Horizon 2020
European Union funding
for Research & Innovation

ITS 4 LAND
innovations for land tenure

Deliverable 6.1
Technical Report

31 July 2018

Version 1.0

Abstract:
Technical report and software prototype of the mobile image 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, Reiner Borchert, Sophie Crommelinck, Claudia Stöcker

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 D6.1 documents the outcome of task T6.1. Together with Deliverable D6.2 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 T6.1 is the development of the image-processing system. The image-processing system provides an easy to use system for boundary delineation based on UAV-based orthomosaics.

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 T6.1 adapts the results of the Work Packages 4 and 5 to the Publish and Share platform. The prototype implementation of the Automate it tool from Work Package 4 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 Automate it tool from Work Package 5 together with the tools and methods developed in Work Package 4 on the Publish and Share platform is the called image-processing system.

Contents

EXECUTIVE SUMMARY 2

ABBREVIATIONS 5

1 INTRODUCTION 6

1.1 THE PUBLISH AND SHARE PLATFORM IN ITS4LAND 6
1.2 MOBILE IMAGE-PROCESSING SYSTEM 8

2 ARCHITECTURE OF THE IMAGE-PROCESSING SYSTEM 9

2.1 OVERVIEW OF THE ARCHITECTURE 9
2.2 TECHNOLOGY STACK USED FOR THE IMAGE-PROCESSING SYSTEM 16

3 ADAPTATION OF THE AUTOMATE IT PROTOTYPE TO THE PUBLISH AND SHARE PLATFORM 18

3.1 ANALYSIS OF THE EXISTING CODE 18
3.2 REDESIGN OF THE PROTOTYPE CODE 22
3.3 RESTRUCTURING AND REFACTORING OF THE CODE 22

4 OPERATION OF THE IMAGE-PROCESSING SYSTEM 24

4.1 INTEGRATION WITH OTHER WORK PACKAGES 24
4.2 DEPLOYMENT 26
4.3 INTEGRATION OF UAV DATA FROM WORK PACKAGE 4 27

5 CONCLUSION 29

6 BIBLIOGRAPHY 31

Abbreviations

COTS	Commercial off-the-shelf
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 System
OAS	OpenAPI Specification
OGC	Open Geospatial Consortium
ORM	Object-relational mapping
REST	Representational State Transfer
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WFS	Web Feature Service
WMS	Web Map Service

1 Introduction

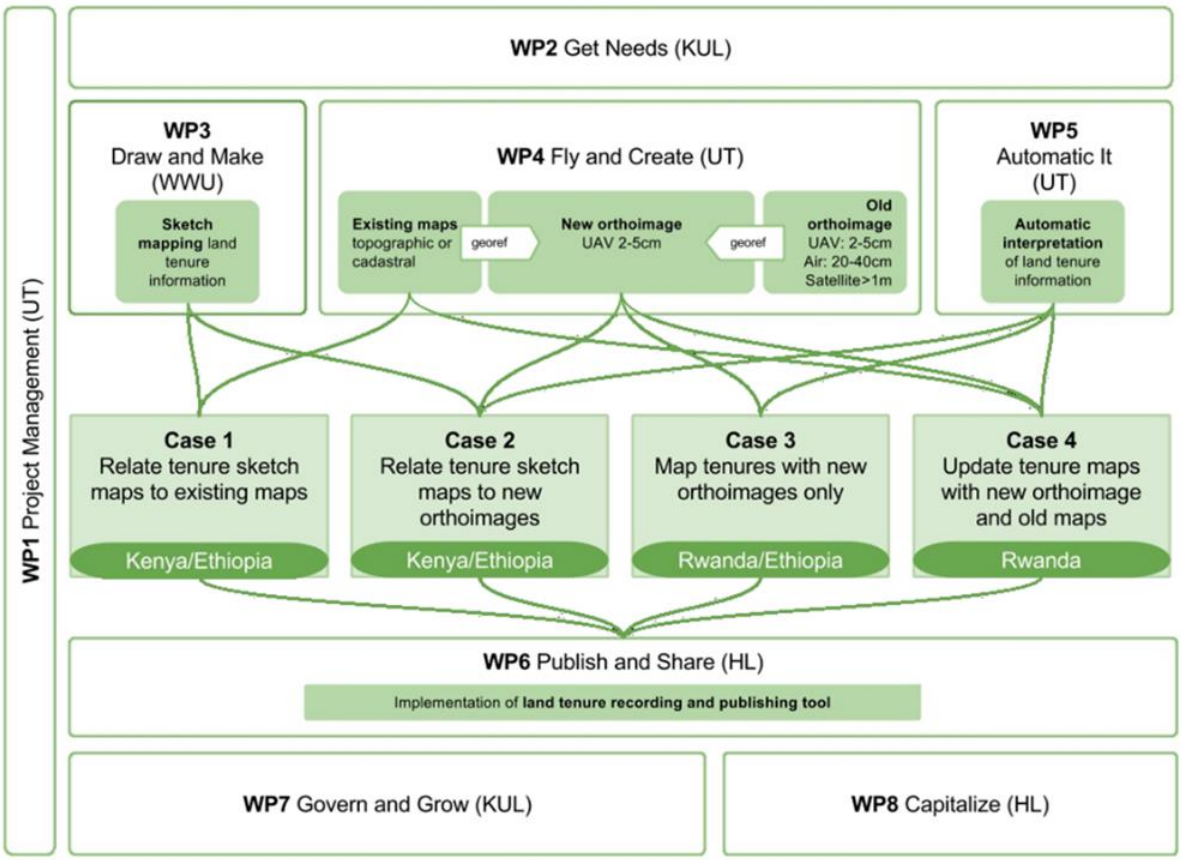
Its4land is a European Commission Horizon 2020 project funded under its Industrial Leadership program, specifically the 'Leadership in enabling and industrial technologies – Information and Communication Technologies ICT (H2020-EU.2.1.1.)', under the call H2020-ICT-2015 – and the specific topic – 'International 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 adoapt 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 image-processing system

The image-processing system is a subset of the entire Publish and Share platform. The aim of the mobile image-processing system is to provide an easy to use system for boundary delineation. The image-processing system allows the in-field or online boundary delineation based on UAV-based orthomosaics. The UAV images are captured and processed by photogrammetric tools and methods as part of the workflow in Work Package 4 (Fly and Create). For details about the integration of Fly and Create into the image-processing system and the Publish and Share platform see section 4.3.

Work Package 5 will deliver methods for (semi) automated extraction of landmarks and boundary delineation from UAV data. The Publish and Share platform provides a framework for integrating these algorithms into standard open source GIS. The in-field processing is an important aspect of the image-processing system. This allows direct and immediate participation of local people. The image-processing system is built on an architecture of the Publish and Share platform that combines a mobile in-field usage with a geocloud environment. Depending on the available infrastructure and the concrete needs, the image-processing system can be deployed in different ways to set the focus more on mobile in-field or geocloud characteristics.

2 Architecture of the image-processing system

The following section describes the image-processing system architecture. The image-processing system is created on the Publish and Share platform. The architecture of the Publish and Share platform is described as far as it is necessary for the understanding of image-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 of the architecture

The image-processing system integrates the workflow developed by Work Package 5 into the technical framework of the Publish and Share platform developed in Work Package 6.
Publish and Share provides the necessary infrastructure to operate the software prototype described in D5.1 [1] in a mobile and geocloud environment. This prototype implements the methodology of the image-processing system.
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, geo spatial, binary and image data
- OGC services for spatial data access
- Authentication and authorisation services

The services are exposed via Representational State Transfer (REST) API [2]. REST is an architectural style for distributed systems, 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) [3] 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 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 in the runtime environment.

Figure 2: High level architecture of the image-processing system.

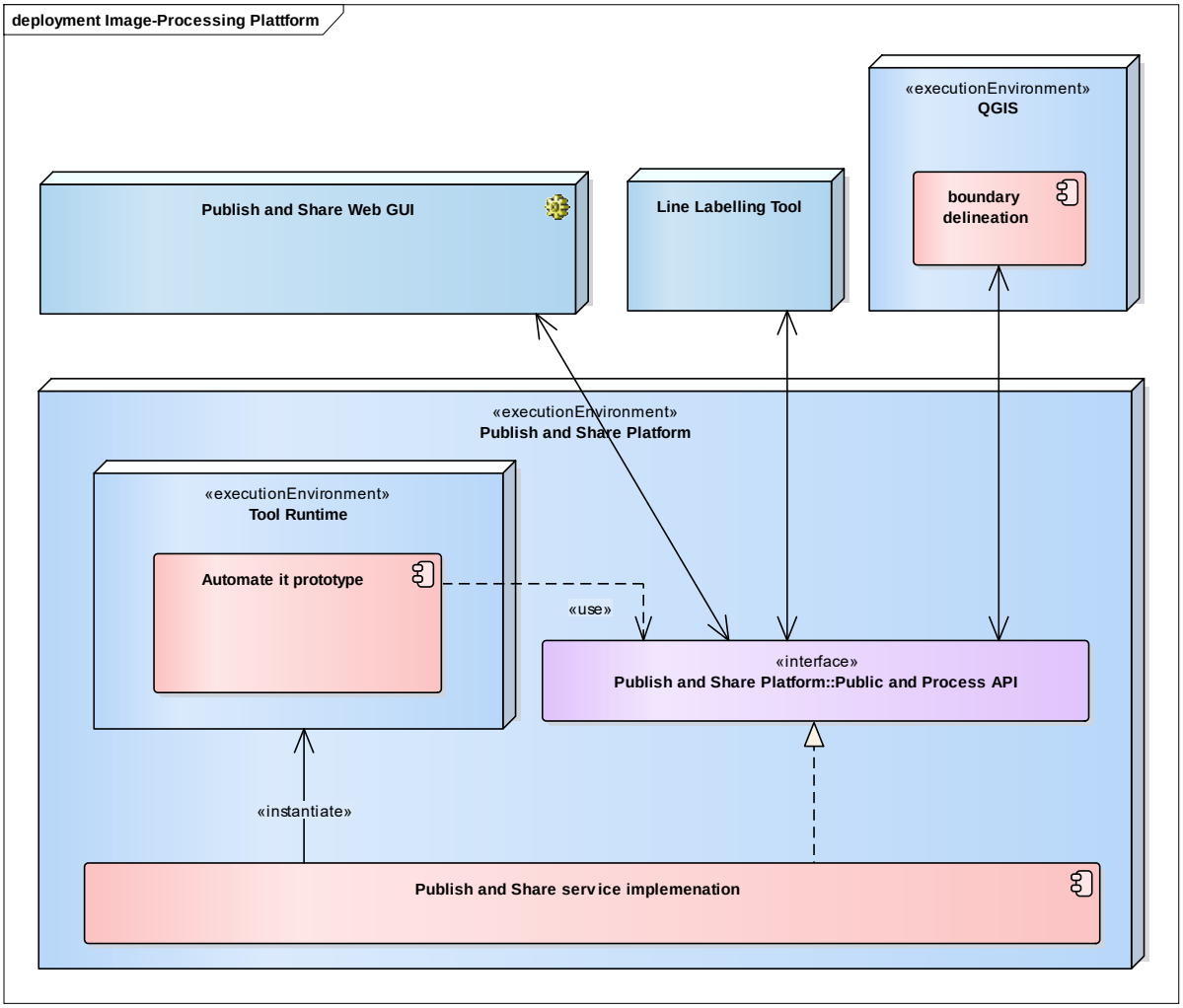


Figure 2 provides a high-level overview of the Automate it prototype implemented as part of the image-processing system on the Publish and Share platform. The main part of the image-processing system is implemented and hosted in the tool runtime environment. This includes all the computational parts for data pre-processing and machine learning. Two additional components are implemented as external applications:

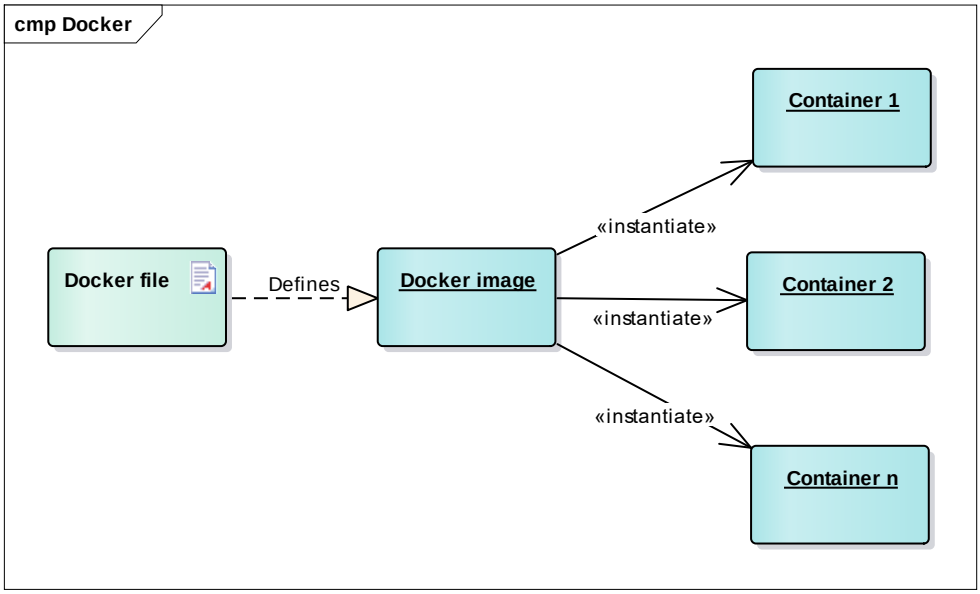
- Labelling of training set data as *boundary* or *no boundary*
- Interactive boundary delineation

The labelling tool will be implemented as a standalone web application. The interactive boundary delineation tool is implemented as a QGIS plugin. Both tools use the Publish and Share API for retrieving data und storing results.

Primarily, the qualitative data processing system uses base-functionality provided by the Publish and Share platform as services. The API allows the Automate it 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.

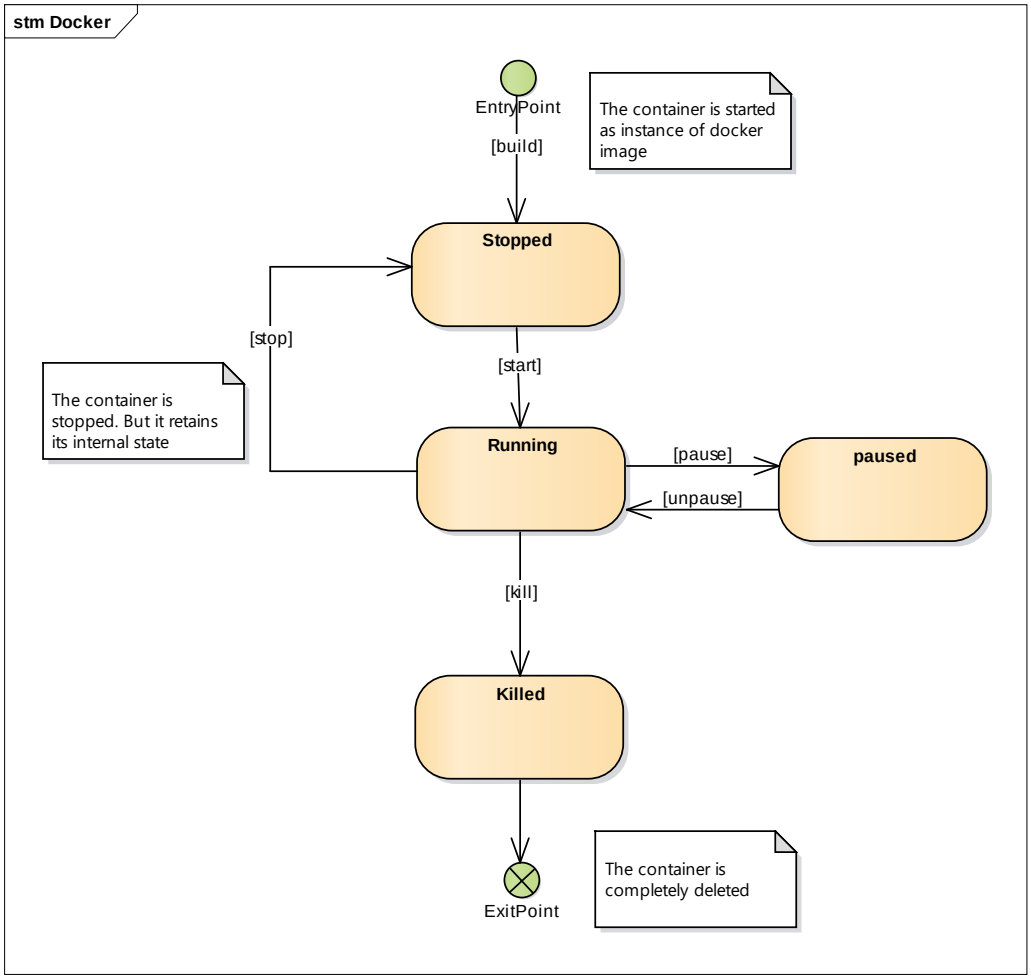
The runtime environment is based on “Docker” [4]. Docker is used to run software packages called "containers". In a typical example use case, one container runs a web server and web application, while a second container runs a database server that is used by the web application. In another example, multiple 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 in to the system for the duration of the run.

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



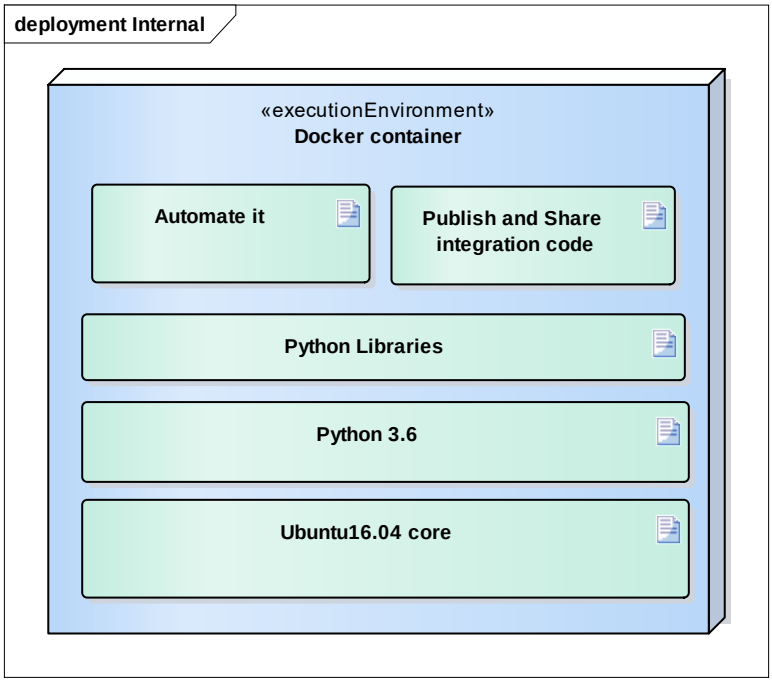
The platform provides a repository of predefined Docker images. The computational part of the image-processing system is implemented in one Docker image (see **Figure 5)**.

A Docker image contains furthermore 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 Automate it 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 Automate it container

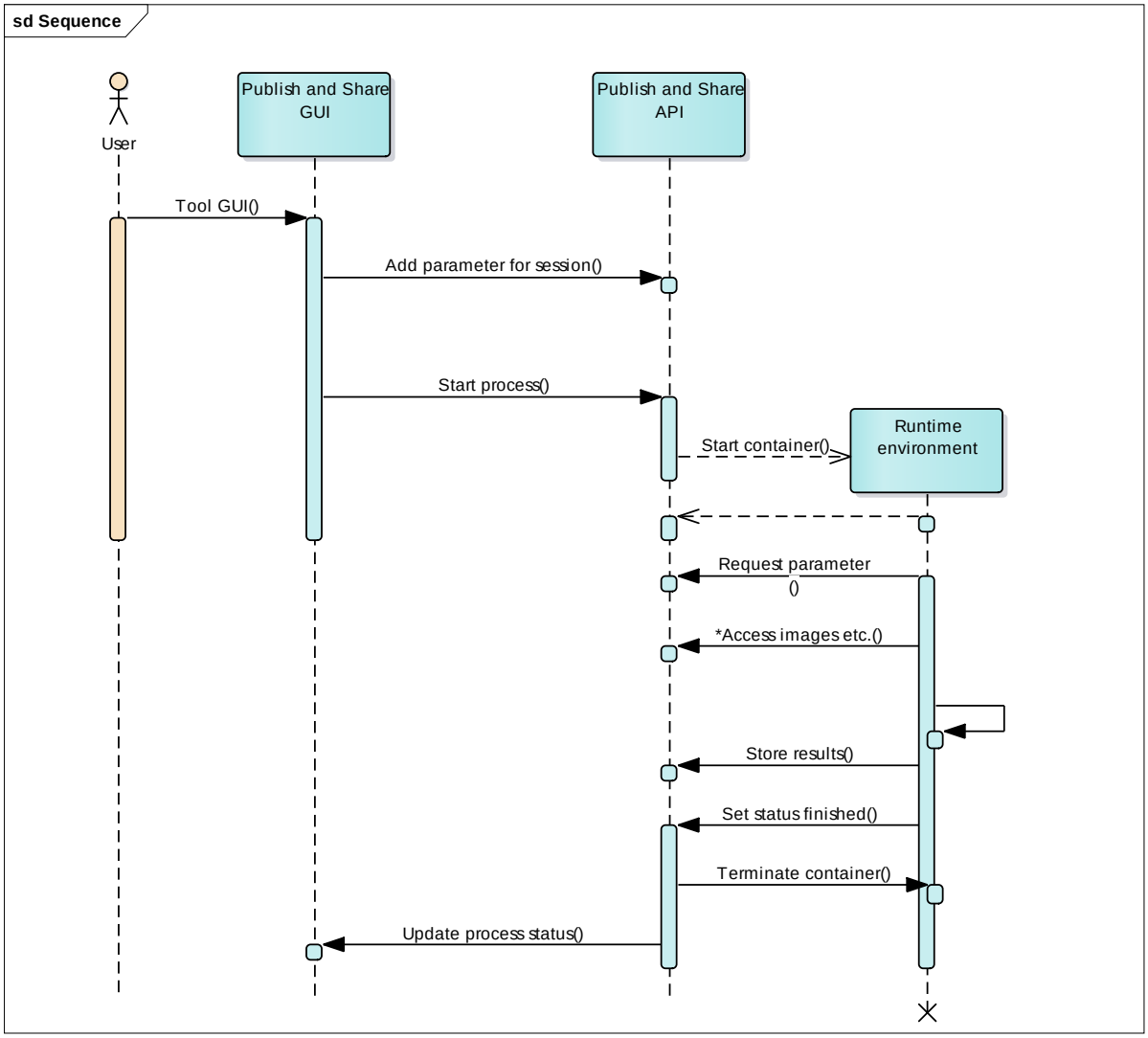


The interaction with the Automate it tools in the container is realised by a GUI implemented in the Publish and Share web client. For Automate it the GUI allows following interactions (see **Figure 8**):

- Create a training set
- Create a classifier
- Predict boundaries

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

Figure 6: UML Sequence Diagram showing the start of the Automate it 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.

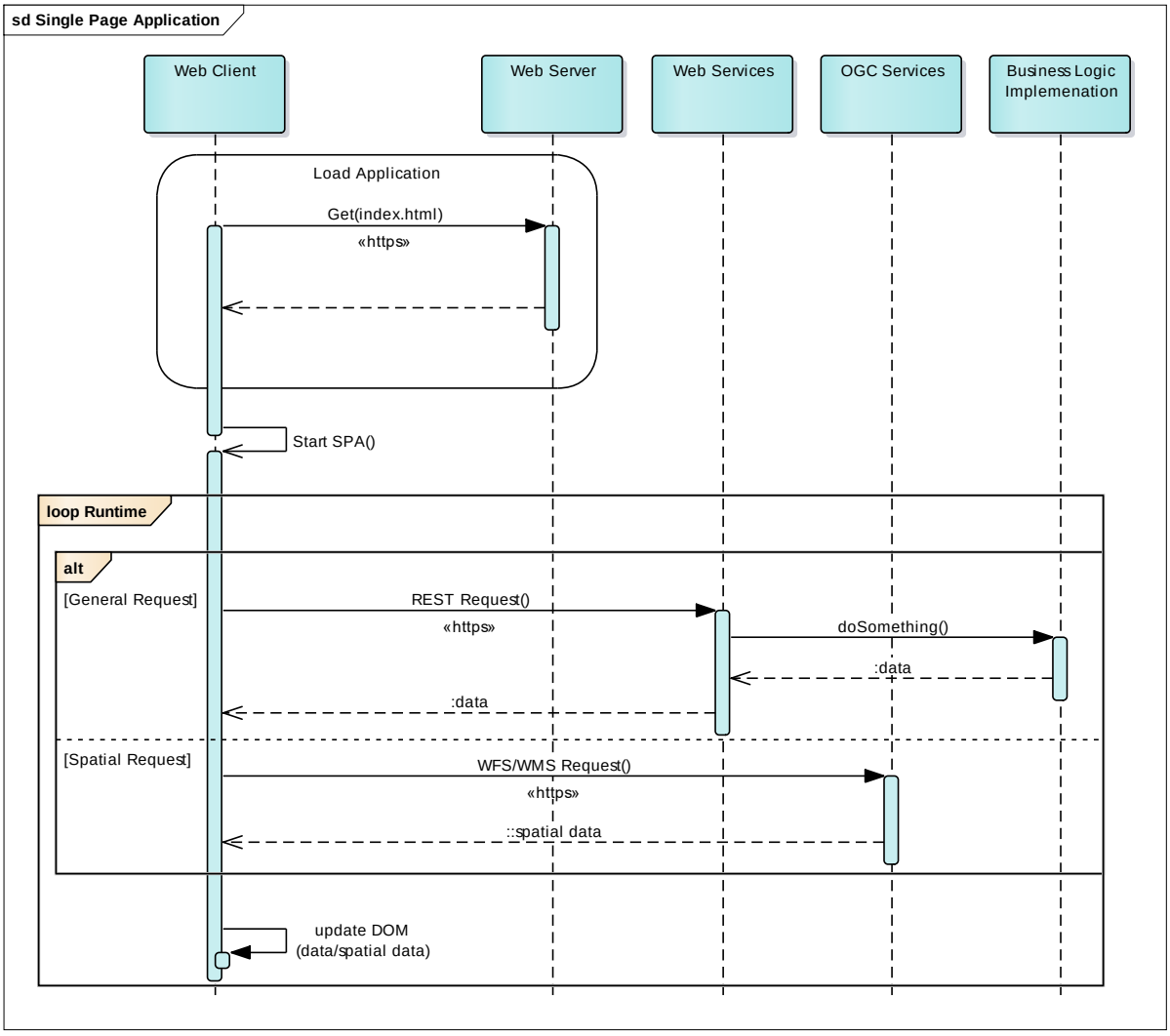
The external applications from Automate it, the labelling and interactive delineation tool, are not executed under the control of Publish and Share. However, both tools use the Publish and Share API to access required resources. This includes authentication and authorisation, as well as tool specific data like training or validation sets.

2.2 Technology stack used for the image-processing system

The technology stack of the image-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**) [5]. The implementation is done in JavaScript [6] based on the JavaScript frameworks React [7] and the web mapping framework ExperMaps [8] to visualize geo data from OGC [9] services like WMS[10] and WFS [11].

Figure 7: UML Sequence Diagram of a SPA Web Application



Besides using the main SPA client, the image-processing system also includes the QGIS Plugin and the Line Labelling tool. The Line Labeling tool is planned by Work Package 5 for future work. The QGIS [12] plugin is developed using Python [13]. The QGIS plugin exists in two version. The original plugin developed in the frame of Deliverable 5.1 was developed

on QIS 2 with Python 2.7. As part of the adaptation for Publish and Share, the plugin is currently migrated to QGIS3 and Python 3.6.

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 [6]. 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 [8]: 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) [3]
- Sequelize [16]: Object-Relational-Mapper (ORM)
- The OGC services are implemented on GeoServer [17]

Data Layer: The data layer provides four kinds 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 Automate it prototype to the Publish and Share platform

In the following section we will describe how the existing prototype developed in the Automate it work package has been adapted to the Publish and Share platform. The combination of the adapted Automate it prototype and the Publish and Share platform together forms the image-processing system. The adaptation of the prototype includes restructuring and refactoring the existing Python code as well as a redesign of the implementation of some steps of the algorithm. The redesign was necessary to allow the execution of the tool as an asynchronous background process.

The existing code was implemented in Python[1][13]. This decision of WP5 remains, the implementation language is still Python.

3.1 Analysis of the existing code

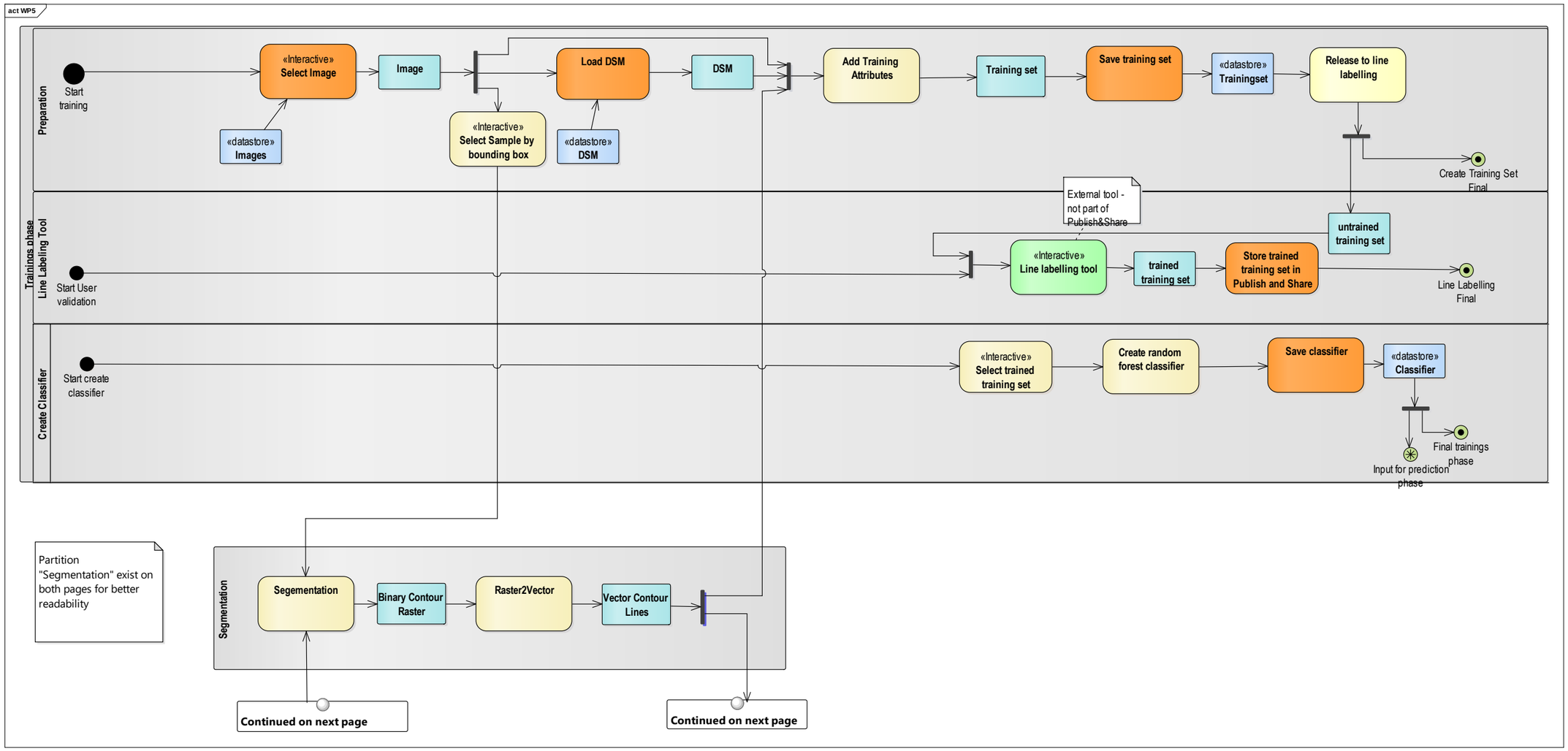
The existing Automate it prototype developed in frame of the deliverable D5.1 concentrates on the utility of the algorithm. The Publish and Share platform was not available at of development, so the initial Automate it development code could not take the Publish and Share API or the runtime environment into account.

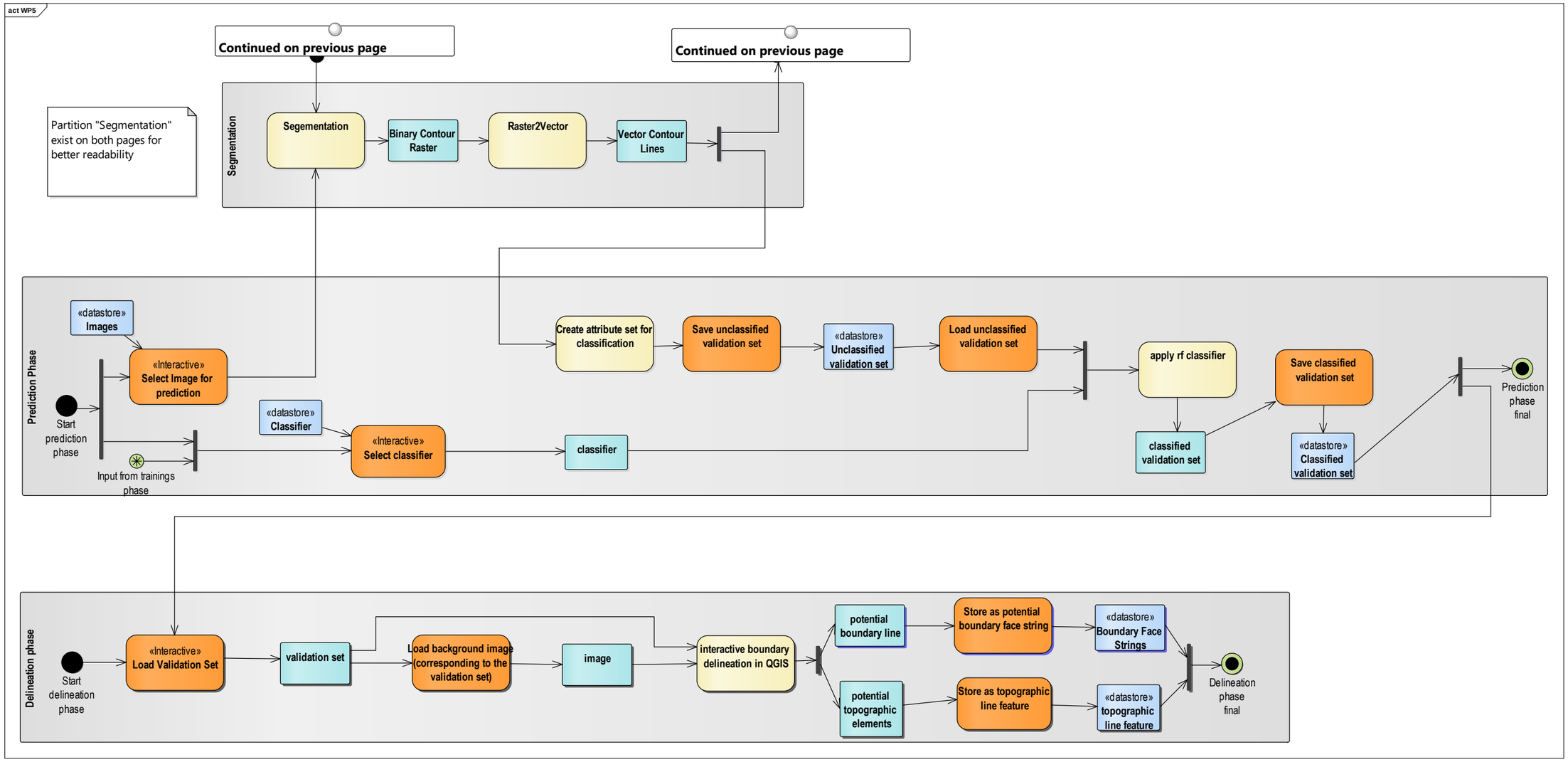
To adapt the Automate it prototype to Publish and Share, some redesign, restructuring and refactoring was necessary. The adaptation is only related to software engineering aspects, it does not affect the algorithm. The following tasks were included:

- Reorganise the code modules
- Adapt the persistence strategy
- Replace used libraries by ones that fit better to the runtime environment

In a first step, the existing Automate it prototype was documented in the form of an UML activity diagram (see **Figure 8**).

Figure 8: UML Activity Diagram of the "Automate it" algorithm




act WP5
Continued on previous page
Continued on previous page
Partition "Segmentation" exist on both pages for better readability
Segmentation
Segementation
Binary Contour Raster
Raster2Vector
Vector Contour Lines
Prediction Phase
«datastore» Images
«Interactive» Select Image for prediction
Start prediction phase
Input from trainings phase
«datastore» Classifier
«Interactive» Select classifier
classifier
Create attribute set for classification
Save unclassified validation set
«datastore» Unclassified validation set
Load unclassified validation set
apply rf classifier
classified validation set
Save classified validation set
«datastore» Classified validation set
Prediction phase final
Delineation phase
Start delineation phase
«Interactive» Load Validation Set
validation set
Load background image (corresponding to the validation set)
image
interactive boundary delineation in QGIS
potential boundary line
potential topographic elements
Store as potential boundary face string
Store as topographic line feature
«datastore» Boundary Face Strings
«datastore» topographic line feature
Delineation phase final

The purpose of this activity diagram is not to document the algorithm in any detail, as this was done in D 5.1 [1]. Instead, the purpose is to divide the existing source code into smaller pieces to identify the objects and control flows 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 the required data structures. The following interactions are included:

- Storing of results
- Requesting source data, e.g. images
- Storing and accessing intermediate results, e.g. training sets, validation sets or classifier
- Connecting data to a continuous process

Process steps 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 shows that Automate it is not a single process. The diagram is organised in four main portions (grey horizontal lanes in the Activity Diagram):

- Training set creation
- Classifier creation
- Prediction
- Delineation

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

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

The already existing code was structured into individual scripts that have to be executed by the user. Since much of the original source code was to be retained, an encapsulating of the existing script promised to be the best approach. Therefore, we decided to restructure and refactor the code following the approach of one main executable with different sets of parameters.

Furthermore, there are two additional partitions:

- Segmentation
- Line labelling

The segmentation partition represents a significant section of the Automate it code that is shared by different parts of the algorithm. The line labelling lane represents the external tool for the interactive training of the training set. This tool is required for the overall process of

interactive boundary delineation, but is only weakly coupled with the Publish and Share platform. Therefore it is treated as an external tool that can be replaced by others tools without affecting the image-processing system.

3.2 Redesign of the prototype code

The existing code of the Automate it prototype needs some redesign to be used in the runtime environment. The most import redesign affects the segmentation part of the algorithm. The segmentation separates the raster data into areas of similar colour and texture, called "superpixels". The edges of these polygons are potential boundaries, but probably most of them represent other structures.

The original code uses a Matlab script for detecting the contour lines in the image. The use of Matlab in the Publish and Share environment leads to some restrictions. First, Matlab is proprietary software that requires a commercial licence for the aspired usage scenario. The nature of Publish and Share as a dissemination platform for the results of its4land does not allow to identify the individual users for named user licences. A named user licence would also contradict the concept of asynchronous background processing. A concurrent licence does not seem to cover the usage scenario of Publish and Share, which intends the use of the platform in different pilot countries. Furthermore, the use of a proprietary software contradicts the idea of open source and the public availability of the tools developed in its4land.

From a technical point of view, the use of Matlab scripts in the interactive Matlab environment does not suit the concept of the runtime environment well, since it increases the size of the Docker image significantly and therefore increases the resource consumption and the starting time of the container. Both are critical values, since the image-processing system should also operate mobile in the field, where only limited hardware resources are available.

In the code of Automate it which is used for the image-processing system, the Matlab script is replaced by the open source tool "GDAL Segmentation", a command line tool developed by Cristian Balint [24]. This tool analyses a raster file, creates superpixels of customizable size, and stores the result file in a vector format (Shape or GeoJson).

3.3 Restructuring and refactoring of the code

The existing Python code of the Automate it prototype was developed in Python 2.7, for compatibility with the current version of QGIS 2 at that time. It was recommended to implement the new project in Python 3 [13], since Python 2.7 has nearly reached the end-of-life status [25] and QGIS 3 was launched in the meantime. The original source code was not designed to be accessible in a server environment or Docker container. Intermediate data is stored in shape files, configured with hard coded absolute local paths. The tool frequently uses GRASS functions, which are part of the QGIS 2.x package. These functions expect layer files (mostly shape format) as input and produce on their part layer files as output.

The following measures were undertaken to adopt the original source code to the Publish and Share platform:

- Converting the source code to Python 3.6
- Replacement of GRASS/QGIS functions by OGR/GDAL libraries, which are accessible without a QGIS installation
- Adaption for operating inside the container and server environment
- Storing configuration in JSON [23] structures.
- Adding of logging and error handling
- Avoiding temporary and permanent shape files. Geometric data structures like training sets or validation sets are stored in GeoJSON structures.
- Removing all persistent file storage in the local file system to make the container stateless.
- Persistence of data structures done via the Publish and Share API.
- Optimization of performance, effectivity, consistency, and clearness
- Converting the QGIS plugin to Python 3.6, QGIS 3, and PyQt 5

After this measures, the Automate it tool is structured in Python modules as follows:

The control flow is implemented in the main module, which is also the entry point to access the tool. Command line parameters can be used to distinguish between the three main lanes:

- **-train**: Creation of a training set from raster data; requires 3 further input parameters (raster data, RGB data, DSM data); output: calculated training set
- **-class**: Creation of a *Random Forest* classifier; input parameter: a calculated training set; output: a classifier
- **-pred**: Prediction of the boundary probability; input: raster data, RGB data, DSM data, classifier; output: a classified validation set

The modules “Segments2Lines” and “AttributeCalculation” are used for the preparation of training and validation sets. These modules process the result of the external segmentation tool to create calculated line vectors for the determination of parcel boundaries.
The module “Classification” creates Random Forest classifiers from training sets and applies existing classifiers to validation sets.
No QGIS/GRASS functions are used anymore, instead the GDAL libraries are used.
The interactive part of the WP5 application is implemented as a plugin for QGIS, called “BoundaryDelineation”. As intended, the plugin is now configured for Python 3.6, QGIS 3.0 and PyQt 5. The user loads the underlying raster file and a classified validation set. Both resources are served by the Publish and Share API. With these resources, the user is requested to determine cadastral and topographic boundary lines.

4 Operation of the image-processing system

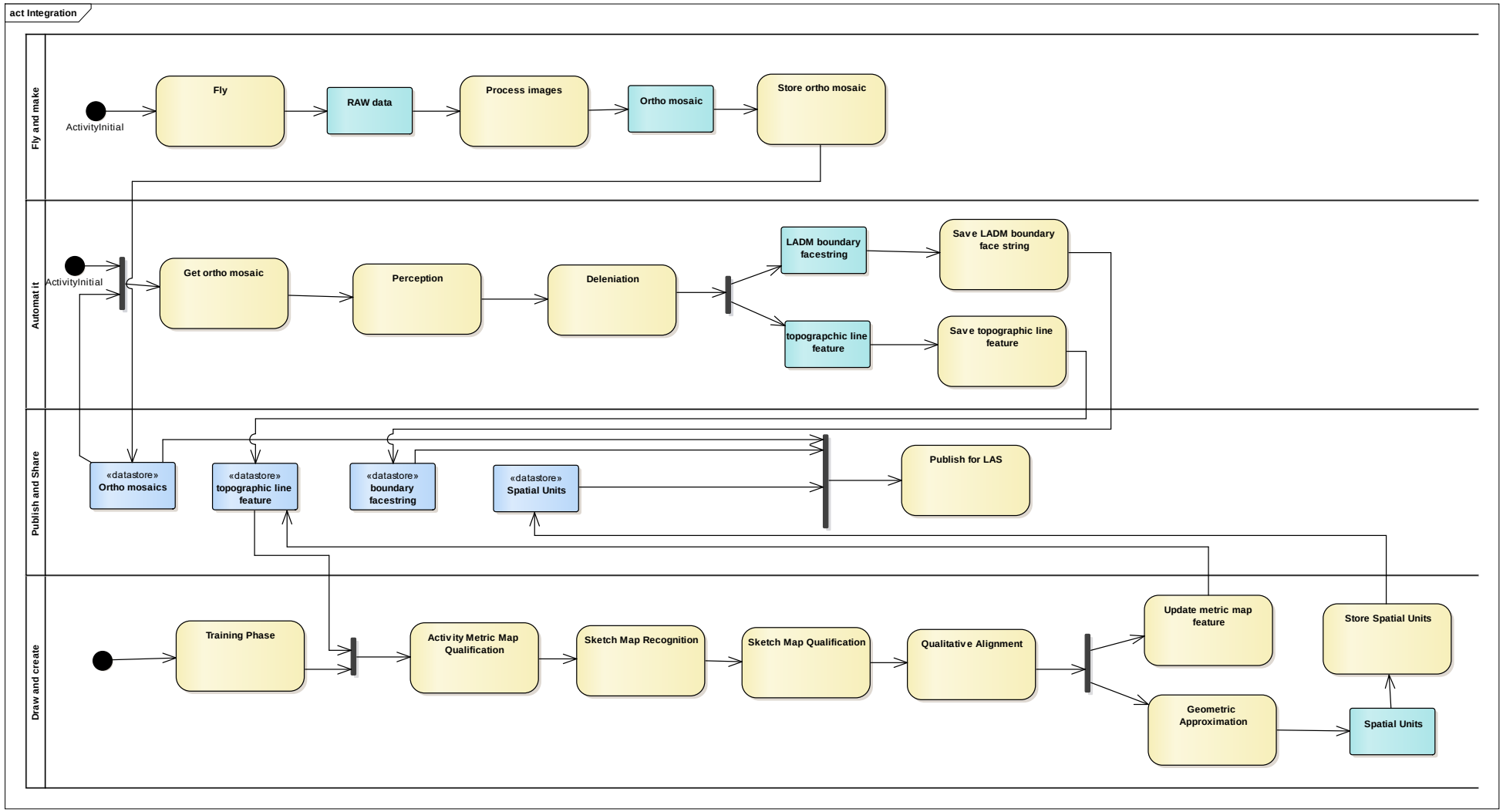
The following section discusses operational aspects of the image-processing system. This includes its integration with the other work packages of its4land and potential deployment strategies.

4.1 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.

Error! Reference source not found. shows the relationship between the different technical work packages of its4land. The image-processing system uses UAV-based orthomosaics as input for the boundary delineation. It produces a) boundary facestrings and b) unclassified topographic linear features. The boundary facestrings are boundaries according to the LADM standard [26]. Publish and Share makes these boundaries available to an LAS. In the LAS these boundaries can be used in the registration process to generate spatial units. The unclassified topographic linear features can be used in the qualitative alignment process in Draw and Make. Draw and Make uses topographic features to identify and locate sketch maps. During this process Draw and Make can identify previously unclassified topographic linear features and update their qualification. All sharing of data is done via the Publish and Share platform The exchange of data between Automate it and Draw and Make can increase the quality of the alignment process over time.

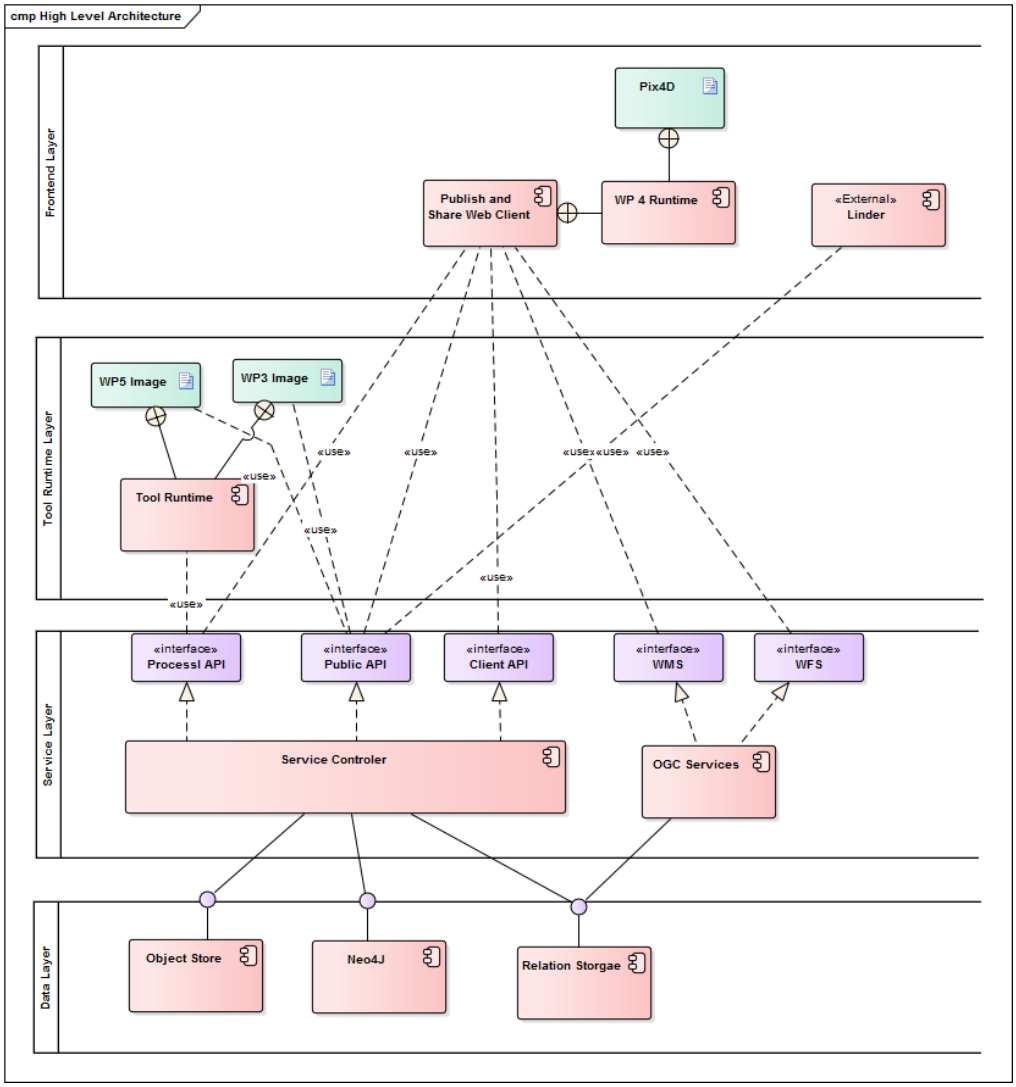
Figure 9: Relationship between the technical work packages of its4land



4.2 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 10: 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
- 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 currently implemented on the Amazon Web Services infrastructure.

At the time of writing this report the infrastructure consist 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.

4.3 Integration of UAV data from Work Package 4

The image-processing system requires orthomosaics as input for the process of boundary delineation. In the context of its4land, these input data are provided as UAV data. The UAV data are captured and processed by tools and methods developed in Work Package 4 which are considered as part of the image-processing system.

Both Work Packages are integrated via the Publish and Share platform (see **Figure 9**). The Work Package 4 has developed a workflow to process the images captured by UAV. This workflow is actually based on the COTS software Pix4D. The version of Pix4D used in its4land is a Microsoft Windows desktop version. This kind of software cannot operate in the Publish and Share Runtime Environment, since Pix4D requires a direct interaction with the user and cannot operate asynchronously in the background.
Therefore, the Work Packages 4 tools and methods are treated as part of the frontend layer. Pix4D operates as an independent client (see **Figure 10**). As a COTS software, an adaption of Pix4D to use the Publish and Share API directly is not possible. The interaction with the Publish and Share platform can be done via the Publish and Share Web Client. The Web Client can be used for upload, download and publishing data.

Work Package 4 is evaluating the alternative software OpenDroneMap [27] as a replacement for Pix4D. OpenDroneMap is an open source software for processing UAV data. The software operates on several variations of the Linux operating system and can be used without user interaction.
If Work Package 4 migrates the workflow from Pix4D to OpenDroneMap, the process of image processing could be also implemented as a tool on the Publish and Share Tool Runtime Environment.

5 Conclusion

In task T6.1 we are developing an image-processing system for hosting the tools for boundary delineation on UAV orthomosaic. 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 image-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 has still 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 image-processing system makes use of two COTS software products (Pix4D and Matlab. We recommend replacing both products by open source alternative. Fist tests for the replacement of Matlab with GDAL have been made. The results indicate that GDAL could fully replace Matlab. Work Package 4 is currently evaluating the software OpenDroneMap to replace Pix4D.

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.

6 Bibliography

[1] S. Crommelinck, M. Koeva, M. Ying Yang, G. Vosselman, S. Ho, I. Buntinx, J. Crompvoets, K. Kundert, J. Sahib, and G. Wayumba, "Technical report and software prototype on methods to automatically map features in orthoimages. its4land Deliverable 5.1," Enschede, 2018.

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

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

[4] "Docker Homepage." [Online]. Available: https://www.docker.com/.

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

[6] ECMA International, "ECMAScript® 2018 Language Specification," Geneva, 2018.

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

[8] "ExperMaps," 2018. [Online]. Available: http://www.expermaps.de/en/.

[9] "Open Geospatial Consortium (OGC)," 2018. [Online]. Available: http://www.opengeospatial.org/.

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

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

[12] "QGIS," 2018. [Online]. Available: https://www.qgis.org/en/site/.

[13] "Python," 2018. [Online]. Available: https://www.python.org/.

[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] C. Balint, “github Repository.” [Online]. Available: https://github.com/cbalint13/gdal-segment.

[25] “Python 2.7 Release Schedule.” [Online]. Available: https://www.python.org/dev/peps/pep-0373/.

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

[27] “OpenDroneMap.” [Online]. Available: https://www.opendronemap.org/.

[28] “OpenStack.” [Online]. Available: https://www.openstack.org/.