Horizon 2020
European Union funding
for Research & Innovation

ITS 4 LAND
innovations for land tenure

Deliverable 6.2
Technical Report

31 July 2018

Version 1.0

Abstract:
Technical report and software prototype of the mobile qualitative data processing system

Project Number: 687828
Work Package: 6
Lead: HL
Type: DEM
Dissemination: Public
Delivery Date: 31 July 2018
Contributors: Christian Timm, Dr. Mohammed Imaduddin Humayun, Stephanie Walter, Dr. Malumbo C. Chipofya, Dr. Sahib Jan, Kaspar Kundert

Its 4 Land
Hengelosestraat 99
Enschede 7500AE

Netherlands
Phone: +31534874532
www.its4land.com

The its4land consortium consists of the following partners:

University of Twente (UT)
KU Leuven (KUL)
Westfaelische Wilhelms-Universitaet Muenster (WWU)
Hansa Luftbild AG (HL)
Institut d'Enseignement Superieur de Ruhengeri (INES)
Bahir Dar University (BDU)
Technical University of Kenya (TUK)
ESRI Rwanda (ESRI).

Executive Summary

Deliverable D6.2 documents the outcome of task T6.2. Together with Deliverable D6.1 these are the first deliverables in Work Package 6 Publish and Share. Deliverable D6.1 and D6.2 are both implementations of its4land tools on the common Publish and Share platform.

The aim of task T6.2 is the development of the qualitative data processing system. The qualitative data processing system allows the processing and classification of sketch maps as well as the maintaining and calculation of the required training set for the used neural networks. The outcome will be spatial units that can be published to a Land Administration System for further use.

The Publish and Share platform can be considered on the one hand as a runtime environment for the tools developed in its4land and on the other as provider of data and information for existing LAS or other tools. Major work in both tasks was the development of the general Publish and Share platform architecture. The developed architecture allows a mobile usage of the its4land tools, as well as online usage following a geocloud approach.

The core elements of the Publish and Share architecture are:

- A set of public REST-APIs to interact with the Publish and Share platform
- A Docker based runtime environment for tools developed in Its4land
- A set of data stores for alphanumeric, geo, binary and image data
- OGC services for data dissemination

Task T6.2 adapts the results of Work Package 3 to the Publish and Share platform. The prototype implementation of the Draw and Make tools from Work Package 3 is restructured and modified to operate in the Publish and Share runtime environment and make use of the Publish and Share APIs.

The implementation of the Draw and Make tools from Work Package 3 on the Publish and Share platform is the called qualitative data processing system.

Contents

Abbreviations

FOSS	Free Open Source Software
GDAL	Geospatial Data Abstraction Library
GRASS	Geographic Resources Analysis Support System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
LADM	Land Administration Domain Model
LAS	Land Administration Domain Model
OAS	OpenAPI Specification
OGC	Open Geospatial Consortium
ORM	Object-relational mapping
REST	Representational State Transfer
SME	Small-Medium-Enterprises
TRL	Technical Readiness Level
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WFS	Web Feature Service
WMS	Web Map Service

1 Introduction

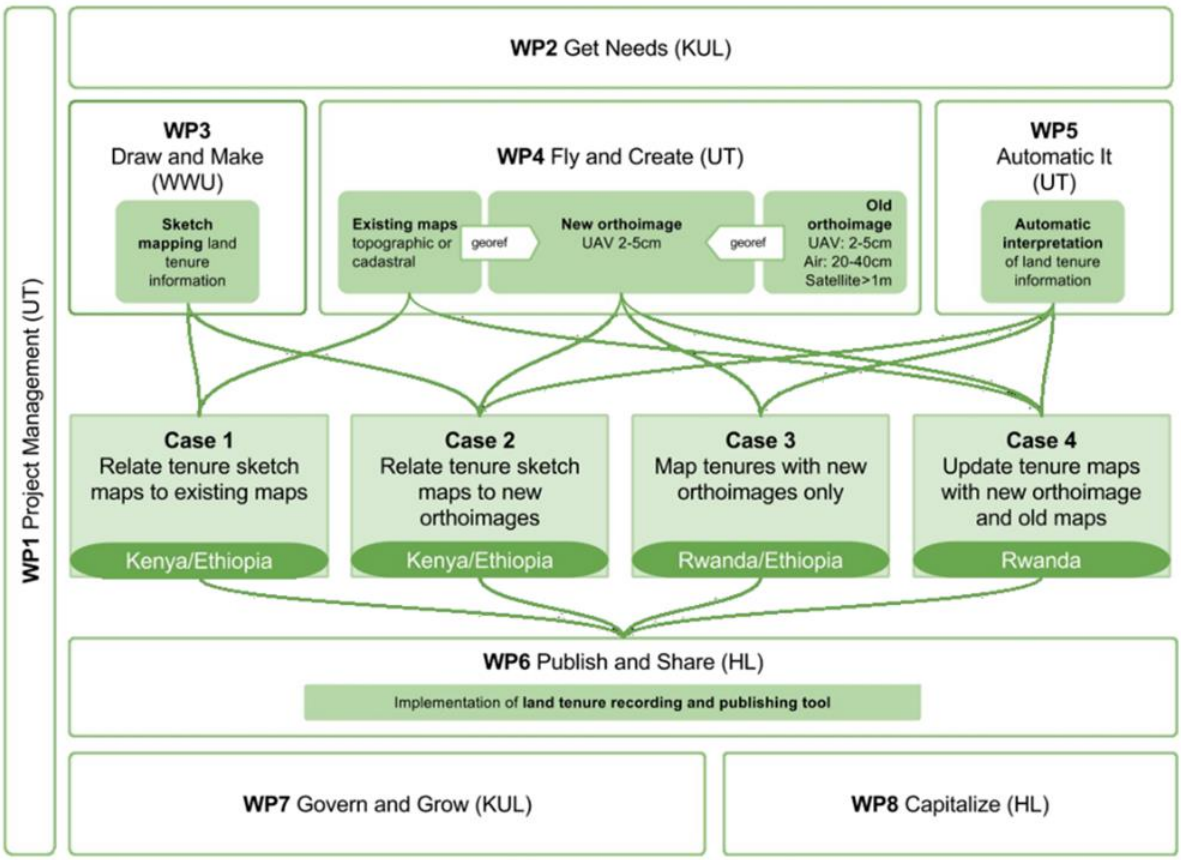
Its4land is a European Commission Horizon 2020 project funded under its Industrial Leadership program, specifically the 'Leadership in enabling and industrial technologies – Information and Communication Technologies ICT (H2020-EU.2.1.1.)', under the call H2020-ICT-2015 – and the specific topic – 'International partnership building in low and middle income countries' ICT-39-2015.

Its4land aims to deliver an innovative suite of land tenure recording tools that respond to sub Saharan Africa's immense challenge to rapidly and cheaply map millions of unrecognized land rights in the region. ICT innovation is intended to play a key role. Many existing ICT-based approaches to land tenure recording in the region have failed: disputes abound, investment is impeded, and the community's poorest lose out. Its4land seeks to reinforce strategic collaboration between the EU and East Africa via a scalable and transferrable ICT solution. Established local, national, and international partnerships seek to drive the project results beyond R&D into the commercial realm. Its4land combines an innovation process with emerging geospatial technologies, including smart sketch maps, UAVs, automated feature extraction, and geocloud services, to deliver land recording services that are end-user responsive, market driven, and fit-for-purpose. The transdisciplinary work also develops supportive models for governance, capacity development, and business capitalization. Gender sensitive analysis and design is also incorporated. Set in the East African development hotbeds of Rwanda, Kenya, and Ethiopia, its4land falls within TRL 5-7: 3 major phases host 8 work packages that enable contextualization, design, and eventual land sector transformation. In line with Living Labs thinking, localized pilots and demonstrations are embedded in the design process. The experienced consortium is multi-sectorial, multi-national, and multidisciplinary. It includes SMEs and researchers from 3 EU countries and 3 East African countries: the necessary complementary skills and expertise are delivered. Responses to the range of barriers are prepared: strong networks across East Africa are key in mitigation. The tailored project management plan ensures clear milestones and deliverables, and supports result dissemination and exploitation: specific work packages and roles focus on the latter.

1.1 The Publish and Share platform in its4land

"Publish and Share" combines the tools and methods developed in "Draw and Make", "Fly and Create" and "Automate it" in a technical platform (see **Figure 1**).
The Publish and Share platform can be considered on the one hand as a runtime environment for the tools developed in its4land and on the other hand as provider of data and information for existing LAS or other tools. The platform will be accessible via service interfaces based on standards from OGC and W3C. The modelling of the interfaces follows the concepts introduced by LADM. External systems like LAS or planning systems can use the service interfaces to integrate data into their own processes, based on specific national rules. The usage scenarios and workflows to combine the its4land tools with land administration systems will be defined and implemented to a prototype level in the Deliverable D 6.4.

Figure 1: Overview of the its4land work packages



The implementation of the Publish and Share platform follows a toolbox approach and will provide a framework of common APIs and services used by all its4land tools. From this toolbox, a user can select those its4land tools fitting his tasks best.
As per the paradigm of geocloud, the tools will be implemented as services accessible via an API based on web standards. Tools that cannot be implemented as a web service, because of their dependencies on libraries, operating systems and desktop specific user interfaces, can be operated on their native platform. Tools which are capable of calling a REST API can make use of any kind of Publish and Share service, like public APIs or storage services. Since these 3rd party tools will be developed outside the scope of its4land, it is the responsibility of the 3rd tool vendor or creator to adapt their tools to Publish and Share.

Its4land's claim is the development of state of the art methods for recording land rights with special consideration of the needs of local stakeholders in developing countries. To achieve a close integration of local stakeholders, the flexibility to adopt local needs and land tenure concepts is required. This is achieved through use of the tools and methods on site. With this in mind, Publish and Share follows two dominant usage scenarios. The first scenario, which we call mobile (offline), offers end-to-end services provided by the its4land geocloud. These are not hosted remotely, but locally at the site of usage. It is intended to be used in areas without network infrastructure.

In the second scenario, which we call geocloud, services are hosted remotely. These can be used anytime and anywhere if the network infrastructure allows it.

1.2 Mobile qualitative data processing system

The qualitative data processing system is a subset of the entire Publish and Share platform. The aim of the qualitative data processing system is the integration of the tools and methods developed by Work Package 3 into the Publish and Share platform.

The qualitative data processing system allows the processing and classification of sketch maps as well as the maintaining and calculation of the required training set for the used neural networks. The outcome will be spatial units that can be published to a land Administration System for further use.

The qualitative data processing system is built on an architecture that allows for a mobile in-field usage in combination with a geocloud environment. Depending on the available infrastructure and the concrete needs, parts of the qualitative data processing can be deployed in different ways to set the focus more on mobile in field or geocloud characteristics.

2 Architecture of the qualitative data processing system

The following section describes the qualitative data processing system architecture. The qualitative data processing system is created on the Publish and Share platform. The architecture of the Publish and Share platform is described here only as far as it is necessary for the understanding of qualitative data processing system architecture. The Publish and Share APIs mentioned in this section are described separately and are available via the share4land platform.

2.1 Overview

The mobile qualitative data processing system integrates the results from Work Package 3 into the technical framework of the Publish and Share platform developed in Work Package 6.
The functionality and methodology for the qualitative data processing was developed by Work Package 3 in 3 deliverables:

- D3.2 Implementation of semantic recognition of sketched objects [1]
- D3.3 Implementation of qualitative representation of sketchmap [2]
- D3.5 Implementation of sketchmap alignment prototype with existing map [3]

Work Package 6 has developed a unified architecture for the Publish and Share platform for hosting tools and data developed or created by the different work packages of its4land. Publish and Share provides a set of services that can be utilized by its4land tools. The main services of the platform are:

- Runtime environment for executing and managing tools provided by other work packages
- Storage services for alphanumeric, binary, geo spatial and image data
- OGC services for spatial data access
- Authentication and authorisation services

The services are exposed via Representational State Transfer (REST) API [4]. REST is an architectural style for a distributed system, especially web services. REST-compliant web services allow the requesting client to access and manipulate web resources by using a uniform and predefined set of stateless operations. Web resources are defined and identified by their URI. Often, a REST API is implemented via the HTTP or HTTPS protocol. The operations on a resource are defined by the HTTP/HTTPS request methods.

Table 1: HTTP/HTTPS methods

Request method	**Description**
GET	Request the specified resource
POST	Create a new resource
PUT	Replace or create a resource
PATCH	Updates a resource
DELETE	Deletes a resource

Additional request methods, like HEAD or TRACE are possible but not common. Publish and Share uses only the request methods documented in **Table 1**.

The OpenAPI Specification (OAS) [5] is used to document the its4land REST API in a standardized form. OAS specifies the REST API in a machine-readable way for describing, producing, consuming, and visualizing RESTful web services.

Publish and Share provides furthermore, a web-based client for visualization of spatial and non-spatial data. This client also includes a user interface to start, stop and monitor the tools implemented on the runtime environment.

Figure 2: High level architecture of the qualitative data processing system.

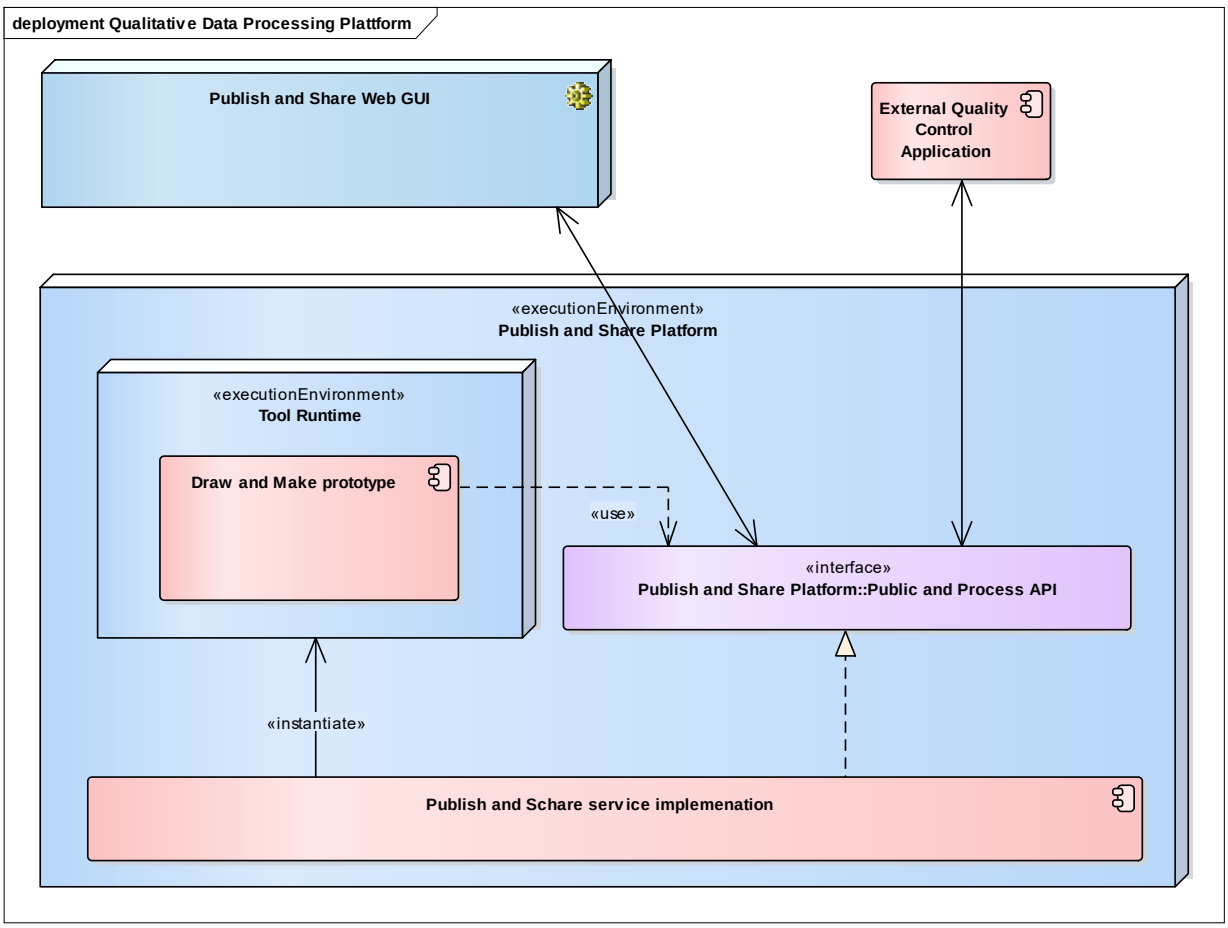


Figure 2 provides a high-level overview of the Draw and Make prototype implemented as part of the qualitative data processing system on the Publish and Share platform.

The main part of the qualitative data processing system is implemented and hosted in the tool runtime environment. This includes all the computational part for data pre-processing and machine learning.

The qualitative data processing system can be extend by two additional components:

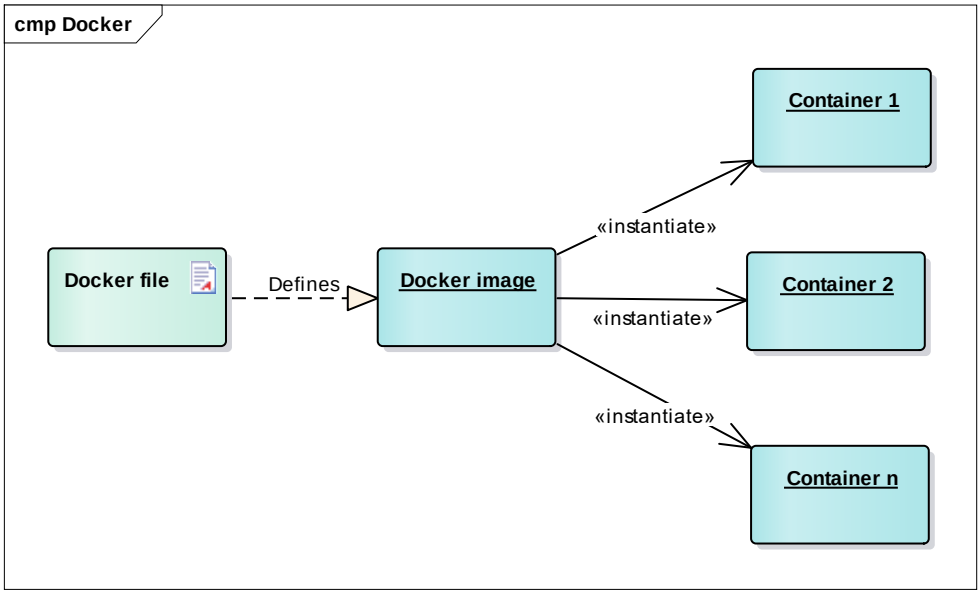
- Manual quality control of the scanned sketch maps
- Manual quality control of the sketch map alignment

These components can be integrated via the Publish and Share public API. Both components are not a prerequisite for the qualitative data processing platform and the process of quantifying qualitative information, but provide additional interactive manipulation of intermediate results of the process. (See **Figure 8**).

Primarily, the qualitative data processing system uses base-functionality provided by the Publish and Share platform as services. The API allows the Draw and Make prototype to use service of the Publish and Share platform and to communicate with tools from other Work Packages in a defined way. Additionally, the qualitative data processing system uses the runtime environment, which is controlled by the Publish and Share platform.

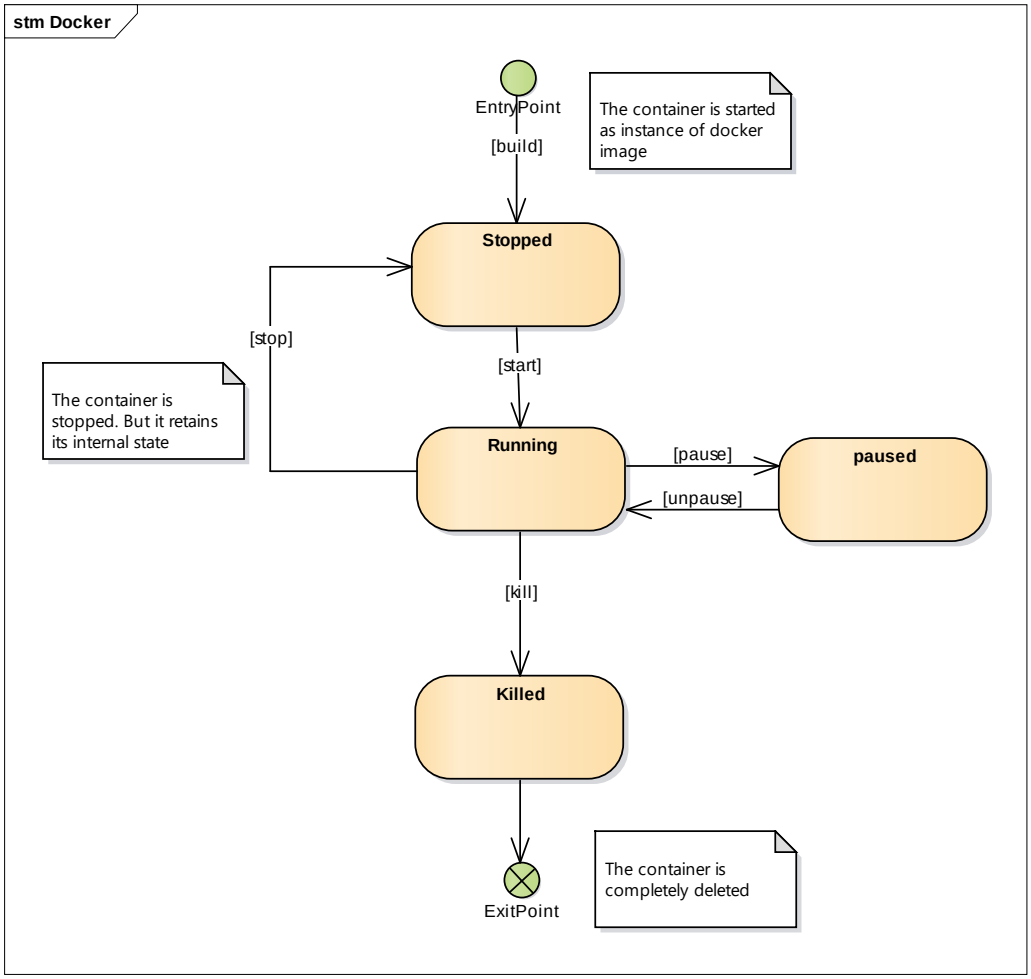
The runtime environment is based on “Docker” [6]. Docker is used to run software packages called "containers". In a typical example, one container runs a web server and web application, while a second container runs a database server that is used by the web application. In another example, multiple containers running the same web application are started in parallel for scaling issues. Containers are isolated from each other and use their own set of tools and libraries. All containers use the same kernel and are therefore more lightweight than virtual machines. Containers are created from "images" which specify their precise contents. Images are defined by a “dockerfile” (see **Figure 3**).

Figure 3: Relationship between Docker file, image and container



A Docker container has a defined lifecycle. In the Publish and Share platform, this lifecycle is controlled by the platform. This includes starting, stopping and terminating a container (see **Figure 4**). The Publish and Share API allows the tool to communicate with the platform when the container is running. The container is started asynchronously as a background process, so even long running calculations will neither block the application nor require a user to remain logged on the system for the duration of the run.

Figure 4: UML State Machine Diagram showing the lifecycle of a Docker container



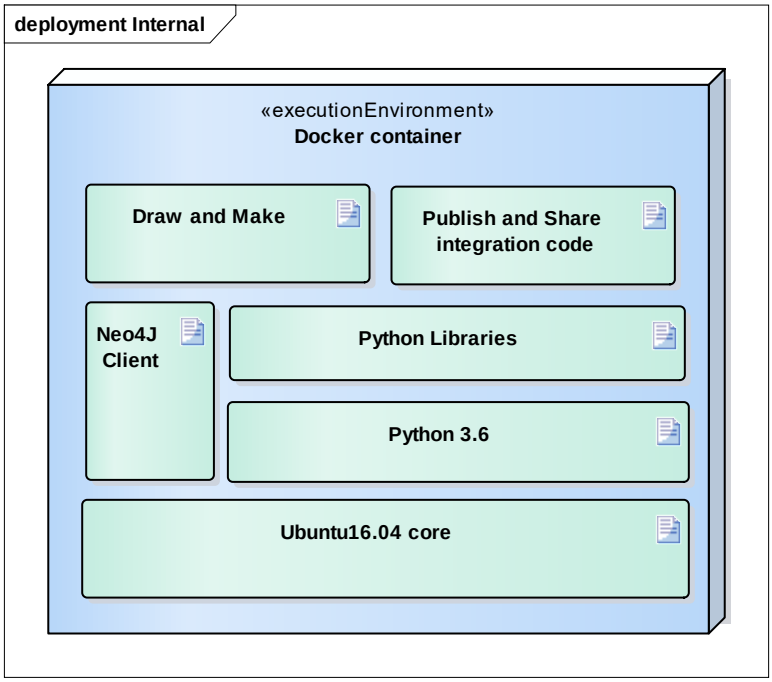
The platform provides a repository of predefined Docker images. The computational part of the qualitative data processing platform is implemented in one Docker image (see **Figure 5)**.

Furthermore, the Docker image contains the code for interacting with the Publish and Share platform to support container management. This includes:

- Logging of errors
- Feedback on the container status
- Feedback on the application status
- Updating the process status

This code is not exclusive to the Draw and Make image, but is used in every Docker image that should be used on the Publish and Share platform. The code implements the process interface and is necessary to manage and monitor a container within the Publish and Share platform.

Figure 5: Content of the Draw and Make container

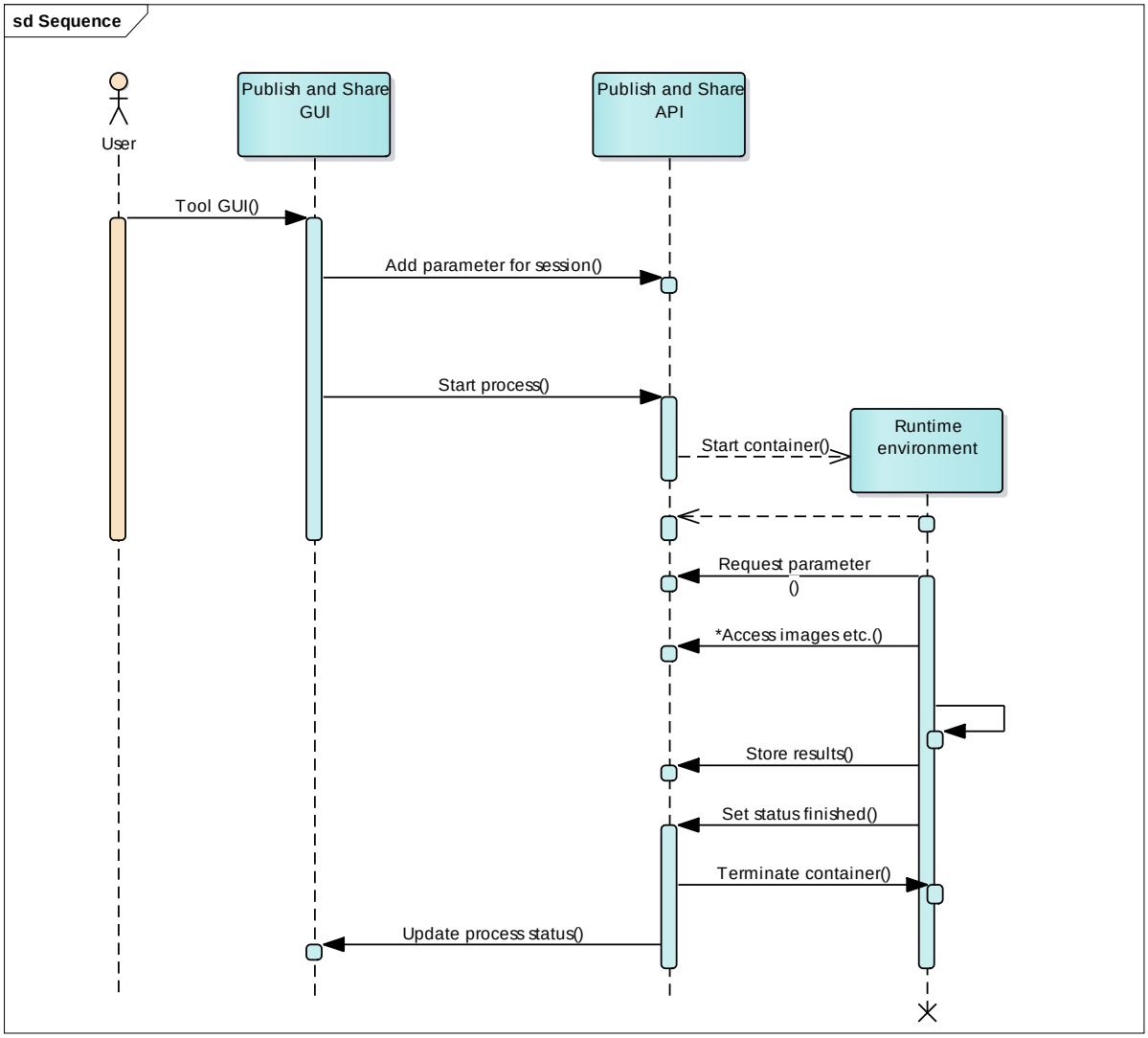


The interaction with the Draw and Make tools in the container is realised by a GUI implemented in the Publish and Share web client. For Draw and Make the GUI allows the following interactions (see **Figure 8**):

- Create training sets and classifier
- Qualify the metric map
- Control the sketch map recognition process
- Control the sketch map qualification process
- Control the qualitative alignment process
- Control the geometric approximation process

For each interaction the GUI collects the required parameters and starts the associated tool. The API makes use of a repository that contains the association between GUI and a concrete part of the Draw and Make implementation. This allows one to start the container with the necessary parameters (see **Figure 6**).

Figure 6: Start of the Draw and Make tool on Publish and Share



The concept of image based containers allows participating teams to develop the tools independently from the Publish and Share platform. Tool providers like the Work Packages 3, 4 and 5 can independently develop and deploy their tools. The tools only have to be registered in the Publish and Share platform once in advance.

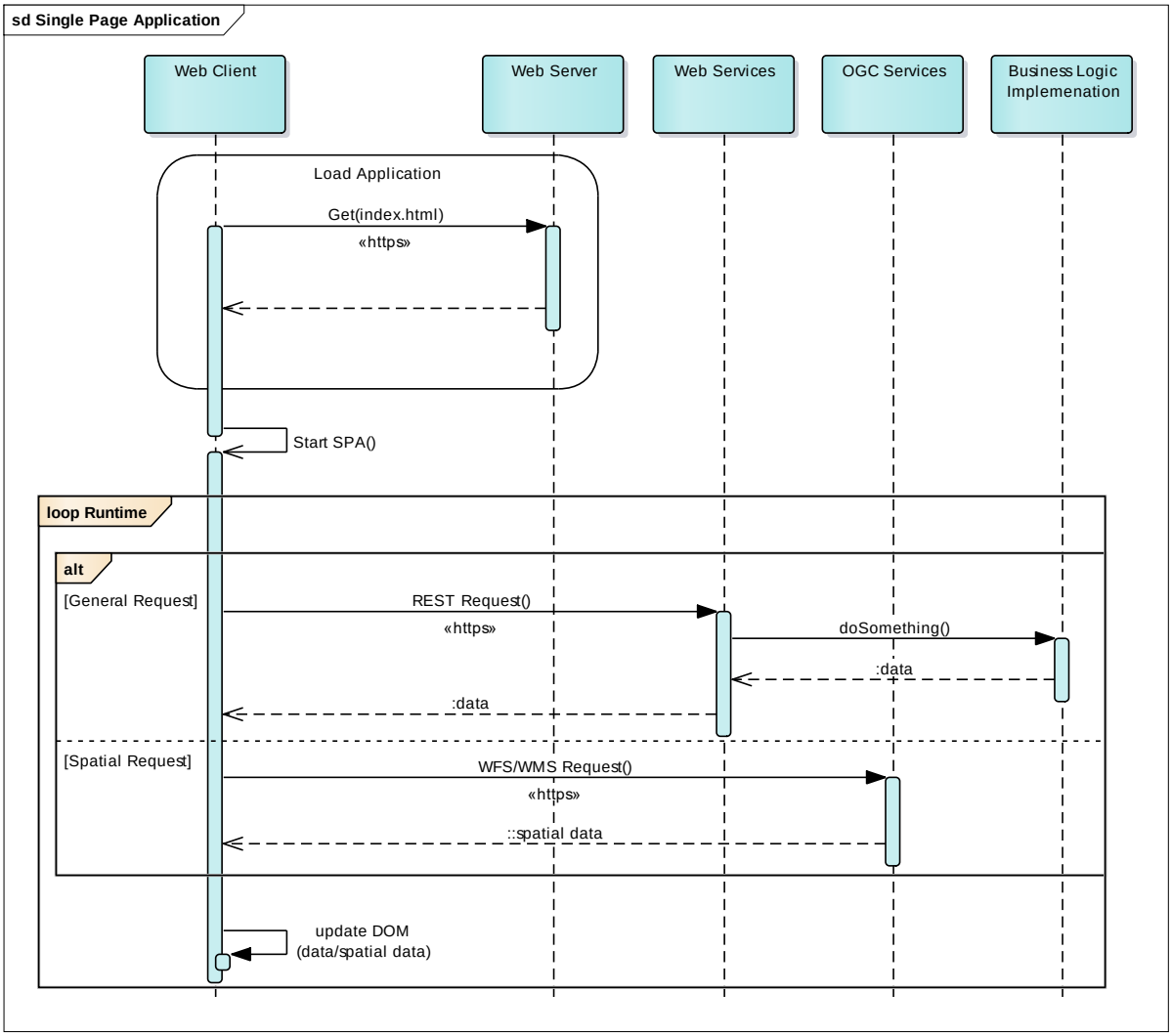
The external applications of Draw and Make for quality control, are not executed under the control of Publish and Share. However, both tools use the Publish and Share API to access required resources. This includes authentication and authorisation, as well as tool specific data like training or validation sets.

2.2 Technology stack used for the qualitative data processing platform

The technology stack of the qualitative data processing system is defined by the Publish and Share platform. The Publish and Share platform consists of four layers:

Front-end layer: The front-end layer covers the user clients. The main Publish and Share web client is implemented as a Single Page Application (SPA) web-client (see **Figure 7**) [7]. The implementation is done in JavaScript [8] based on the JavaScript frameworks React [9] and the web mapping framework ExperMaps [10] to visualize geo data from OGC [11] services like WMS[12] and WFS [13].

Figure 7: Sequence of a SPA Web Application



Besides using the SPA client, the qualitative data processing system can be extended by additional system specific frontend tools. Two tools for quality control are currently under consideration.

Tool runtime layer: The runtime layer is implemented using Docker. The tools running on the runtime layer are encapsulated inside the Docker image. Publish and Share makes no specifications about the internal structure of the image, as long as the tool itself can be executed via the command line.

Service Layer: The services are accessible via REST API or OGC WMS and WFS services. The REST API is implemented on the Node.js runtime environment [14] in JavaScript [8]. By its very nature, a platform like Publish and Share requires a large number of different frameworks and libraries. Most of them are not relevant for the understanding of the architecture. So we list only the architecturally relevant frameworks and libraries:

- ExperMaps [10]: Provides the necessary services for authentication, authorization, session management, layer management and the runtime for the REST API
- Swagger [15]: Specifies, document, implement and manage a REST API based on the OpenAPI Specification (OAS) [5]
- Sequelize [16]: Object-Relational-Mapper (ORM)
- The OGC services are implemented on GeoServer [17]

Data Layer: The data layer provides four kind of storage classes:

- Object-relational storage

  The object-relational storage is implemented in PostgreSQL [18] in combination with PostGIS [19]. The object-relational database stores any kind of structured alpha-numeric data and also vector geometries.

- Graph oriented storage

  The graph oriented storage is implemented in Neo4J [20]. This storage class is used to store e.g. constraint networks.

- Object storage

  The object storage is implemented in the demonstrator by using Amazon Web Service S3[21]. In other environments, the Amazon cloud storage service S3 can be replaced by an S3 compatible open source solution like Mino [22]. The object storage is used to store any kind of file-based or unstructured data, including images and JSON [23] files.

- Block oriented file system storage

  The block oriented file system storage is implemented by the concrete hosting platform of Publish and Share and can vary between different sites of installation. In general this is a UNIX-style file system. It is used for storing files that should not be stored in the object storage. E.g. images that should be published via GeoServer [17]

3 Adaptation of the Draw and Make prototype to the Publish and Share platform

In the following section we will describe how the prototype of the Draw and Make tool was modified in order to adapt it to the Publish and Share platform. This includes identification of issues in the current implementation which impair its direct adoption by the platform, refactoring the existing program structure and finally compliance with recommended structural guidelines laid out by the community for Python –the programming language used for development. In addition to getting the prototype into a proper shape for integration with the platform, other objectives pursued were the ease of development across distributed teams, better maintainability through unit tests, the ability to package and distribute the developed library as a standard Python package and ease of deployment.

Since the Publish and Share platform was not yet available during the development of the prototype, the prototype aimed at implementing research ideas and algorithms while focusing more on their utility. In the following sections, we outline the steps carried out to organize the Draw and Make prototype into a form that is easily adaptable to the platform.

3.1 Analysis of the implemented prototype

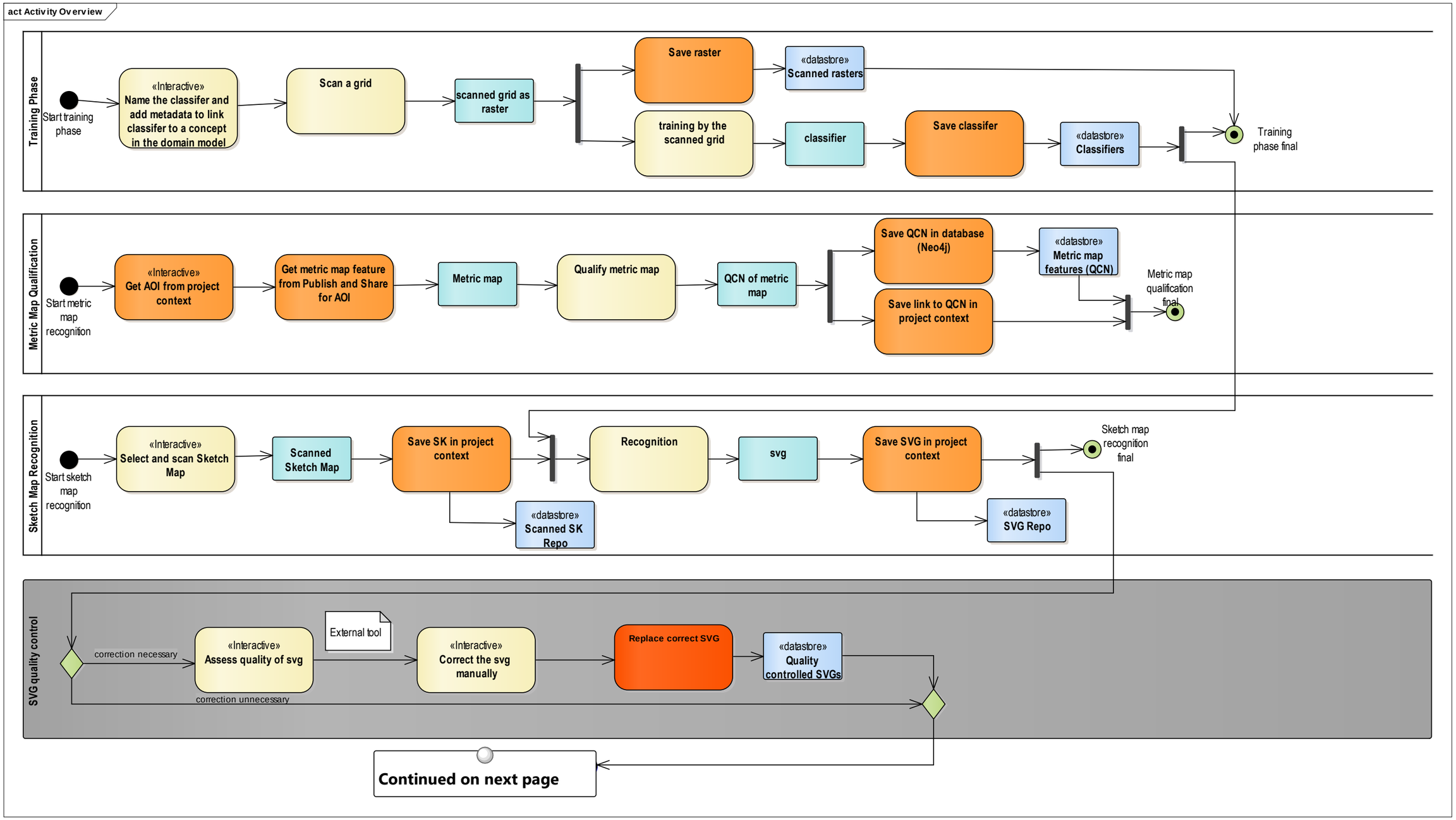
The functionality provided by the Draw and Make tool can be broadly categorized into three distinct tasks from the perspective of software development. The table below summarizes them along with the required inputs and outputs generated by each task. The final column gives a rough estimate of the computational cost incurred in terms of time, although in practice this is highly dependent on the size and number of features in the input maps.

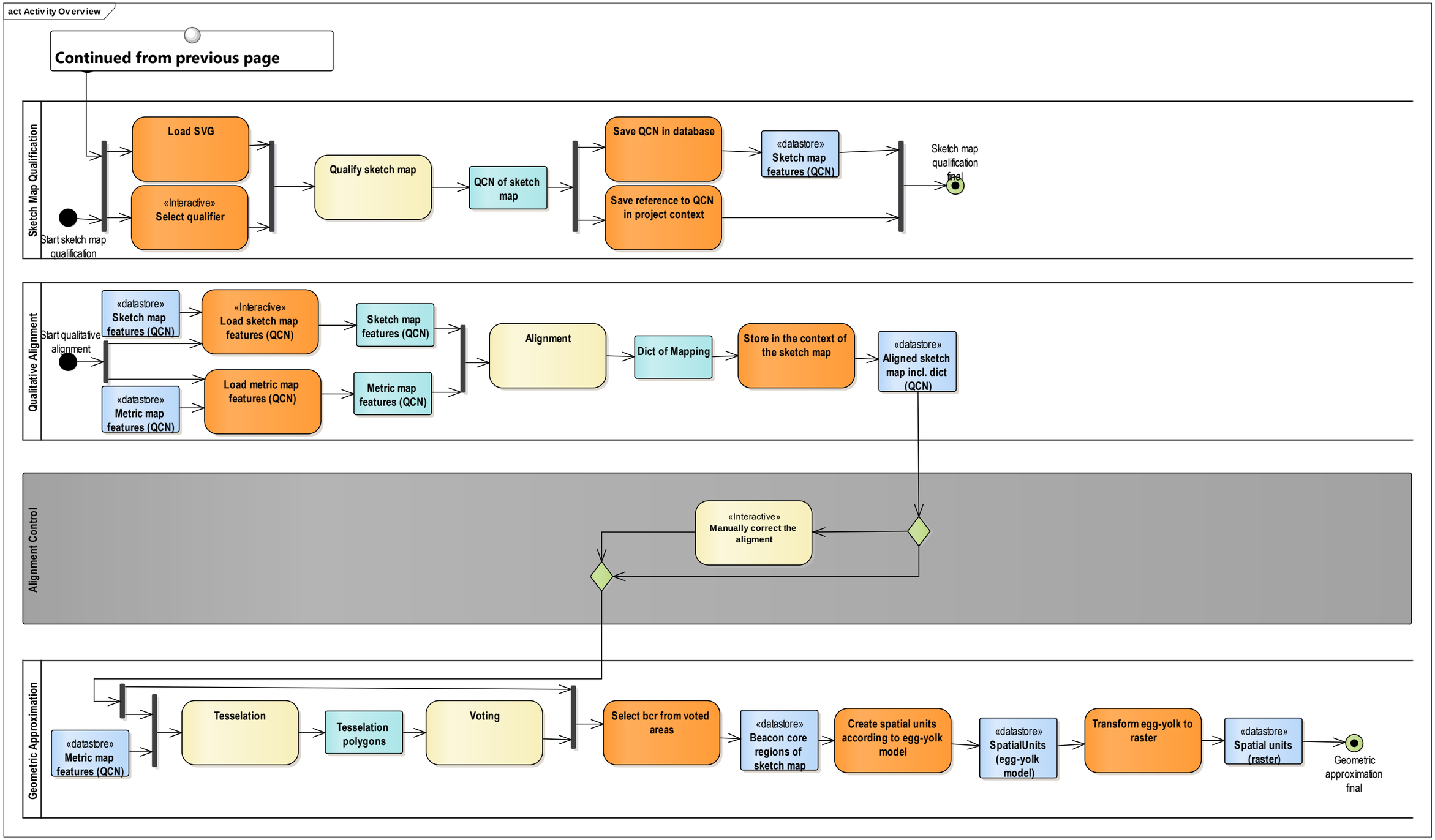
Table 2: Categorized implementation tasks in Work Package 3 - Draw and Make

Task	**Inputs**	**Output**	**Computational cost**
Sketch recognition and object detection	Raw sketch map, symbols used in sketch maps	Classifier to perform detection of sketched objects	Medium (detection) to High (training)
Qualitative spatial processing	Classified sketch map, metric map, qualification calculi to use	Qualitative constraint network, matched objects (sketch to metric)	Medium to High
Approximate boundary generation for vague or ambiguously defined regions	Matched map objects, qualitative constraint network	Boundaries of regions as per the egg-yolk model [24]	Medium

While **Table 2** presents a broader picture, implementing it requires further subdivision of tasks and organizing the control flow. A more detailed overview of the various steps involved in Draw and Make prototype can be observed in the documented UML activity diagram (**Figure 8**)

Figure 8: Activity Diagram of the Draw and Make algorithm




act Activity Overview
Continued from previous page
Sketch Map Qualification
Start sketch map qualification
Load SVG
«Interactive»
Select qualifier
Qualify sketch map
QCN of sketch map
Save QCN in database
Save reference to QCN in project context
«datastore»
Sketch map features (QCN)
Sketch map qualification final
Qualitative Alignment
Start qualitative alignment
«datastore»
Sketch map features (QCN)
«Interactive»
Load sketch map features (QCN)
Sketch map features (QCN)
«datastore»
Metric map features (QCN)
Load metric map features (QCN)
Metric map features (QCN)
Alignment
Dict of Mapping
Store in the context of the sketch map
«datastore»
Aligned sketch map incl. dict (QCN)
Alignment Control
«Interactive»
Manually correct the aligment
Geometric Approximation
«datastore»
Metric map features (QCN)
Tesselation
Tesselation polygons
Voting
Select bcr from voted areas
«datastore»
Beacon core regions of sketch map
Create spatial units according to egg-yolk model
«datastore»
SpatialUnits (egg-yolk model)
Transform egg-yolk to raster
«datastore»
Spatial units (raster)
Geometric approximation final

The purpose of this activity diagram is not to document the algorithm in detail. This was done in the Deliverables D3.2, D3.4 and D3.5 [1][2][3]. The purpose is to modularize the source code, identify the object and control flow inside the source code. This allows identifying those parts of the algorithms that require an interaction with the Publish and Share platform, as well as required data structures. Examples for those interactions are:

- Storing of results
- Requesting source data, like images and maps
- Storing and accessing intermediate results, like training sets, classifier and the qualitative constraint network
- Chaining data in a continuous workflow

Steps in the process that require interaction with the Publish and Share platform are coloured in orange. Steps that require input from the user, e.g. selecting input data are tagged with the stereotype “interactive”.

The diagram is organised in eight main partitions:

1. Training phase
2. Metric map qualification
3. Sketch map recognition
4. SVG quality control
5. Sketch map qualification
6. Qualitative alignment
7. Alignment control
8. Geometric approximation

The partitions 4 and 7 describe interactive tools that are external applications, which are not part of the Publish and Share platform itself. Both tools can be used to interactively correct intermediate data to improve results. The tools are not necessary for the usefulness of Draw and Make, since they do not change the core process. Nevertheless both tools can make use of and update intermediate data by using the Publish and Share API.

The partitions represent self-contained blocks of Draw and Make with a distinct input and output. Each of these blocks are defined by a separate entry point. From a software engineering perspective, these blocks can be implemented in different ways:

- Individual executable with input parameters
- API of an executable that acts as a server
- One main executable with different sets of parameters

The existing prototype was developed by several authors working in parallel on different tasks. The resulting code was functionally organized in the form of executable Python scripts, each of which performed related tasks such as qualification (both sketch and metric),

qualitative alignment or geometric approximation. The following issues needed to be fixed or improved upon to adapt the prototype to the Publish and Share platform:

- Redundancy - as the same functionality was repeated across individual modules. There was a lack of clear separation between general reusable components and application specific components
- Replication and deployment require considerable manual effort
- Host filesystem dependence for I/O operations
- Computationally expensive tasks block other processes from running until they terminate
- Changing runtime parameters required modification of source code
- Inconsistent coding style. Poorly documented code.

In the next section, we discuss how these issues were tackled to improve the code structure and quality to a form suitable for integration with the platform.

3.2 Redesign and restructuring of the prototype code

Each of the following subsections discusses steps that were addressed to upgrade the prototype to a level of readiness for adaptation to the Publish and Share platform.

3.2.1 Addressing reusability and modularity

To minimize redundancy, the common functionality was moved to a work package specific library, which could then be imported by other modules. Common functionality specific to Draw and Make includes operations such as reading and writing to sketch map format (SVG) and the metric map format (GeoJSON), qualifying spatial data using different calculi, performing spatial operations and querying the qualitative constraint network via the Graph database (Neo4j).

Authors working on the individual tasks are now able to implement tools and applications reusing the functionality provided by the library. Rather than individual executable scripts, the new modular structure also makes it possible to include the functionality of the entire tool in a single one with parameters, which is the preferred method for deployment on the Publish and Share platform. Parameters can be provided via the command line interface and a configuration file.

3.2.2 Packaging and Deployment

An additional benefit of organizing code as a separate library is the flexibility it brings in terms of packaging and deploying code. By restructuring code in the form of a standard

Python package, it is now possible to export, upload and share it via the Python package repository[25]. This allows the algorithms developed for Draw and Make to be used not just within the scope of the its4land project, but also by other interested parties who want to develop and contribute to Free Open Source Software (FOSS).

The implementation reuses a number of third party libraries, which previously required manual installation. These are now provided as metadata in a single text file, which tools provided by the Python ecosystem can use, to automate the installation of the third party libraries via a single command.

The interdependence between the host operating system, available libraries and Python packages which use them means that despite best efforts, runtime issues can be encountered due to incompatibilities. The switch to Docker containers mitigates this and makes it possible to package the entire tool along with its dependencies into a single image, thereby providing a consistent runtime environment. An additional benefit is that computationally expensive operations now run within a container. The container itself is managed by the Publish and Share platform and can be started, paused, stopped and run in the background, foregoing the need for asynchronicity at the tool level itself. Moving to a container based architecture also requires that I/O operations requiring data persistence use APIs provided by the platform, since locally generated files last only for the lifetime of a container. Use of containers also eases deployment in the mobile (offline) usage scenario, e.g. on a computer used in the field.

3.2.3 Ease of collaborative development and maintenance

Since the development of Draw and Make involves collaborative development by multiple authors, a set of consistent guidelines with regard to the syntactic structure of programs is needed to ensure consistency in source code. To this end, we began the process of switching the codebase to guidelines recommended by the PEP8 standard[26]. The standard defines naming conventions, comment styles and general layout for consistent and readable source code. Automated tools are used to ensure compliance with the guidelines during the development phase.
Another important addition is the use of unit tests. The goal here is to test the entire functionality provided by the library, in order to prevent regressions and ensure that addition or modification of a feature does not introduce bugs. A unit testing framework is now available to aid future development of the Draw and Make tool.

4 Metric Map Features

The Publish and Share platform offers various services for the different its4land tools, e.g. a service to obtain georeferenced topographic features which are required in Draw and Make to align the sketch maps.

This service contains an API endpoint and a model for topographic features. The API endpoint is implemented in the Public API. The endpoint allows querying a topographic feature by specifying a query window. The endpoint returns the selected topographic feature as a GeoJSON structure. The API also provides an endpoint for inserting new topographic features.

The Publish and Share platform contains a model for classification of topographic features. The model is structured hierarchically in 3 levels:

Table 3: Levels of the Publish and Share general feature model

Level	Table representation in **Figure 9**	Description
Feature group	t_fgroup	Top level classes
Feature type	t_ftype	A mandatory sub class under a group.
Feature subtype	t_fsubtype	A not mandatory sub class under a type.

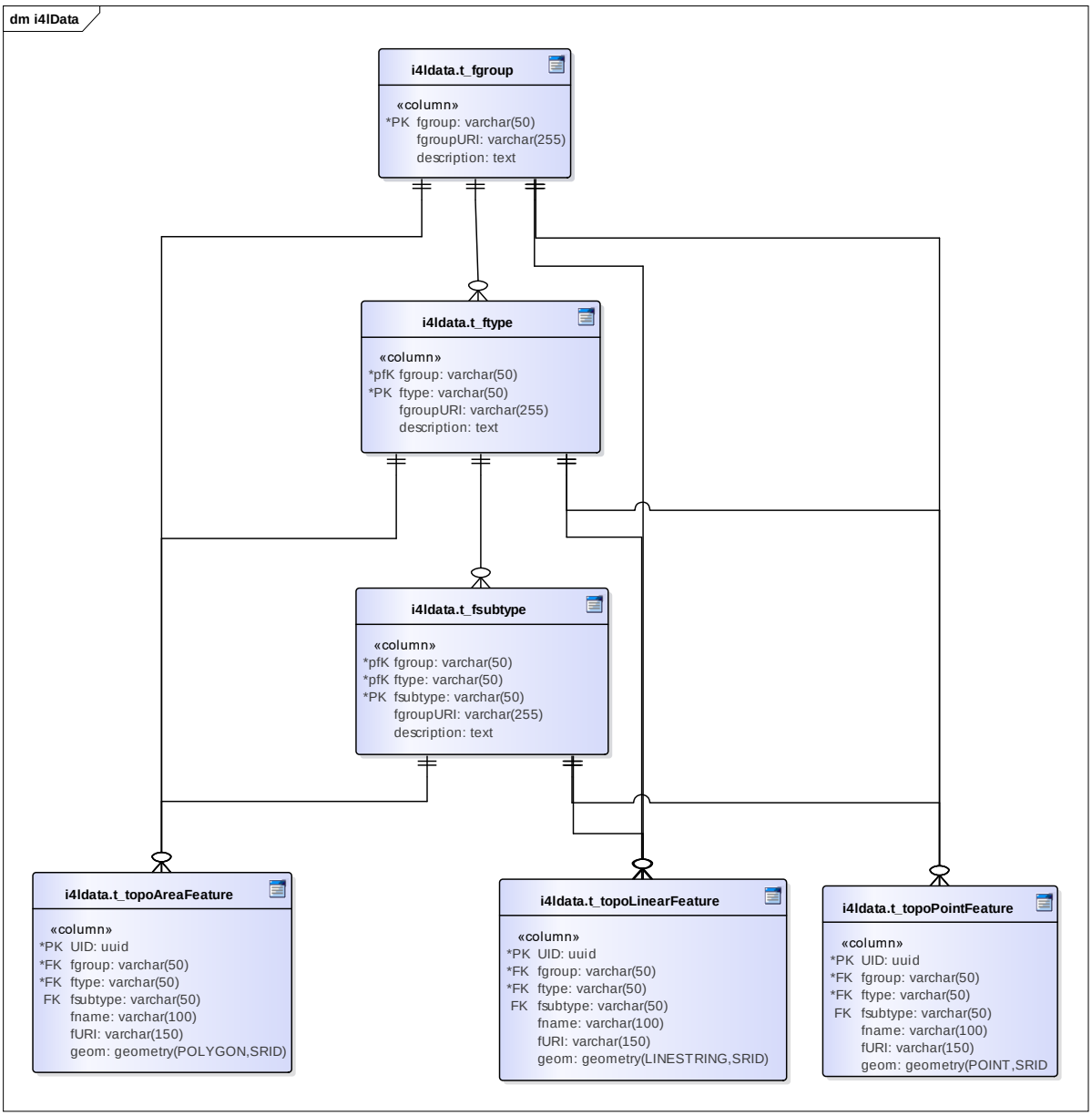
The hierarchical structure can host a simple domain ontology of topographic features. All three levels are representing classes. Due to the hierarchy, a feature group subsumes feature types and a feature type subsumes feature subtypes. Publish and Share does not specify any specific ontology by default.

The structure of the model and its implementation is deliberately kept simple **Figure 9** shows the database representation of the feature model. Instances are handled separately per geometry type in the tables “t_topo[Point|Linear|Area]Feature”. Possible types are point, line and polygon according to the ISO/TC 211 and OGC simple feature specification [27].

The availability of topographic vector data in the three east African pilot countries is very limited. Therefore there is no ontology which can be derived from those data.
For the implementation of the prototype, the lack of vector data is compensated by a manual acquisition of suitable data for the alignment in Draw and Make. The capture is based on the Map features catalogue defined by OpenStreetMap [28]. The OpenStreetMap feature catalogue uses a key value structure for classifying features. The key represents a brief classification of features (e.g. “boundary”, “landuse” or “natural”). The value refines the classification (e.g.: “boundary”:”protected_areas”, “landuse”:”farmyard” or “natural”:”grassland”). We mapped the OpenStreetMap Key to the feature group, the OpenStreetMap Value to the feature type and did not use the feature subtype.

Another feature group we named its4land has been added to the catalogue. This group contains classes that are specific to its4land (e.g. unclassified linear features that were created in the Automate it tool) or are too specific to be used in a general feature catalogue (e.g. Boma).

Figure 9: Database representation of the feature model in Publish and Share (Information Engineering Notation)



5 Implementation of the geometric approximation

Work Package 6 does not only adapt the existing Draw and Make prototype from Work Package 3 to the Publish and Share platform. In the context of Work Package 6 also the implementation of the geometric approximation part of the qualitative data processing system was also done.

Figure 10: Deliveries related to Draw and Make

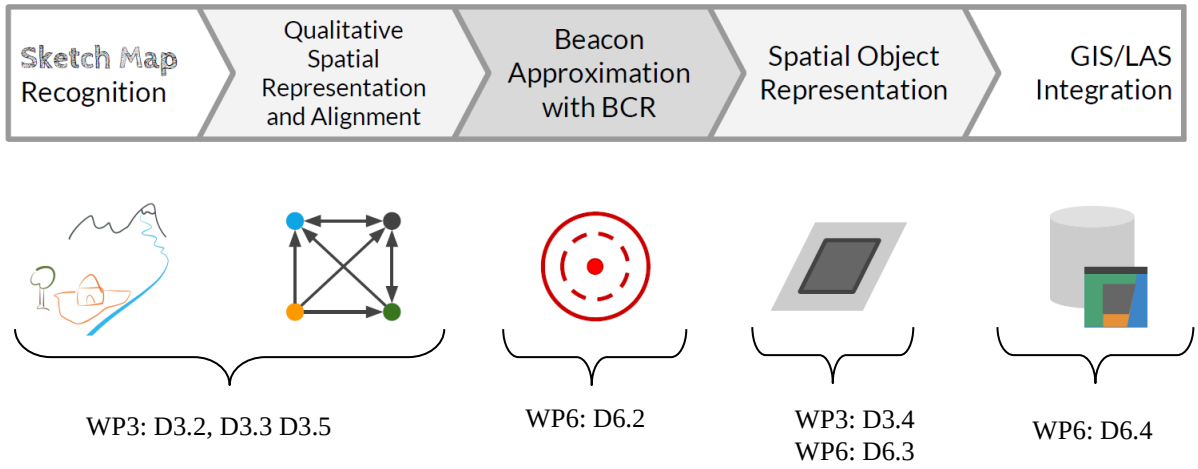


Figure 10 illustrates how various deliveries are combined to form the entire tool chain of Draw and Make.
Deliverable 6.2 constitute the step of approximating the aligned sketch maps to georeferenced geometries. This is the last step in the Draw and Make Tool before the geometries are passed to the Publish and Share platform and stored as SpatialUnits (see **Figure 10**).

Previous steps in the workflow aimed at linking objects drawn in sketch maps with georeferenced objects on a metric map, thus providing us with a list of objects in the sketch maps matched to their metric map equivalent. What is missing is the spatial extent of the area of interest, bordered by beacons represented on the sketch map. Determining its boundary is of particular interest for land administration purposes. Owing to the qualitative nature of the description, the boundaries tend to be ill-defined, and difficult if not downright impossible to pinpoint accurately. The goal of geometric approximation is to come up with a possible boundary using such beacons. We can attempt to identify the minimal and maximal boundaries of such regions by first identifying the core regions where the surrounding beacons are anchored. The three steps in the workflow to arrive at the geometric approximation are briefly described below:

1. Creation of the beacon core region (BCR)

Since the location of the beacons themselves are not precise, the BCR serves as the core region where the beacon is anchored. In order to demarcate the BCR, the following steps are performed:

a. Tessellation –Both the sketch map and the metric map are tessellated (partitioned) to obtain one tile for each matched object. Given a beacon and the tile it overlaps with in the sketch map, we can find the corresponding tile in the metric map (the tile of the equivalent matched object). Depending on the tessellation method used, a tile corresponds to a spatial property of the object it contains e.g. Voronoi polygons correspond to the notion of distance. To obtain an adequate BCR, we also need tessellations which take into account other properties such as alignment. In this way, a number of tessellations are carried out e.g. :

   - Tessellations using the Voronoi diagrams
   - Tessellation using the StarVars calculus[29]
   - Using other calculi

b. Voting of BCR polygons – If a beacon overlaps a tessellated sketch map tile, and there exists an equivalent metric map tile, the latter gets a vote as a candidate location for the beacon. This voting process is then repeated for the different tessellations and for all beacons. The end result of voting is a set of candidate regions (tiles) for each beacon.

c. Selection of BCR from voted areas - the intersection of all candidate regions for a given beacon is considered to be its smallest anchor and forms the BCR of the given beacon. BCRs may be refined further by considering qualitative spatial constraints between unmatched objects.

2. Identifying minimal and maximal boundaries

Once the BCRs have been identified, the boundaries are constructed by taking the outer concave hull of the BCRs which border the area of interest. The minimal extent is obtained by a polygon with minimal area which touches the surrounding BCRs. The use of minimal and maximal boundary extents corresponds to the egg-yolk model[24], which is a model to represent and reason about indeterminate boundaries.

3. Transformation to raster

The egg-yolk model results in two boundaries. This is transformed to a raster to obtain a field based representation. The raster cells within the minimal extent belong to the core region that is definitely in the area of interest. The cells between the minimal and maximal extents correspond to the upper bound of the area. Everything outside of the maximal extent is not considered part of the feature.

6 Operation of the image-processing system

The following section discusses operational aspects of the image-processing system. This includes its integration with the other work packages of its4land and potential deployment strategies.

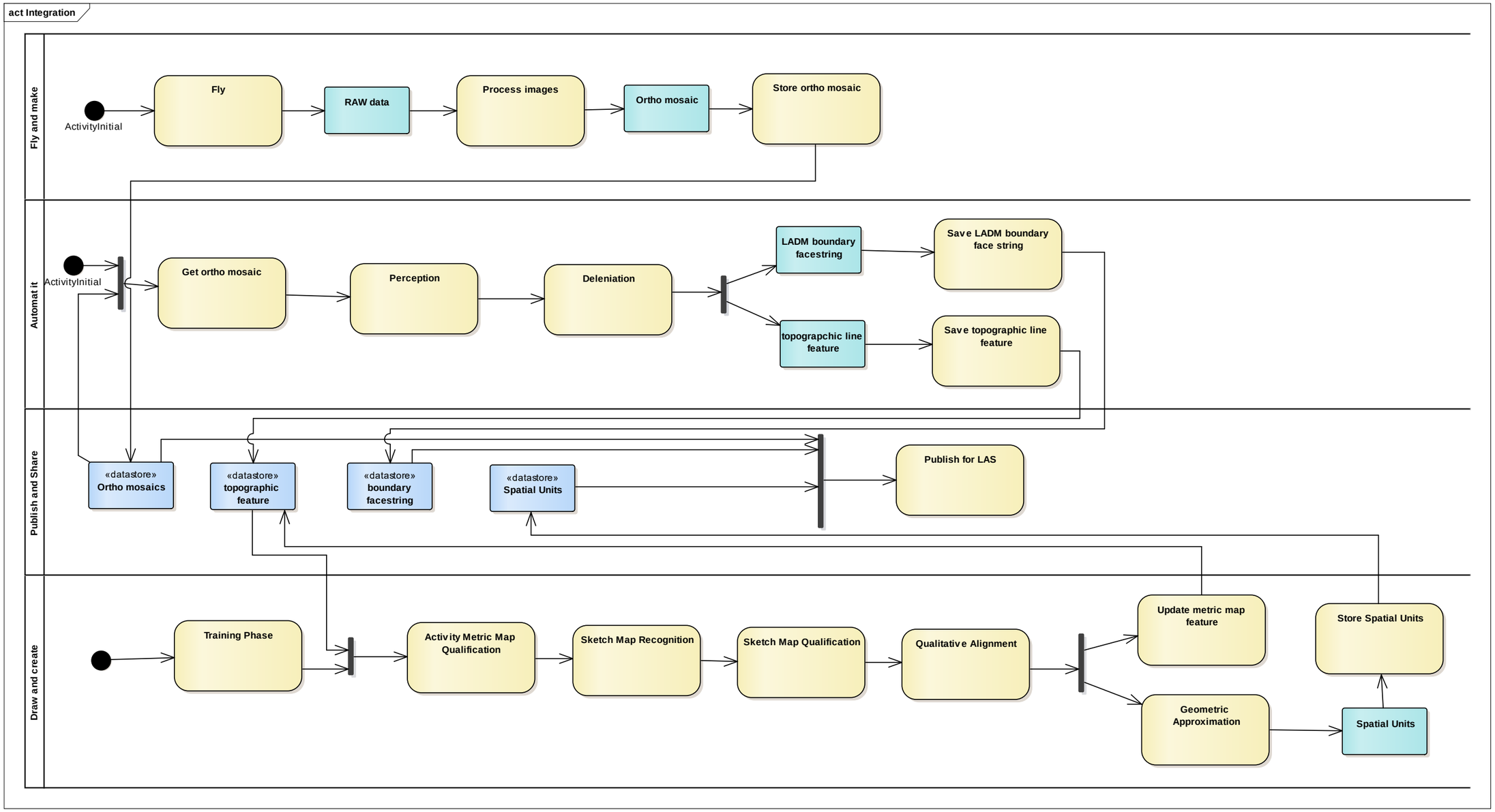
6.1 Integration with other work packages

The overall aim of its4land is the development of a set of tools to support land tenure registration in developing countries, by introducing innovative methods.

Figure 11 shows the relationship between the different technical work packages of its4land. The qualitative data processing system is primarily based on qualitative data which are captured and processed on the platform itself. Draw and Make uses additional topographic features to identify and locate sketch maps in the alignment process. These features are served and maintained by the Publish and Share platform. Most of these features are imported from external sources. Additional linear features are provided by the Automate it tool. The features are by default unclassified. During the alignment process, Draw and Make can identify some previously unclassified topographic linear features and update their classification. This will increase the quality of the alignment process over time.

The main output of Draw and Make are potential Spatial Units according to the LADM standard [30]. Publish and Share makes these boundaries available to an LAS. In the LAS these Spatial Units can be used in the registration process to document land tenures.

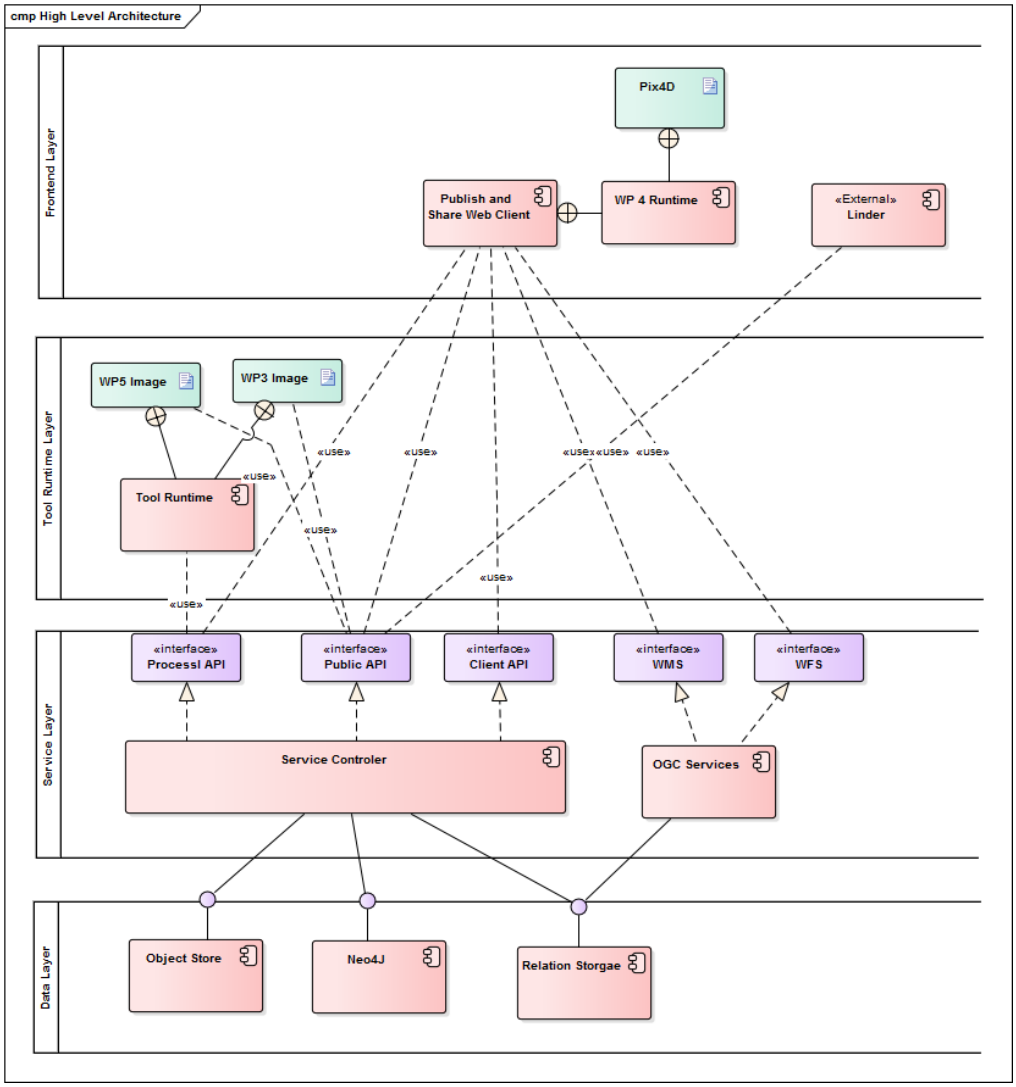
Figure 11: Relationship between the technical work packages of its4land



6.2 Deployment

The preferred usage scenario for Publish and Share is its online use in the geocloud. The developed architecture however also supports a complete offline on-premise deployment and use, e.g. for data processing while in the field or on a mission.

Figure 12: Layered architecture of Publish and Share



Depending on the usage scenario, two ways of deployment exist:

Mobile (offline):

- All Layers are deployed on a single (self-contained) infrastructure (e.g. a laptop)
- Hyper-converged infrastructure (A software-defined IT infrastructure that virtualizes all of the elements of conventional "hardware-defined" systems)

Geocloud (online)

- Data and Service layer implemented on the cloud service
- Tool Runtime layer implemented on the cloud service
- Front-end layer (Publish and Share Web Client) served by cloud service, executed on local front end machines
- Front-end layer (Desktop or external tools) installed and executed on local front end machines.
- Demonstrator on Amazon Web Services

The source code will be available on the its4land GitHub account. This repository is under active development and represents actual source code on a daily base. The repository is the source for the build process, which compresses the source code. The resulting "built" version of the Publish and Share demonstrator can be deployed to the test environment. The test environment is going to be implemented on the Amazon Web Services infrastructure.

At the time of writing this report the infrastructure will consists of these building blocks:

- 1 EC2 Instance for the Service Layer.
- 1 EC2 Instance for the Runtime Layer
- 1 RDS Instance for hosting the PostgreSQL data base
- 1 EC2 Instance for hosting the shared file system and Neo4J
- 1 S3 bucket as object storage

The current source code is optimized for operating in the selected cloud environment. It needs to be adapted for on premise deployment or deployment on other cloud environments. Since the complete source code is available, this necessary adaptation can be done by the operator of the target environment.

7 Conclusion

In task T6.2 we are developing a qualitative data processing system for hosting the tools developed in Work Package 3. Developed on the Publish and Share platform, it will provide a scalable runtime environment, a unified set of services for data management and interaction with tools developed in other work packages.

The most challenging part was the definition of an architecture that allows both, mobile offline and geocloud online use. Most parts of the qualitative data processing require significant computing capacities and hardware devices (large format scanner) that are not suitable for in-field usage. The importance of a mobile (offline) usage of the platform seems to be decreasing. Therefore we recommend to focus the future development on the geocloud online model.

Currently the testing and deployment capacities for Publish and Share and the qualitative data processing system are very limited, because Work Package 6 has no Budget for acquiring its own hard- and software or use a commercial cloud provider. Currently Publish and Share is going to be implemented on an Amazon Web Service infrastructure. The financing of the operation still needs to be clarified. We recommend providing a budget for an appropriate testing and deployment infrastructure.

The use of the Amazon Web Services as infrastructure is in line with the geocloud approach. From a purely technical perspective, the Amazon infrastructure works very well. Nevertheless, we recommend evaluating other hosting alternatives. Various factors will influence such an evaluation. For example, without claim to completeness:

- The costs of an alternative (both for implementation, and on for operation). As well as the balance between costs and benefits.
- Technical suitability and the planned usage scenario.
- Transparency and security issues.
- Privacy and compliance policies.

The qualitative data processing requires topographic data for the alignment of the sketch maps. For the actual demonstrator, the topographic data are captured and classified based on the OpenStreetMap feature catalogue. We recommend developing a custom ontology for the topographic features used in the alignment process.

The development of the architecture was impeded by lack of clear usage scenarios. Who will use the system? What is the concrete problem to solve? We recommend defining several problem oriented use cases that clearly describe a business problem and how to solve it with the its4land tools.

The Publish and Share platform will be developed continuously together with the other work packages. Focus will be on the stabilization and improvement of the API and on the GUI to enhance the user experience.

The next two deliverables of Work Package 6 focus more on the dissemination aspect of Publish and Share.
Deliverable 6.3 will extend Publish and Share to handle the approximated geometries from the qualitative data processing system according to the LADM extension developed in Work Package 3.
Deliverable 6.4 will define a workflow and interface to publish the output of the different work packages to a LAS. This will also include the common GUI in Publish and Share to integrate the different tools.

8 Bibliography

[1] J. Sahib, C. Murcia, M. Chipofya, A. Schwering, C. Schultz, B. K. Alemie, R. Wayumba, and C. Timm, “Semantic recognition of sketched objectse. Its4land Deliverable 3.2,” Enschede, 2017.

[2] N.N, “Implementation of qualitative representation of sketchmap. Its4land Deliverable 3.3,” Enschede, 2018.

[3] N.N, “Implementation of sketchmap alignment prototype with existing map. Its4land Dekiverable 3.5,” Enschede, 2018.

[4] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” University of California, Irvine, 2000.

[5] SmartBear Software, “OpenAPI Specification Version 2.0,” 2014. [Online]. Available: https://swagger.io/specification/v2/.

[6] “Docker Homepage.” [Online]. Available: https://www.docker.com/.

[7] D. Flanagan, *JavaScript: The Definitive Guide 6th Edition*. 2011.

[8] ECMA International, “ECMAScript® 2018 Language Specification,” Geneva, 2018.

[9] “React. A JavaScript library for building user interfaces,” 2018. [Online]. Available: https://reactjs.org/.

[10] “ExperMaps,” 2018. [Online]. Available: http://www.expermaps.de/en/.

[11] “Open Geospatial Consortium (OGC),” 2018. [Online]. Available: http://www.opengeospatial.org/.

[12] OGC, “Web Map Service (WMS),” 2018. [Online]. Available: http://www.opengeospatial.org/standards/wms.

[13] OGC, “Web Feature Service (WFS).” [Online]. Available: http://www.opengeospatial.org/standards/wfs.

[14] “node.js,” 2018. [Online]. Available: https://nodejs.org/en/.

[15] “Swagger,” 2018. [Online]. Available: https://swagger.io/.

[16] “Sequelize.” [Online]. Available: http://docs.sequelizejs.com/.

[17] “GeoServer Homepage,” 2018. [Online]. Available: http://geoserver.org/.

[18] “PostgreSQL Homepage,” 2018. [Online]. Available: https://www.postgresql.org/.

[19] “PostGIS Homepage,” 2018. [Online]. Available: https://postgis.net/.

[20] “Neo4J Homepage,” 2018. [Online]. Available: https://neo4j.com/.

[21] “Amazon Web Services S3 Homepage,” 2018. [Online]. Available: https://aws.amazon.com/s3/?nc1=h_ls.

[22] “Mino Homepage,” 2018. [Online]. Available: https://www.minio.io/.

[23] Internet Engineering Task Force, “RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format.” 2017.

[24] A. G. Cohn and N. M. Gotts, “The ‘Egg-Yolk’ Representation of Regions with Indeterminate Boundaries,” in *Proc. GISDATA Specialist Meeting on Geographical Objects with Undetermined Boundaries, GISDATA Series*, 1996.

[25] “Python Package Index.” [Online]. Available: https://pypi.org/.

[26] “PEP 8 -- Style Guide for Python Code.” [Online]. Available: https://www.python.org/dev/peps/pep-0008/.

[27] OGC, “OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture,” *Open Geospatial Consortium, Inc*, 2010.

[28] “OpenStreetMap Features.” [Online]. Available: https://wiki.openstreetmap.org/wiki/Map_Features.

[29] J. H. Lee, J. Renz, and D. Wolter, “StarVars-effective reasoning about relative directions,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2013.

[30] ISO/FDIS 19152, “Geographic information - Land Administration Domain Model (LADM),” vol. 2012, pp. 1–118, 2012.