Coastal Erosion from Space

Web Services Specification

Ref: SO-RP-ARG-003-055-008

Customer: ESA

Contract Ref.: ESA/AO/1-8758/16/NL/PSI-LG

This page is intentionally left blank

Signatures

	Name	Company or Institute	Date
Prepared by	Steve Emsley	ARGANS	06/02/2020
Authorised by	Martin Jones	ARGANS	06/02/2020

SIGNATURES ---- **1**

1 EXECUTIVE SUMMARY ---- **2**

2 PROCESSOR ARCHITECTURES ---- **3**

3 DATA-DRIVEN CRUD SERVICE ---- **4**

4 MAP-BASED LOCATION SERVICE ---- **5**

5 DATABASE DESIGN ---- **6**

6 PRODUCT SELECTION ---- **6**

7 NAMING CONVENTION ---- **7**

8 FILE SYSTEM ORGANIZATION ---- **9**

9 METADATA ---- **10**

10 INFRASTRUCTURE ---- **11**

11 APPLICATION PROGRAMMING INTERFACE (API) ---- **12**

12 POSTPROCESSING ---- **12**

13 CONCLUSION ---- **13**

1 Executive Summary

This is a response to a RID to:

Define and prototype the web interface in order to distribute the products and to visualise them according to users' need

Progress of the UI can be accessed via the ARGANS development server:

https://erosion.argans.co.uk/

USER: webdev

PASS: 4RGw3bd3v

This document is live i.e. it will evolve during Phase 2 based on user requirements and design/implementation development.

The Coastal Change User Interface will evolve throughout the project and after feedback from users. The deployment options are to incorporate the coastal change web services and processing within the AdwäisEO infrastructure, in close network proximity to the data source.

The Phase 2 web service will consist of a standard client-server 3-tier architecture with a REST framework.

The critical path is:

- Define ontological file hierarchy
- Apply file naming convention
- Describe & define metadata requirements
- Build & populate database
- Implement a Python DJANGO RESTful web service
- Add web Interface for database access / display records
- Add any post-processor e.g. re-formatter
- Test on development environment
- Deploy to production environment

The choice for framework will be Python Django as being a high-level web framework is good for rapid development and clean, pragmatic design. It provides a high-level vocabulary for, for instance, processing forms and authentication.

The requirements, specification, design and implementation may change but the overview presented provides flexibility. It is a layered design with loose coupling between components using Agile methods and rapid prototyping with a dedicated user group.

2 Processor Architectures

Deployment to meet the feasibility approach during Phase 1 was distributed across three processing centres and 5 sites; using desktop computers, virtual servers and cloud 'infrastructure as a service' which provided a R&D environment to develop algorithms and process test data. Each processor requires some operator intervention, usually scene selection, and the challenging objective is to provide an end-to-end processing chain providing user value products based on pre-existing data, either from an online repository or uploaded by a user definition. A main objective, within the time frame of Phase 2, is to provide a Pilot Service Demonstration – a user Interface (UI) & web application to provide partner users (and their wider stakeholder community) with access to data and products applicable to their area of interest.

At the very top-level the web service will adhere to the client-server model structure, specifically the webserver will serve web pages to the user client.

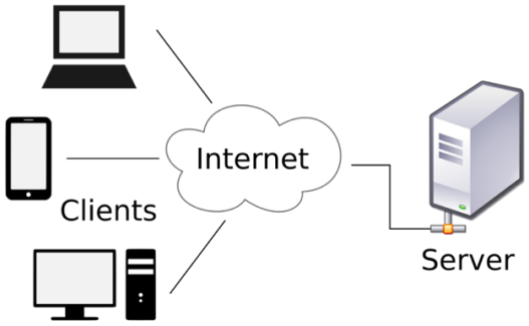

Figure 1 : Web Service Client / Server Model

The Phase 2 web service will consist of a standard client-server 3-tier architecture with a REST framework. The three logical tiers are the Presentation tier, the Business Logic or Application tier and the Data Store tier.

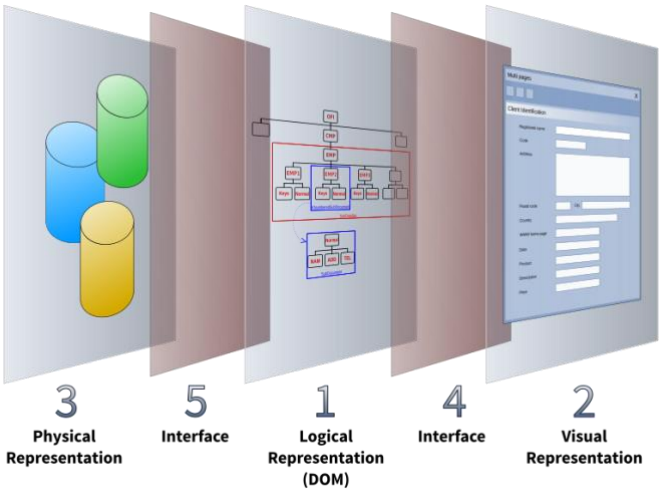

Figure 2 : Logical Structure of 3-tier / 3-layer Architecture

This architecture and framework are FOSS (Free Open Source Software), and using a Component Based Architecture (CBA) will not be included in this specification as considered and used as 'black boxes' i.e. they are acquired "off the shelf" as fully functioning capabilities.

The web application required for Phase 2 is to provide a **view** of the available data store that simplifies access to data products already processed and available within a data storage **model.** An option for the Presentation Layer is the MVC (Model-View-Controller) Design Pattern where the controller provides the logic that mediates in the interaction and **controls** how the model is viewed, i.e. how the user access data.

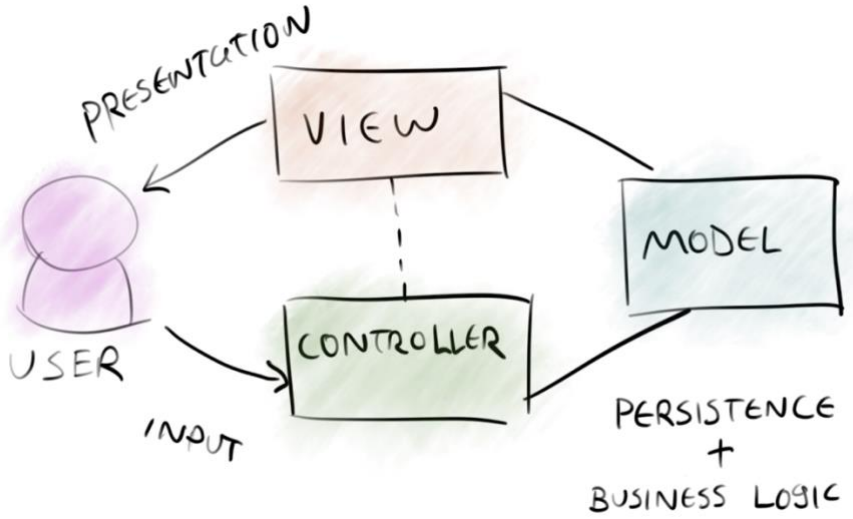

Figure 3: Basic Model-View-Controller Design Pattern

MVC is a common pattern used to provide multiple consistent views of the same underlying data. Perhaps over-engineered to fulfil the requirements the Pilot Service demo, which is to enable users to search and retrieve data from a defined set of locations and times, i.e. a CRUD service (Create-Retrieve-Update-Delete). Best practice, especially security-awareness, and to apply "higher-level logic" a service layer is often used to avoid allowing direct-access to the repository.

Deployment

AdwäisEO are providing storage space for the input/output data and will provide infrastructure to host the web services. Initially the web server & application server will be maintained on one Virtual Machine, or Docker container, and a database server on another virtual machine. The advantage in using a Docker container for the web layer is that it is easily scalable if required.

3 Data-driven CRUD service

The solution proposed is the usual CRUD basic functions of persistent storage. All but Retrieval will be administrator-only functions.

Read/Retrieve

READ procedures read the data based on input parameters. Similarly, RETRIEVE procedures grab records based on input parameters.

The service is data-driven since the repository is a finite resource; essentially for each test site ***R*** there exists ***N*** products for ***T*** dates.

$$\forall\, q \;\longmapsto\; \exists\, q_{0..n} \in Q \;\mid 0$$

Where either the query string triggers a match(s) within the data repository, or it does not. To ensure that the set of responses to a query string is not zero the user needs to be guided to select from what data holdings are available, not to enter search criteria that result in the empty set.

The functions create, update and delete are available to the database administrator and to processing scripts but are not considered components of the UI in Phase 2.

4 Map-based Location service

The Google Map API is used to display available data sites as icons on a scalable world map as seen on https://coastalerosion.argans.co.uk/ and shown here:

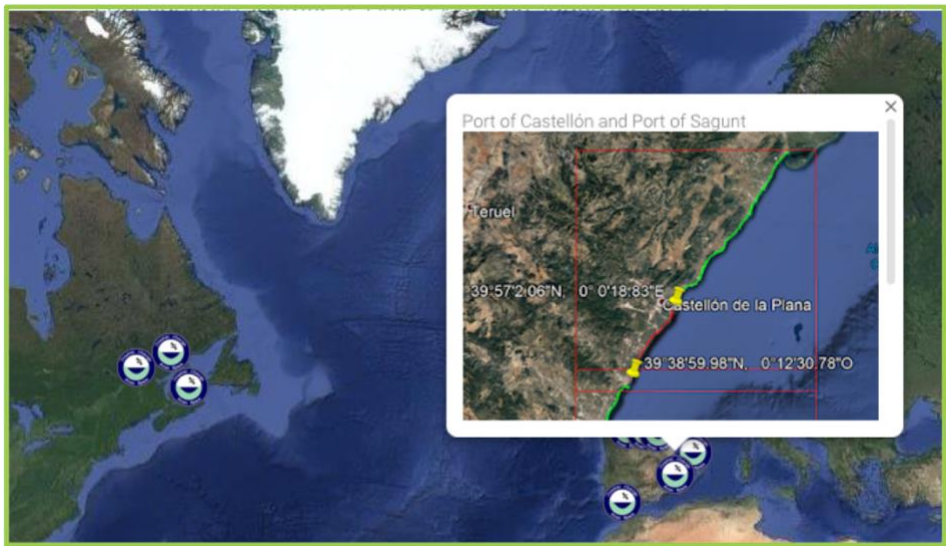

Figure 4 : Areas of Interest (AOI) Browser

The JavaScript used in the demonstrator is hard coded as the set of test sites is small. It would be trivial to extract the markers by a database lookup. The popup info window content will evolve, for instance with database access a summary of data holdings categorised by type e.g. waterline, shoreline etc. and date range could be displayed.

The map includes a clustered display that groups nearby sites according to zoom level. And with only 20 sites clustering has been disabled. And since the number of test sites are small and discrete there is no significant gain in enabling selection of data sites using a stretchable bounding box.

5 Database Design

The Model in MVC is the data source, a combination of a database and filesystem. These both combine to become a data repository, which could be scaled up by assigning additional infrastructure if demand or capacity exceed planned expectations.

A database design would associate each processed product file with the spatial and temporal coordinates at acquisition, or range if a composite from multiple input images, product descriptors and site information. A minimal schema would be:

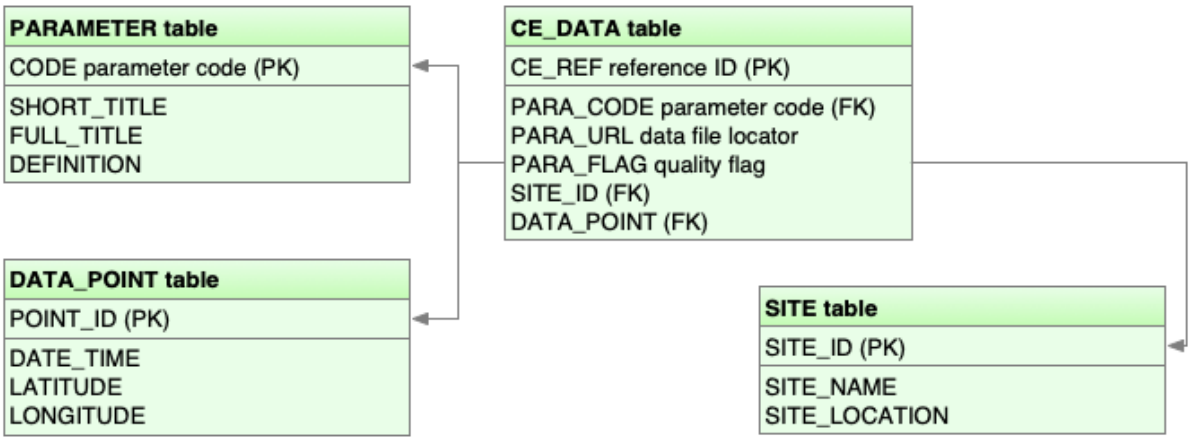

Figure 5 : Top-level Database Table Structure

The database schema will evolve to capture all information that may be needed to satisfy all user search queries. The dynamic content of the site pages, that are linked in the map info popup, use the database to populate a static template to provide the user with feedback and defaults when entering search criteria.

Implementation

The database server will be hosted on a virtual machine and the first option is to use Postgres with postGIS plugins, this is essentially a relational database management system (RDBMS) with geospatial information systems (GIS) bindings. However, in a previously developed web service both Postgres and MongoDB, a NoSQL database, were provided as options and that may be considered for the Coastal Erosion service.

6 Product Selection

In the most basic implementation, each site-specific page will contain a form which will allow the user to enter a start and end date and a product type. This will trigger an SQL query and display a list of hyperlinks to the URLs of data products that match the query.

Since for some sites usable satellite imagery may be scarce the user shall be provided with feedback identifying visually using a calendar motif in what year/month/day and how many data products are available in the repository.

A catalogue will be provided on each site-specific page offering an expandable view of the filesystem to enable direct access to the expert user.

7 Naming Convention

Filenames must be searchable by Regular Expression (RE) and selectable by glob i.e. wildcard matching. This mandates a strict naming convention.

Naming Convention (version 1.4)

Product names will conform to the schema:

`CE_<startdate>_<type>_<category>_<level>_<bbox>_<qualifier>{_<enddate>_}_<proc_date>.<extension>`

where

- The filename shall contain a 2-character code that identifies it as a product of the Coastal Erosion Project. i.e. CE
- The **<startdate>** shall contain a reduced accuracy ISO 8601 format date in Zulu time i.e. 12-character date / time in the format YYYYMMDDhhmm e.g. 201910161230
- The filename shall contain a product **<type>** designator from the list:

Table 1 : Naming Convention - Product Types

Type Code	Description	Notes
WL	Waterline	OPT / SAR
SL	Shoreline	Datum-based
SF	Seafront	Feature-based
LL	Littoral Line	Feature-based
DL	Debris Line	Feature-based
BT	Bathy-topography	OPT / SAR or VHR
BP	Beach Profile	Bathy-topography derived
LC	Land Cover	Feature-based
ER	Erosion Rate	Rate of change of an observable feature
SR	Sediment Rate	Volumetric rate of change of sediment
CR	Co-Registered	Co-registered image file

- The **<category>** shall contain a 2-character code identifying the processor category from the list:

Table 2: Naming Convention - Product Categories

Category Code	Description
OB	Observation-based
DB	Datum-based
FB	Feature-based
WF	Wavefield

Category Code	Description
MB	Model- Based[1]
MX	Mixed[2]

- The **<level>** shall contain a 2-character code identifying the processing level of the syntax L2 | L3 | L4, where L2 is a single observation, L3 is a composite product from the same sensor e.g. a timeseries, and L4 is a fusion of composite product from multiple sensor-based and/or model-derived products.
- The **<bbox>** contain the co-ordinates of the bounding box of the feature contained, i.e. for a shape file of the enclosing polygon, in sexagesimal notation to an accuracy of 1 second, representing 30 metres at the Equator, formatted as ddmmss{ N | S | E | W} of the lower-left latitude and longitude and upper-right latitude and longitude, separated by an hyphen and using the EPSG:4323 (WGS 84) co-ordinate reference system (CRS).

For example, 504401N001429E-511858N012212E designates a product spanning the area of interest from Beachy Head Lighthouse to Ramsgate Harbour.

- The product **<qualifier>** codes are identified as:

Table 3 : Naming Convention - Product Qualifiers

Product Type	Qualifier
WL (waterline)	S2A/B (Sentinel 2 MSI) L5 (Landsat 5) L8 (Landsat 8) S1A/B (Sentinel 1 SAR)
SL (shoreline)	MHWS (mean high water springs) MLWN (mean low water neaps) MSL (mean sea level)
SF (seafront)	S2A/B (Sentinel 2) L5 (Landsat 5)
BT (bathy-topography)	S2A/B (Sentinel 2 MSI) L5 (Landsat 5) L8 (Landsat 8) S1A/B (Sentinel 1 SAR) WV1-4 (WorldView) PL1A/B (Pléiades) GEO1 (GeoEye 1)
all	MX (multiple sensors)

- If a time series the **<enddate>** shall contain a reduced accuracy ISO 8601 format date in Zulu time i.e. 12-character date / time in the format YYYYMMDDhhmm.
- The **<procdate>** shall use the reduced ISO 8601 date format, YYMMDD, and is required in case there are multiple versions of the same product.

[1] A 'model-based' product is one using multiple inputs and based on a model of sediment flux and/or erosion rate.

[2] A MX designator is used if the final product merges several intermediaries of different category.

- All codes will be separated by underscore, there will be **no white space and no special characters** like ' " * # $ / { ([
- All file names will end in a filename extension and all files that are the same type will have the same 3-character lowercase file extension.

CE_201910161210_SL_DB_L2_504401N001429E-511858N012212E_MHWS_200311.shp

a shape file datum-based shoreline of the Kent coast based on the mark of high tide acquired on the 16th October 2019 at 12:10

Note the reason for the selection of the naming convention schema defined above is that product files stack in a list with all important product characteristics at the same fixed length boundary, which assists both human selection and simplifies selection by globbing and manipulation by Regular Expression.

CE_201910161210_SL_DB_L2_504401N001429E-511858N012212E_MHWM_200311.shp

CE_201910161210_WL_OB_L2_504401N001429E-511858N012212E_S2_200311.shp

CE_201910161210_BT_WF_L2_504401N001429E-511858N012212E_WorldView1_200311.shp

CE_202010161210_SL_DB_L3_504401N001429E-511858N012212E_MHWM_202010161210_200311.shp

CE_201910161210_BT_MX_L4_504401N001429E-511858N012212E_MX_202010161210_200311.shp

8 File System Organization

The first iteration of the data distribution service was using SFTP to connect and browse the data provided under a folder structure that first differentiates between input and output and then, for output, sorts data first by area of interest, then data type, and then year and month.

{input | output} → AOI → datatype → YYYY → MM

One purpose of the file naming convention was to ensure that if stored within the same folder product types would be uniquely distinguishable and selectable. If co-registered input images are named according to the convention there is no need to separate input from output, as co-registered images are an output of the pre-processing. With the recommended file system organization all data products are browsable by AOI and then by date, with all products are the same area and day all available in the same folder.

AOI → YYYY → MM → DD

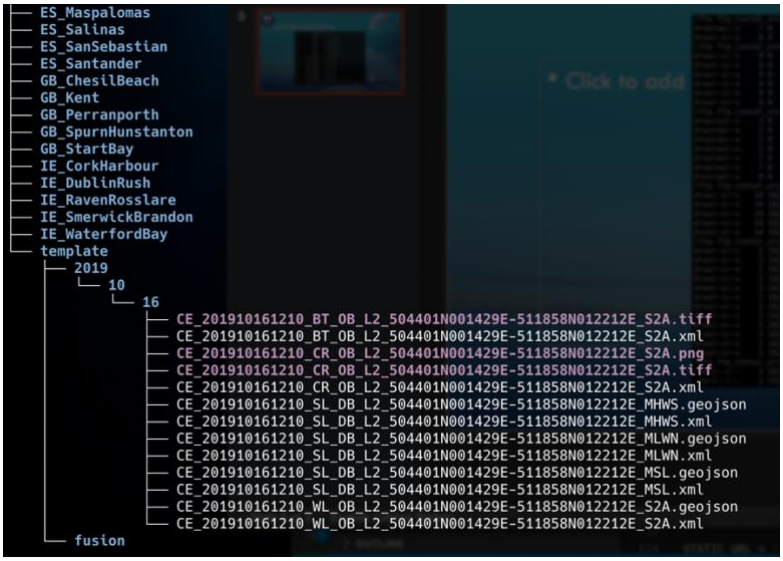

Figure 6: Mock-up of filesystem / naming convention organization

It will be seen from the preceding figure that if co-registered images are considered as another output, as they are the output of the pre-processor, and re-named according to the naming convention then even if in the same folder the products will stack alphabetically convenient for the user i.e. all outputs will be grouped with other outputs in nearest date order. Similarly similar products, such as shoreline based on differing datum will also be grouped together.

9 Metadata

Metadata is data that provides information about other data i.e. data about data. **Geospatial metadata** provides the type of metadata applicable to geographic data and information with features to be described in a metadata repository, or data inventory.

A metadata record is a file of information, usually presented as an XML document, which captures the basic characteristics of a data or information resource. It represents the who, what, when, where, why and how of the resource. Geospatial metadata an also be used to document geospatial resources including data catalogues, mapping applications, data models and related websites.[3]

There are several forms, of which we consider:

- **Descriptive metadata** is descriptive information about a resource. It is used for discovery and identification. The file naming convention provides sufficient & necessary components to describe the file contents.
- **Structural metadata** is metadata about containers of data. It describes types, versions, relationships and other characteristics of the product. The minimum requirement for all products is traceability to the container of files that includes the Uniform Resource

[3] MIT Libraries Guide: "Federal Geographic Data Committee (FGDC) Metadata". On MIT Libraries website, visited 16 October 2006Archived 18 October 2006 at the Wayback Machine

Indicator (URI) of all associated data, i.e. the input image(s) and any auxiliary data files (ADF), and a record of the processor history including name and version.

For most users only the descriptive metadata will be required, which should provide the user with type descriptor, location and date of the product. The file naming components will be mirrored in the database as attributes and providing a **searchable data catalogue**.

Metadata specification

Table 4 : Metadata tags, types & descriptions

Metadata Tag	Metadata Type	Short Description
ProductName	string	Filename without extension
ProductURI	string	e.g. https://servername/?RESTstring
ProductType	enum	e.g. { waterline | shoreline | seafront | … }
LocationName	**string**	e.g. Wexford Bay
LocationBBox	**array**	[lower left corner lat/lon, upper right corner lat/lon]
DateAcquired	**ISO8601**	YYYYMMDDThhmm{-YYYYMMDDThhmm}
DateProcessed	**ISO8601**	YYYYMMDDThhmm
DataFormat	**enum**	{ ESRI shapefile | KML | GeoJSON | GeoTiff | … }
ProcessorType	**enum**	{ waterline | shoreline | … }
ProcessorVersion	**string**	e.g. 1.4
InputFileURI	string set	e.g. [https://servername/?RESTstring]
AuxFileURI	string set	e.g. [https://servername/?RESTstring]
ProductProjection	string	e.g. WGS 84 | EPSG:4326

NOTE all of these will be maintained on a database NOT as a separate XML file. It may be on demand created as an XML file, or JSON, YAML … other metadata formats may apply.

The best metadata is metadata strongly coupled, i.e. associated, with the data it describes. Ideally there will be metadata components included within the product although this depends entirely on the format of the product as some formats allow inclusion of arbitrary metadata and others do not. This is why netCDF is such a useful format, because it allows arbitrary metadata although in many EO scenarios will constrain as much of this metadata to the Climate Forecasting (CF) Convention. If metadata is maintained as a separate XML (other formats are available, that are less machine-orientated and more human-readable) there is always the problem of the end user losing it, i.e. disassociating the data product and the corresponding metadata.

10 Infrastructure

Common sense, or “best practice”, is to separate the webserver and database server as separate virtual machines, or docker components. The initial deployment will use virtual machines.

Docker is a set of ‘platform as a service’ products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

For performance, if required, a third container could be deployed to run the controller, the web application, although initially this role will be taken by the webserver.

The web application developed during Phase 2 will be deployed as a VM and/or docker container to AdwäisEO. The latter choice has the advantage that the data is held on AdwäisEO storage, also used to host the long-term archive of ESA EO data.

11 Application Programming Interface (API)

The API is the intermediary that facilitates communication between the client and server applications. The client i.e. the user's browser, sends a message to the server which results in the delivery of data. A Representational State Transfer (REST) API is available and supported within an MVC pattern. REST is a *software architectural style* used to create conforming web services that are called RESTful services.

The long academic name is due to its first being used in a University of California, Irvine doctoral thesis in 2000[4]. It defines a set of constrains to allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. The term is intended to evoke an image of how a well-designed Web application behaves: it is a network of Web resources (a virtual state-machine) where the user progresses through the application by selecting resource identifiers such as https://erosion.argans.co.uk/barcelona/shoreline/2017 and resource operations such as GET or POST (application state transitions), resulting in the next resource's representation (the next application state) being transferred to the end user for their use.

Fortunately, fully understanding REST is not necessary to implement a RESTful web service as software frameworks are available. One is the Python Django REST framework[5], which since the web application prototype will be based on Python Django will be the framework most likely used.

12 Postprocessing

The user may require more options for data selection e.g. to quick view results before downloading. The input file sources for any product are recorded as metadata, which will be used to provide a quick look preview if the user is seeking to download source data.

A post-processor will be able to render shape files as a layer over the source image and, in the case of multiple shape files e.g. a time series, the option will be provided to view the shape files as an animation.

Using a RESTful interface allows the query URL to both define the data product location and to trigger any available post-processing. Consequently, we are considering providing users an on-demand data download service in which they can select a preferred format and, by using an appropriately parsed URL GET request, select, for example, whether to receive shape files as ESRI, KML or GeoJSON formatted files.

[4] Roy Fieldng, the author, is the co-founder of the Apache HTTP Server project and Senior Principle Scientist at Adobe Systems of San Jose, California.

[5] Available at https://django-rest-framework.org

In addition, the user may be interested in either a single data product or a collection of data products. Post-processing will perform necessary packaging so that the user only needs to download a single package rather than a set of individual files.

NOTE that functionality like filtering records by the user, or sorting records, or visualising metadata statistics are actually client-side processes and not, formally, considered as post-processing although are included here as 'to the user' they appear to be a post-process.

13 Conclusion

The User Interface and backend web application has been specified as version 0.0. Development and a prototype providing access to test site specific web pages was substantiated in response to a RID raised at MTR. The UI / web service will evolve throughout Phase 2. Current data access is via an FTP server; however, the webserver plan has described internal milestones that will deliver a fully functioning prototype demonstrator by mid-August and the final service demonstration will be live from the November.

END OF DOCUMENT