intersected\_buffers which will later store the polygons resulting from the buffer operation. It also adds several fields to this new layer for storing attributes of the buffer polygons.
- The script then initiates an empty list (rows\_to\_insert) to store information about the buffers that successfully intersect with the parcel polygons.
- The script proceeds to loop through each feature (address point) in record\_data using an UpdateCursor.
- For each address point, it creates a buffer of a defined radius around the point. It then checks if the buffered point intersects with any of the parcel polygons.
- If there is no intersection, the buffer radius is increased (by 30% of the original radius) and the intersection operation is attempted again.
- If the buffer intersects with a parcel polygon, the buffer is converted to a single part (in case it is multi-part) and the attributes for that buffer are added to the rows\_to\_insert list.
- Once all address points have been checked and relevant buffers stored in rows\_to\_insert, the script loops through this list and adds each record as a new row in intersected\_buffers.
- The script then performs a spatial join operation between the original polygon layer (selected\_parcel\_zone2) and the newly populated intersected\_buffers layer, creating a new layer named record\_parcel\_check. This layer contains information from both input layers for features that share the same location.
- Finally, the script cleans up the in-memory workspace to free up resources, deleting temporary layers created during processing.

(ii) % of results of the parcel area that is exposed to flood for each of the 3 return periods for each of the flood perils

(iii) Maximum depth for each flood return period for each of the flood perils

The following steps refer to the **Step 2 ArcPy processing script [2-find-parcel-intersect](refer to Diagram 4)**

This code calculates the risk exposure related to parcels. It intersects flood risk data with parcels, calculates the flood risk for each parcel for different flood types and return periods, and then summarizes this information in a new table.

Code link: [https://github.com/webtrackerxy/gis-check-parcel-in-flooding/blob/main/find\\_zone\\_intersect.py](https://github.com/webtrackerxy/gis-check-parcel-in-flooding/blob/main/find_zone_intersect.py)

- Set up variables: The input features to be intersected are defined. They are 'record\_parcel\_check', which contains parcel data, and 'selected\_flooding\_zone\_5m\_square\_grid2', which contains flood zone data. The output of the intersect operation will be stored in 'parcel\_flooding\_zone\_intersect\_output'.
- Define flood types and periods: The flood types and periods to be analyzed are defined. Flood types include 'surface\_water' and 'river'. Flood periods include '1: 50 year', '1: 100 year', '1: 500 year'. These will be used to filter the intersected polygons later on.
- Unique Parcel IDs: It collects all unique parcel IDs from the 'record\_parcel\_check' layer.
- Loop through parcels: For each unique parcel ID, it selects the parcel from the 'record\_parcel\_check' layer and intersects this parcel with the flood zone layer. It calculates the total area of this parcel for later use.
- Calculate exposure and depth: It calculates the percentage of the parcel that overlaps with flood zones and the maximum depth of flooding for each combination of flood type and period. This is done by creating a feature layer of intersected polygons for the specific flood type and period, calculating the total area of these polygons, and comparing it to the total area of the parcel. It also keeps track of the maximum flood depth for each type and period.
- Result compilation: The results are compiled in a dictionary that includes the parcel ID, flood type, flood period, flood exposure percentage, and maximum flood depth. This dictionary is then added to a list.
- Table creation and data insertion: It creates a new table called 'RiskExposureInParcel', defines the fields and their types, and then inserts the data from the result list into this new table.

(iv) % result of the parcel exposed to Wildfire and the returned single numeric attribute

I did not implement this task as there is no specification for the wildfire API. But the steps would be similar to the (i) and (ii).

The code calculates the wildfire risk exposure related to parcels. It intersects wildfire risk data with parcels, calculates the wildfire risk for each parcel to get the hazard level attributes then summarizes this information in a new table.

- Set up variables: The input features to be intersected are defined. They are 'record\_parcel\_check', which contains parcel data, and 'selected\_wildfire\_zone\_5m\_square\_grid', which contains wildfire zone data. The output of the intersect operation will be stored in 'parcel\_wildfire\_zone\_intersect\_output'.
- Loop through parcels: For each unique parcel ID, it selects the parcel from the 'record\_parcel\_check' layer and intersects this parcel with the wildfire zone layer. It calculates the total area of this parcel for later use.
- Calculate exposure: It calculates the percentage of the parcel that overlaps with wildfire zones. This is done by creating a feature layer of intersected polygons, calculating the total area of these polygons, and comparing it to the total area of the parcel.
- Result compilation: The results are compiled in a dictionary that includes the parcel ID, wildfire exposure percentage. This dictionary is then added to a list.
- Table creation and data insertion: It update the table called 'RiskExposureInParcel', defines the fields and their types, and then inserts the data from the result list into this new table.

(v) if we have the building data, the flood and Wildfire impact to the 2D area of the buildings on the site using the asset values. We would only need the % of the area exposed from all the buildings within the parcel.

(vi) If the building data is available this would be in addition to the parcel data. Whilst generally all asset values are contained within buildings we may also insure assets outside the building eg stock of cars stored outside of a car manufacturing site.

The following steps refer to the **Step 3 ArcPy processing script [3-find-parcel-building-intersect] (refer to Diagram 4)**

This code calculates the flood risk exposure of each building based on their intersection with different flood zones and stores the result in a newly created table. Each record in the table corresponds to a building, and the various fields in the record provide details about the flood risk exposure for that building.

Code link: [https://github.com/webtrackerxy/gis-check-parcel-in-flooding/blob/main/find\\_parcel\\_building\\_intersect.py](https://github.com/webtrackerxy/gis-check-parcel-in-flooding/blob/main/find_parcel_building_intersect.py)

- Define input features as parcels and flood zones layers.
- Execute a spatial join operation between building and parcel layers. This operation assigns each building a corresponding parcel ID based on their spatial intersection. In other words, it finds out which parcel each building belongs to.

- Retrieve all unique building IDs (along with their corresponding parcel ID) from the spatially joined layer created in the previous step.
- Define flooding periods ("1: 50 year", "1: 100 year", "1: 500 year") and flood types ("surface\_water", "river").
- Iterate over each unique building ID, and for each building, perform the following operations:
  - Create a subset (Feature Layer) from the joined building layer where the building ID matches the current one in the loop.
  - Intersect the selected building with the flood zone layer.
  - Calculate the total area of the selected building.
  - For each combination of period and type, calculate the percentage of the building area overlaid by the flood zone and the maximum flood depth. These values are then stored in dictionaries.
  - Merge the exposure and depth results into a single dictionary per building, and append it to a list that will hold the data for all buildings.
- Once all the buildings have been processed, create a new table named 'RiskExposureInBuilding' in the GeoDatabase, with specified fields relating to flood exposure and maximum depth.
- Delete any existing data from the newly created table.
- Finally, insert the flood risk results into the new table. This risk information related to the building's intersection with different flood zones. Each record corresponds to percentage of building's risk exposure within a parcel.

## 4. API Development

Build an API to deliver client results quickly (under 1 second per location) and to allow for integration with external systems.

### Current development and production environment

- ArcGIS Pro (Advanced Desktop): for detailed spatial analysis and data visualization.
- ArcGIS Portal: to provide a central place for storing and managing maps, apps, and other geographic assets.
- ArcGIS Enterprise (Cloud platform): for deploying GIS in the infrastructure.
- Geodatabase: for storing and managing geographic information.
- Image Server: for serving large volumes of imagery and raster data.
- Azure DB: for cloud storage of the data.

Given the above resources, it considers leveraging ArcGIS Server and ArcGIS Enterprise in creating a GeoProcessing Service. GeoProcessing services are versatile in that they can perform any GIS operation and return results in many formats, including GeoJSON. It can develop a script in ArcGIS Pro, publish it to the ArcGIS Enterprise as a GeoProcessing service, and then use it as the API endpoint.

### Here is a proposed solution

- **Develop GeoProcessing Script in ArcGIS Pro:** To create a script tool using Python (ArcPy). This script will query the "RiskExposureInParcel" and "RiskExposureInBuilding" tables in the PostgreSQL database and return the results as GeoJSON. Referring to Diagram 2, each table of "RiskExposureInParcel" and "RiskExposureInBuilding" has one to one relationship to tables of client data and parcel. Input any attribute in client data (e.g. id, name, asset value), it is possible to query their linked parcel and the risk percentage of exposure.

Simple example of the code as following:

```
import psycopg2
import json
import arcpy

# DB connection
conn = psycopg2.connect(
    dbname="db_name",
    user="db_user",
    password="db_password",
    host="db_host"
)

# Input parameters
record_id = arcpy.GetParameterAsText(0)

# Cursor to execute queries
cur = conn.cursor()

# SQL to fetch data from RiskExposureInParcel, ClientRecord, and Parcel
sql_parcel = f"""
SELECT
selected_record_data2.name, RiskExposureInParcel.floodexposure50yearsriver,
selected_parcel_zone2.geom FROM RiskExposureInParcel
INNER JOIN selected_record_data2 ON RiskExposureInParcel.recordid = selected_record_data2.id::text
INNER JOIN selected_parcel_zone2 ON RiskExposureInParcel.parcelid = selected_parcel_zone2.gml_id::text
WHERE RiskExposureInParcel.recordid = '{record_id}';
"""

sql_building = f"""
SELECT selected_record_data2.name, RiskExposureInParcel.floodexposure50yearsriver,
selected_parcel_zone2.geom FROM RiskExposureInParcel
INNER JOIN selected_record_data2 ON RiskExposureInBuilding.recordid = selected_record_data2.id::text
INNER JOIN selected_parcel_zone2 ON RiskExposureInBuilding.parcelid = selected_parcel_zone2.gml_id::text
WHERE RiskExposureInBuilding.recordid = '{record_id}';
"""

try:
    # Execute the SQL
    cur.execute(sql_parcel)
    result_parcel = cur.fetchall()

    cur.execute(sql_building)
    result_building = cur.fetchall()
except Exception as err:
    print("Error executing SQL: {0}".format(err))

# Format the result as GeoJSON
geojson_parcel = {
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "properties": properties + ", " + str(floodexposure50yearsriver),
            # Replace with actual geometries
            "geometry": {"type": "Polygon", "coordinates": coordinates}
        } for properties, floodexposure50yearsriver, coordinates in result_parcel
    ]
}

geojson_building = {
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
```

```

        "properties": properties + ", " + str(floodexposure50yearsriver),
        # Replace with actual geometries
        "geometry": {"type": "Polygon", "coordinates": coordinates}
    } for properties, floodexposure50yearsriver, coordinates in result_parcel
    ]
}

# Print out the GeoJSON
arcpy.AddMessage(json.dumps(geojson_parcel, indent=2))
arcpy.AddMessage(json.dumps(geojson_building, indent=2))

# Close cursor and connection
cur.close()
conn.close()

```

- **Publish the Script as a GeoProcessing Service:** After the script has been tested in ArcGIS Pro, it can be published to ArcGIS Enterprise as a GeoProcessing service. Once published, it will be accessible as a RESTful web service. This means that it can receive HTTP requests from external systems.
- **Optimize Performance:** Each table in the 'RiskExposureInParcel' and 'RiskExposureInBuilding' has a one-to-one relationship with the client data and parcel tables. The process is purely database query-based and does not involve any other operations. Therefore, the response rate is expected to be very fast. However, if the response time of under one second per location is not achieved, improvements may be necessary. These could involve optimizing SQL queries, ensuring the database is properly indexed, setting up the GeoProcessing service for high demand, and potentially running the service on more powerful hardware. Additionally, the service may need to be monitored and resources adjusted as necessary, based on demand.

## 5. Data Visualization

Implement a GIS-based system to visualize the risk results on a map. This should allow for editing of parcel/radial, adding a building, and connecting an associated parcel.

ArcGIS Experience Builder [<https://www.esri.com/en-us/arcgis/products/arcgis-experience-builder/overview>] provides a quickly develop web experiences using drag-and-drop components alongside configurable settings.

### a) Visualize the risk results on a map:

Experience Builder supports adding maps and connecting to external services. It supports GeoJSON from an API endpoint and make it as a feature layer. It is possible to create a widget to add this layer to a web map.

### b) Allow for editing of parcel/radial, adding a building, and connecting an associated parcel:

Experience Builder comes with built-in widgets that facilitate feature editing on a map.

- The layer in the ArcGIS Online or Portal map should be set to be editable. This can be done within the layer's settings.
- The map should be added to the Experience Builder app.
- An Edit widget should be added to the app. This widget should be configured to interact with the newly added map widget.

- The Edit widget should be configured to dictate the types of edits that can be carried out.
- This can empower users to add, delete, and update features, including their geometry and attributes.

Overall, ArcGIS Experience Builder provides a powerful platform for creating interactive web applications, but for more complex or customized functionality, there may be a necessity to explore its Software Development Kit (SDK), or the development of a custom application utilizing the ArcGIS API for JavaScript might be considered.

## 6. Non ESRI based solutions and advantages

While ESRI products like ArcGIS Pro, Portal, and Enterprise offer a rich suite of tools for GIS analysis and visualization, there are open-source alternatives available which might be useful.

- **QGIS:** An open-source desktop GIS software. It is similar to ArcGIS Pro and provides a wide range of tools for spatial analysis and data visualization. QGIS also supports a variety of plugins that can extend its functionality.
- **PostgreSQL with PostGIS extension:** The open-source relational database system currently in use has the capability to store and query spatial data effectively. This system is extended by PostGIS, which adds support for geographical objects. Thus, it presents itself as a suitable alternative to the proprietary Geodatabase system.
- **GeoServer:** An open-source server for sharing geospatial data. It could serve as an alternative to ArcGIS Enterprise and Image Server for certain uses. GeoServer allows for serving vector and raster data over the web using open standards like WMS, WFS, WCS, WPS, and REST.
- **pgAdmin 4:** This open-source PostgreSQL management tool is already being utilized. It serves as a comprehensive design and management interface for PostgreSQL databases.
- **GDAL/OGR:** A library for translating and processing raster and vector geospatial data formats. These libraries provide functionalities equivalent to some aspects of ArcGIS Data Management Toolbox.
- **Leaflet or OpenLayers:** Open-source JavaScript libraries for interactive maps on the web. They could serve as alternatives to ArcGIS JavaScript API for building web mapping applications.

### Advantages of using open-source tools

- **Cost:** Open-source software is free to use, distribute, and modify.
- **Flexibility:** Open-source software can be modified to suit the specific needs.
- **Interoperability:** Open-source software often adheres to open standards making it easier to integrate with other software.

- **Community Support:** Most open-source projects benefit from active communities who can offer support and create new tools and functionalities.

Please note that while open-source tools can offer many benefits, they also require more setup and configuration than proprietary software. Also, while the community support can be great, they might not have the same level of customer service as proprietary software. Therefore, choosing between proprietary and open-source tools often depends on the specific requirements and resources of the project. Therefore, it is not preferred due to existing built environment.

## 7. Timeline to deliver the service

	Item	Description	Required time to complete
1	Data processing	Develop a geoprocessing workflow in ArcGIS Pro that extracts risk of exposure in <i>selected</i> client's parcels or buildings. This development process includes stages of testing, quality check, and final release to production.	3 weeks
3	API Development	Develop a GeoProcessing service as an API on the ArcGIS Enterprise platform to provide access to exposure attributes from client data. This development process includes testing, quality control, and final release to production.	1 week
4	Web App Development	Utilize ArcGIS Experience Builder to develop a front-end web application. This web application will employ APIs and data sourced from the data processing workflow. The development process includes testing, quality control, and eventual release to production.	3 weeks

Given the complexity of the task, a realistic timeline might be within 2 months. This accounts for the time needed to gather and integrate data, develop the risk calculation service and API, test the system thoroughly, and make necessary adjustments based on test results.

Timeline for implementation would also depend on various factors including the existing infrastructure, data size and complexity, technical skills of the team, and more.