sen4cap
common agricultural policy

Sentinels for Common Agricultural Policy

Information as a Service

Introduction

1) **Project objectives and status**

2) **Python client demo**

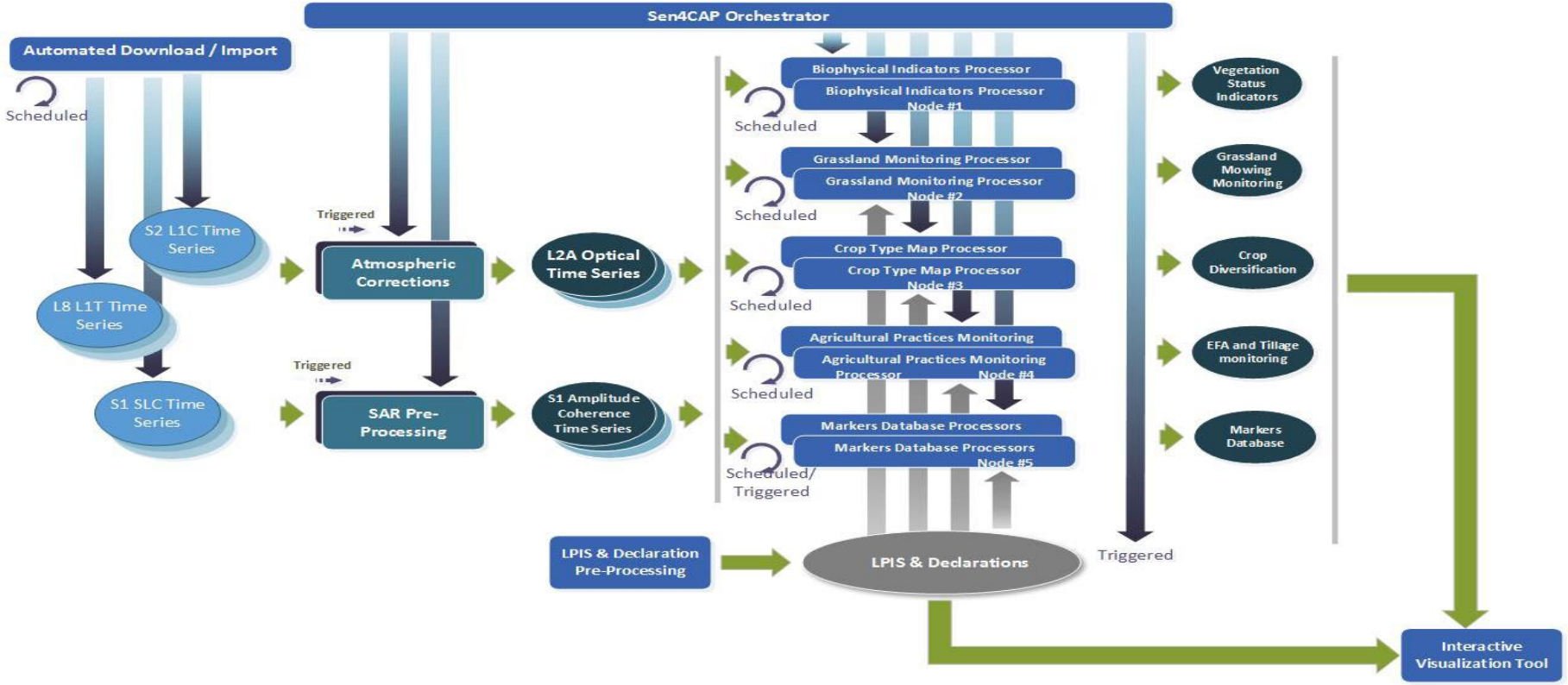
3) Scientific enhancements

4) Q&A

Project Objectives – Current Status

- **Kick-Off Meeting (KOM) – 13th of March**
- **Preliminary Design Review (PDR) – 5th of September**
- Mid Term Review (MTR) – 12 of December
- Acceptance Review (AR) - KO + 12M
- Service Showcase Demonstration (SSD) - KO + 14M
- Service Readiness Review (SRR) - KO + 16M
- Final Review (FR) - KO + 18M

Architecture of the Sen4CAP system (reminder)

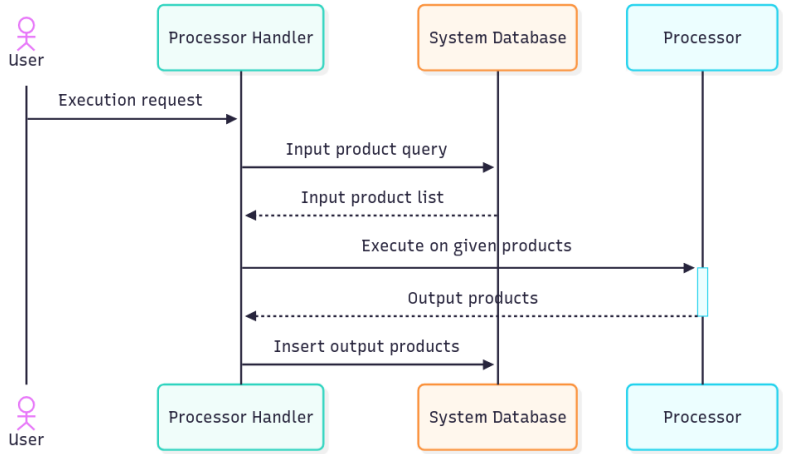


Project Objectives

- **Sen4CAP functionality as cloud-based services**
  - Convert the core functionalities of SEN4CAP into cloud-based services
    - refactor into modular cloud-based services exposed via client libraries and API interfaces
  - User access through a Python library and standard RESTData Processing APIs (OGC Processes API standard)
  - Modular design–being able to use individual functionalities independently and decouple them for deployment, scaling and maintenance
  - Containerization and microservices architecture following OGC Application Packaging best practices
- **Public cloud deployment and integration**
  - Deploy the services in a public cloud environment (co-located with Sentinel data archives)
  - Onboard the services into the ESA Network of Resources (NOR) service catalogue
- **Open-Source development**
  - Release the Sen4CAP code as open-source, but well-structured and fully documented
  - Adhere to open-source development best practices
  - Support community contributions and long-term maintenance

Project Objectives - Support for STAC

- Basically, use STAC: lightweight, extensible, popular, multiple implementations
- Keep PostGIS, ingest L2As, expose a STAC gateway (PgSTAC, stac-fastapi)
- Issues with consistency in products format
  - e.g. granule names in Sentinel-2 products (CREODIAS vs. CDAS)
- Processor handlers can query STAC instead of PostGIS
- Managing S3 access keys for DIAS buckets (L2A) vs. custom products



Project Objectives – OGC Application Packages

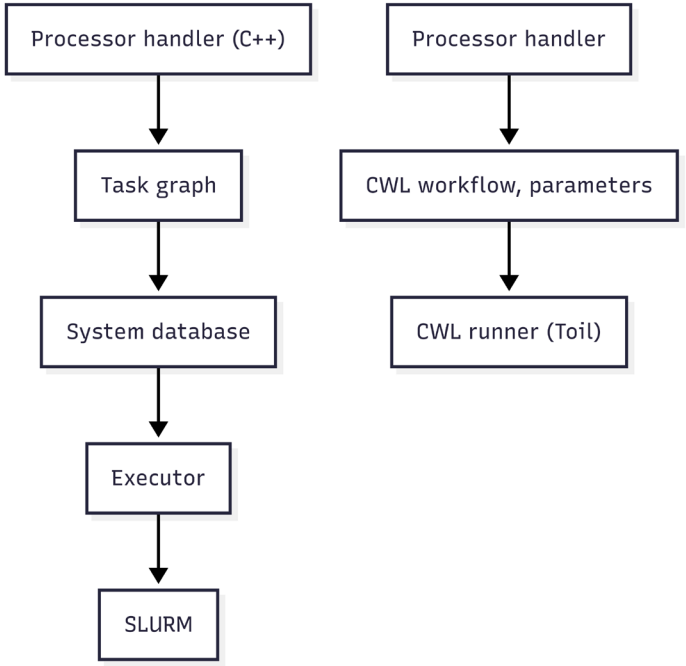
- **Porting to OGC Application Packages**
  - a way to describe, package and share geospatial applications
  - enables workflow and applications portability between systems / platforms
  - enables integration into existing platforms and catalogs (NoR, APEx)

- **Application structure**
  - packaged as OCI images
  - Common Workflow Language / CommandLineTool descriptors
  - command-line tools

Project Objectives – CWL

- We need CWL for OGC Application Packages anyway
  - replace SLURM with CWL runner, even for local deployments
  - gut out executor (cleans up redundancy, simplifies the system database)
  - might lose some job monitoring / management features

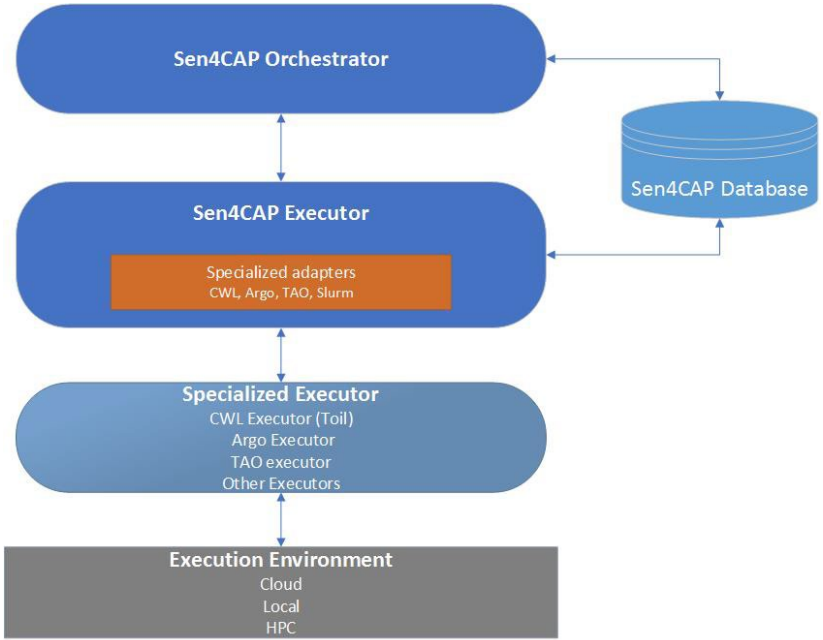
- Turn processor handlers into separate scripts (Python or anything, really)
  - support registration of new handlers via the system database



Projct Objectives - Containerisation Strategy

- Upgrade and consolidate container images
  - upgrade postgis/postgis and/or switch to imresamu/postgis
  - port processors to new version of OTB (might require custom image because of OpenJPEG fix)
  - port Python/GDAL scripts to newer GDAL (CentOS 7 Docker sandbox issue)
  - consider using a custom GDAL image (size, OpenJPEG)
  - upgrade t-rex and/or switch to successor (bbox-tile-server)
  - package crop type R classification script into new image
  - where applicable, switch to Alma (Rocky) Linux 10 (EL-compatible)

Project Objectives - Standalone deployments

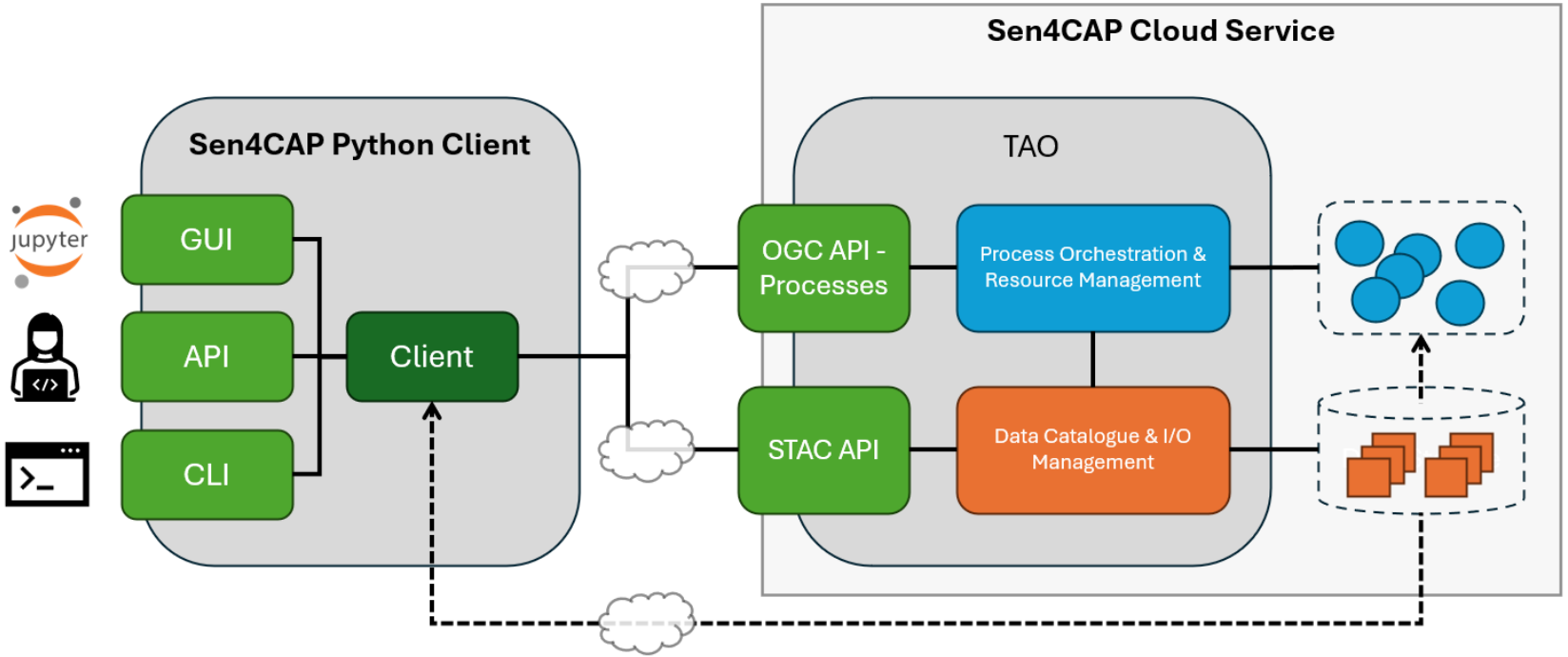


- Stand-alone installs similar to the current system
- System database, slightly simplified
- Web UI
- API (for UI, REST, OGC API Processes)
- Orchestrator
- Downloaders
- L2A processing
- S1 pre-processing
- Simplified executor (via e.g. Toil)
- MVT, STAC etc.

Cloudification Strategy

PYTHON CLIENT DEMO

Architecture



Front-end Design

- Client functionality
  - List processes, get process details
  - Submit processing request
  - List jobs, get job details, cancel job
  - Get job results
- Functionality accessible through 3 user interfaces
  - API: for Python scripts and using in own libraries
  - GUI: for convenient use in Jupyter notebooks
  - CLI: for the terminal and in shell scripts
- Implementation assumes a back-end web service compliant to OGC API - Processes - Part 1: Core

Back-end Design

- A web service compliant with OGC API - Processes - Part 1: Core
- Planned extension of the TAO API server
- Take into account EO context and data access via STAC → EOEPCA
    - Introduction - EOEPCA System Architecture
    - Introduction - EOEPCA Processing Building Block
    - Design - EOEPCA Processing Building Block
- Implementation hints:
    - OGC API Processes with ZOO-Project

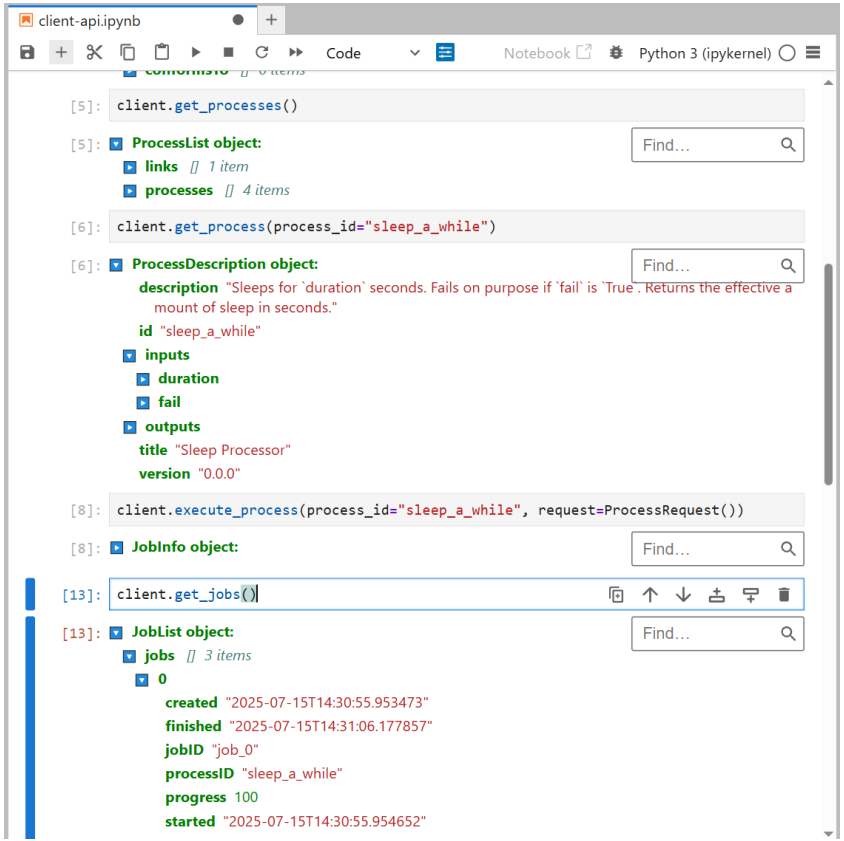
Client Implementation

- Interfaces implementations
  - API:

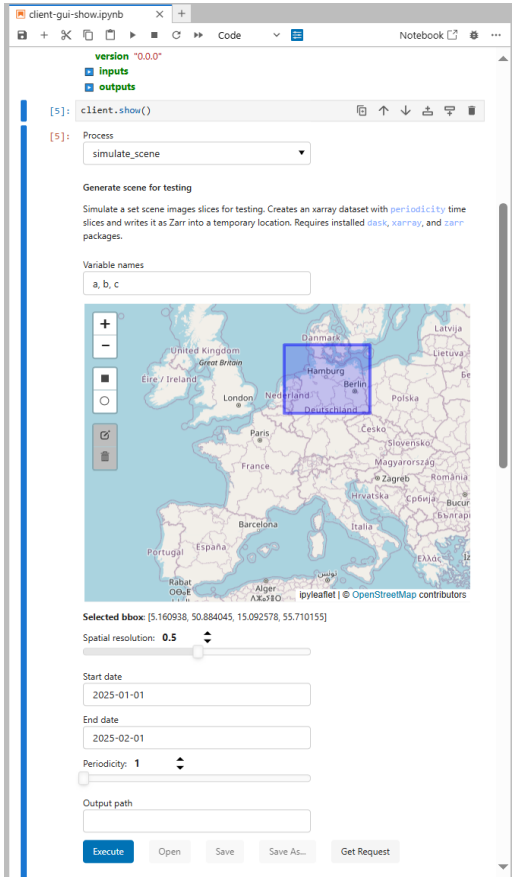
    pydantic - data validation using Python type hints,
    httpx: async multi-purpose HTTP toolkit
  - GUI: panel - powerful data exploration & web app framework
  - CLI: typer - build user-friendly CLIs from Python type hints

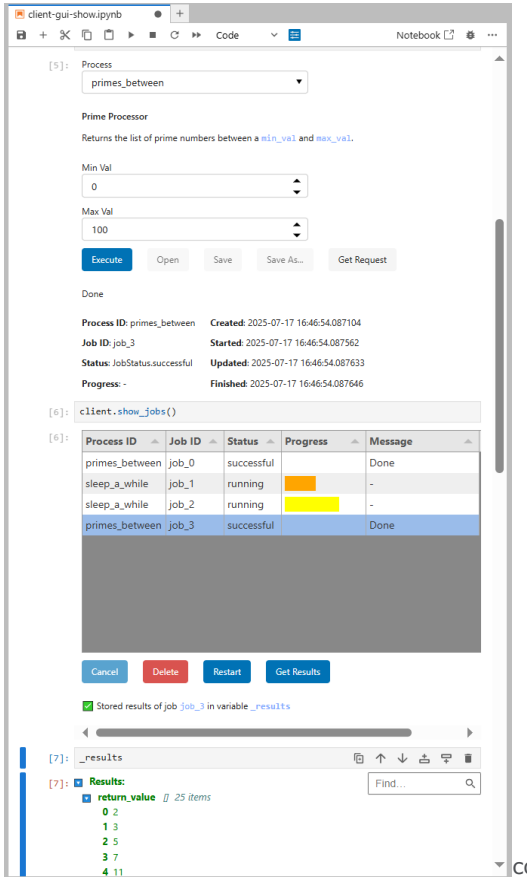
- Generate code from OpenAPI.yaml that describes the server API
  - datamodel-code-generator
  - openapi-pydantic

Python API in Jupyter Lab



Processing GUI in Jupyter Lab

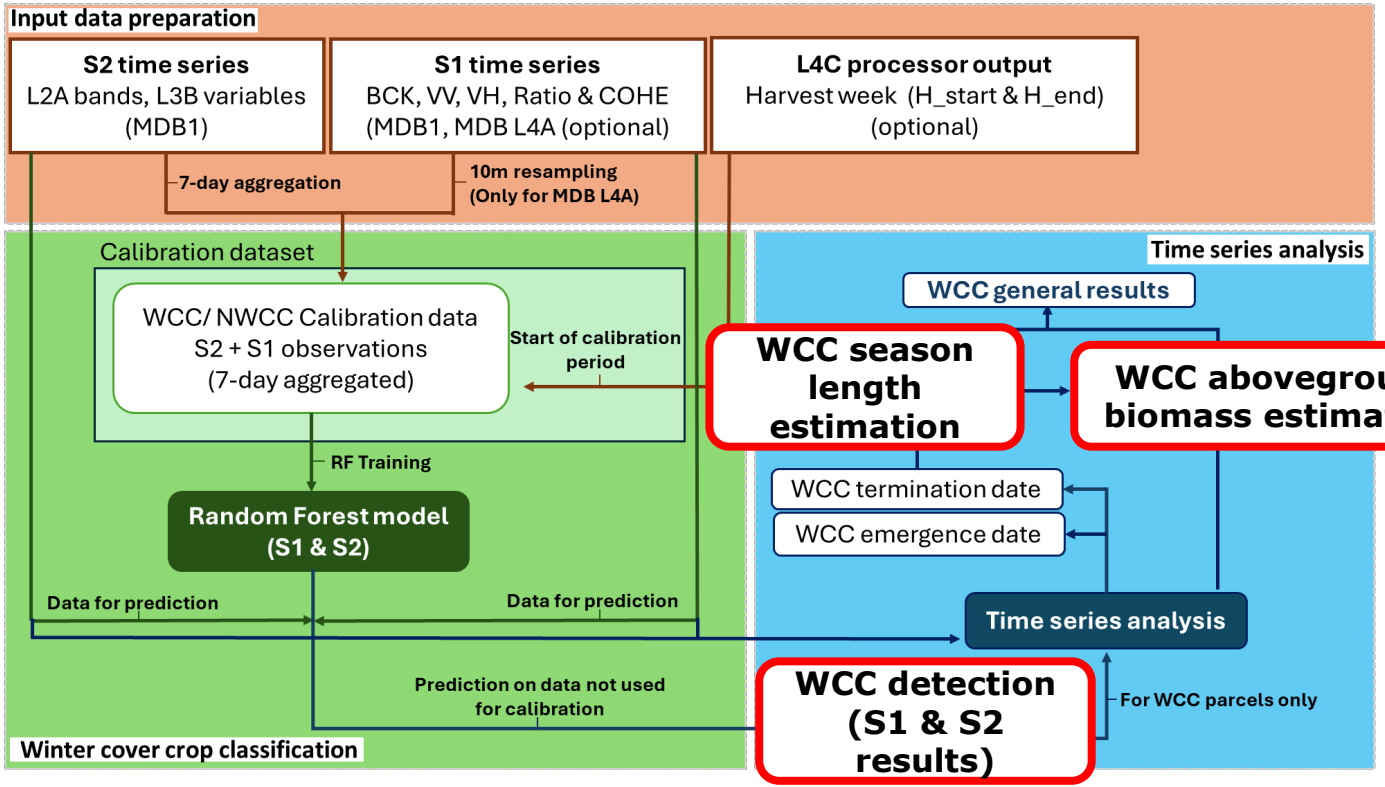




Processing CLI in terminal emulator

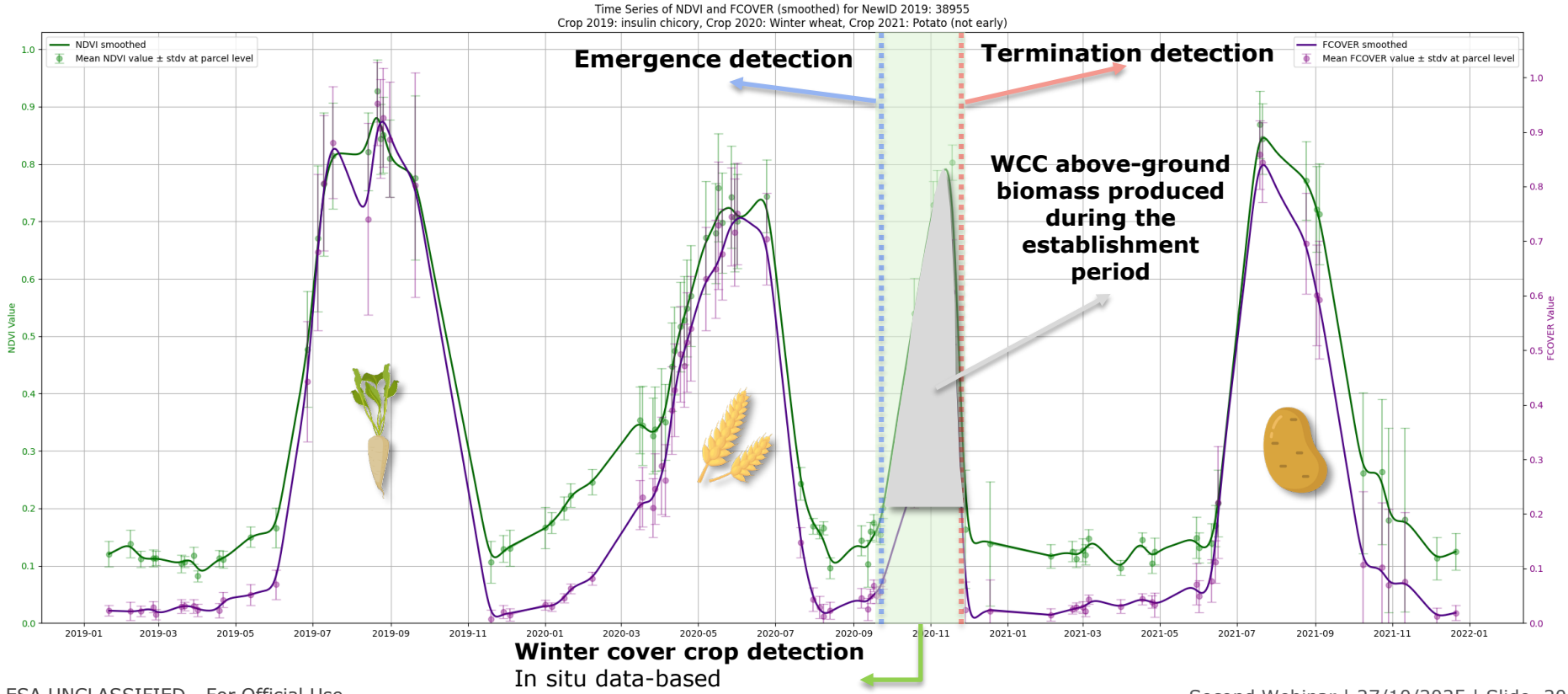
```
Usage: sen4cap-client [OPTIONS] COMMAND [ARGS]...

Client tool for the Sen4CAP service.

The tool provides commands for managing processing request templates, processing requests,
processing jobs, and gets processing results.
You can use shorter command name aliases, e.g., use command name "vr" for
"validate-request", or "lp" for "list-processes".

 Options
 --verbose                -v        Verbose output
 --install-completion               Install completion for the current shell.
 --show-completion                  Show completion for the current shell, to copy it or
                                    customize the installation.
 --help                             Show this message and exit.

 Commands
 version              Show version and exit.
 configure            Configure the client.
 list-processes       List available processes.
 get-process          Get process details.
 validate-request     Validate a processing request.
 execute-process      Execute a process.
 list-jobs            List all jobs.
 get-job              Get job details.
 dismiss-job          Cancel a running or delete a finished job.
 get-job-results      Get job results.
```

Winter cover crop (WCC) processor : workflow and expected outputs

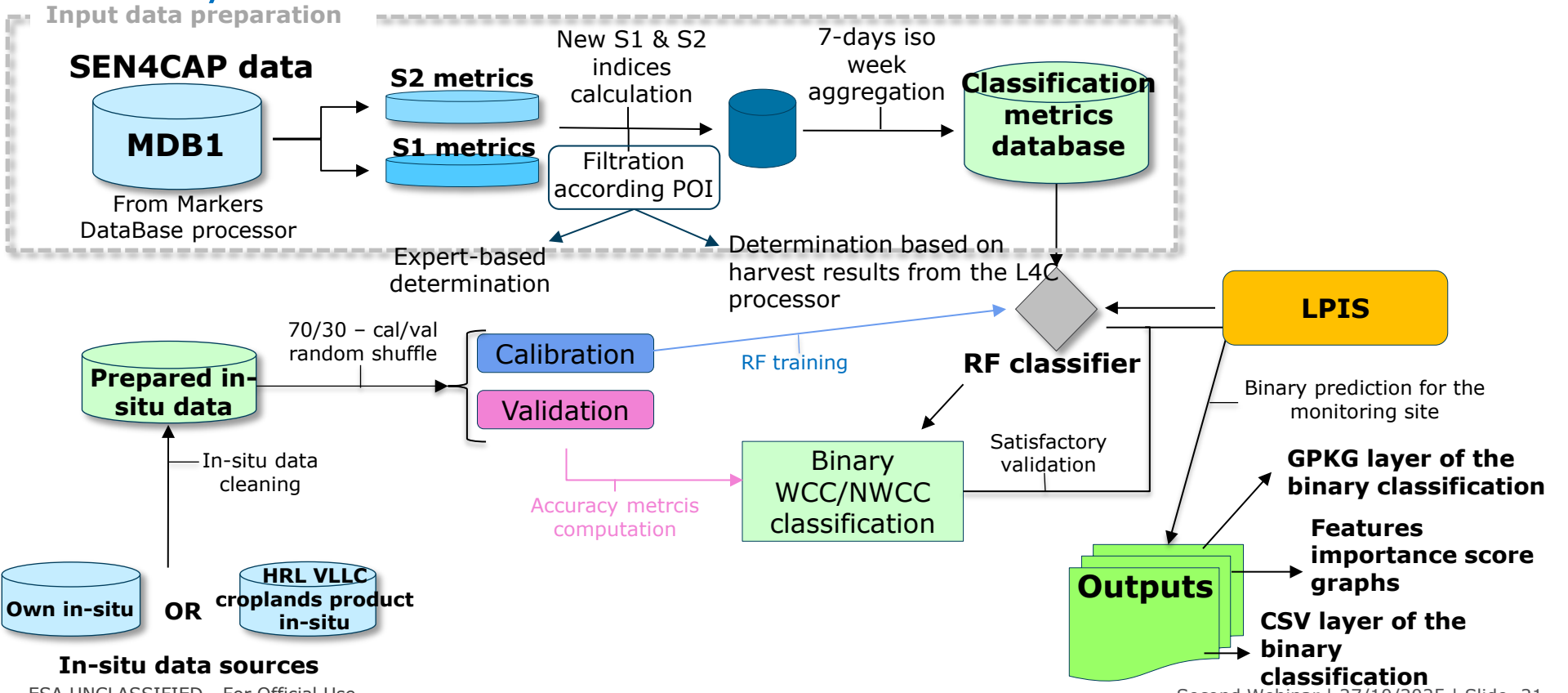


Determined based on user survey responses

WCC : in-situ data and threshold-based approaches

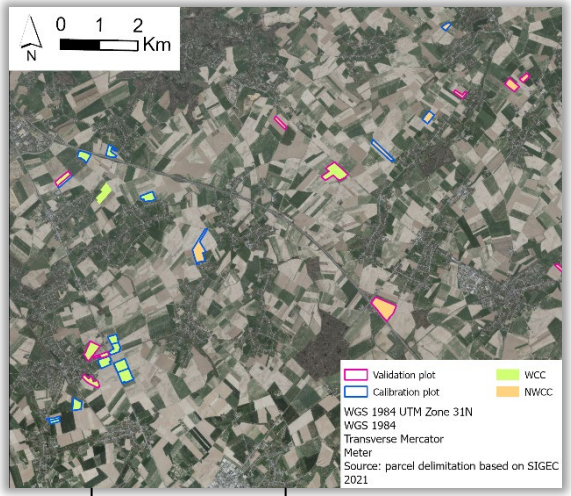


WCC binary detection in-situ data-based algorithm Summary workflow



WCC processor: two scenarios for in-situ data acquisition

Own in-situ data



Input format for importing into the system:

- CSV file
  - Min 2 mandatory attributes

Only Ori_id attribute in the in-situ data file

NewID attribute already present in the in-situ data file

Linking Ori_id with NewID attribute

Original ID for parcels

Internal ID for parcels

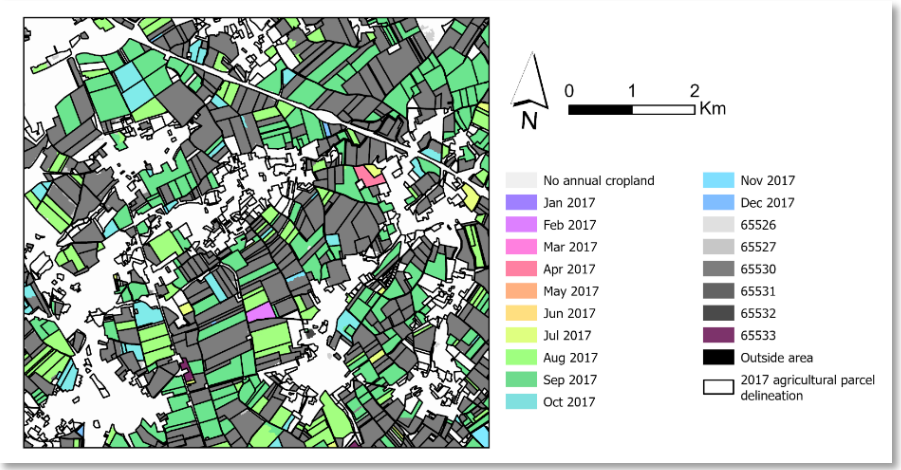
WCC label

NWCC label

Ori_id	NewID	Class
2300	1	1
48	2	2

WCC processor: two scenarios for in-situ data acquisition

In-situ data created from the « Secondary Crops Emergence (SCE) » layer of the HRL VLLC croplands product

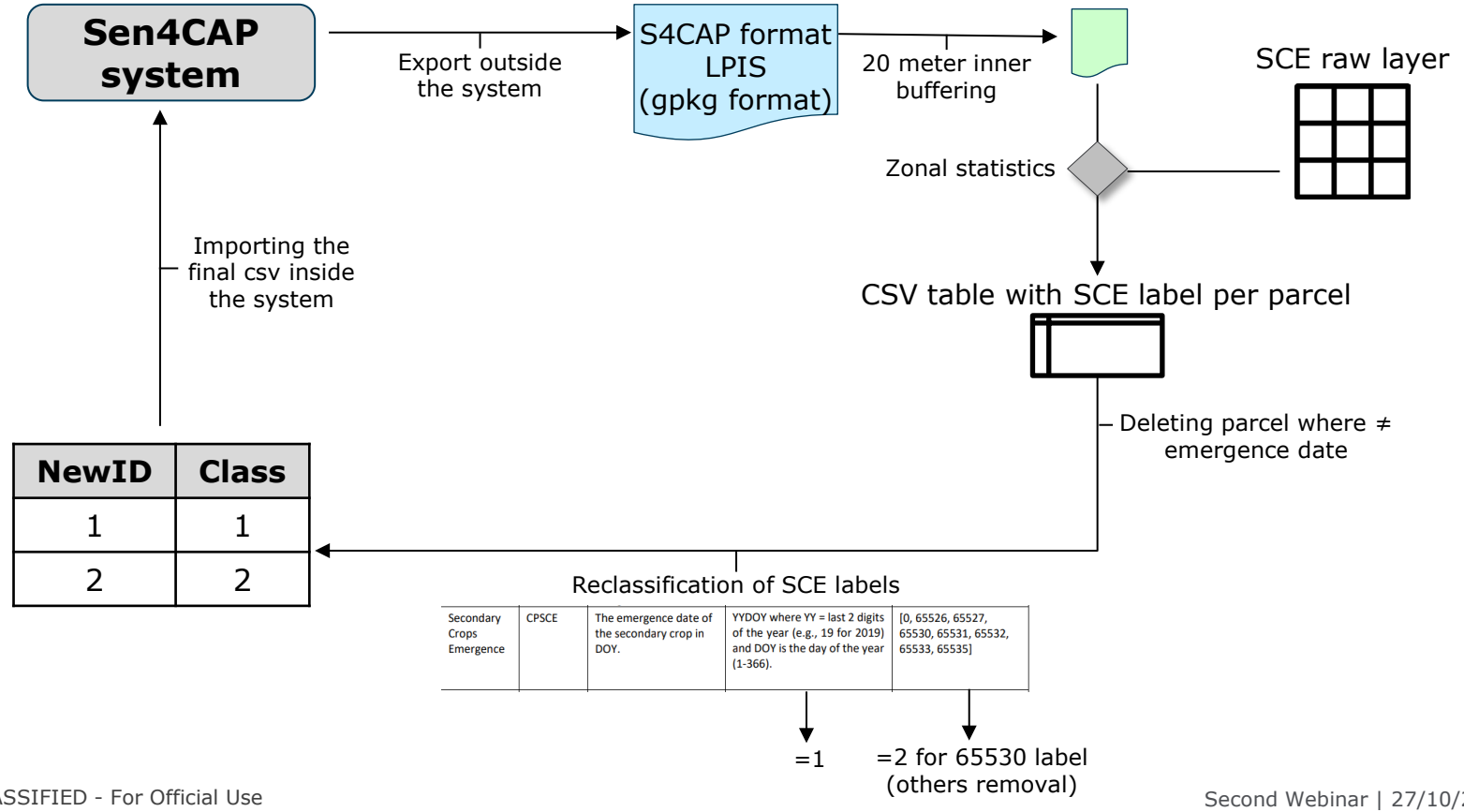


Original properties of SCE data:
- Format : tif file (raster data)
- Extent: Europe
- Available archives: 2017 – 2021 (enough to train the model)

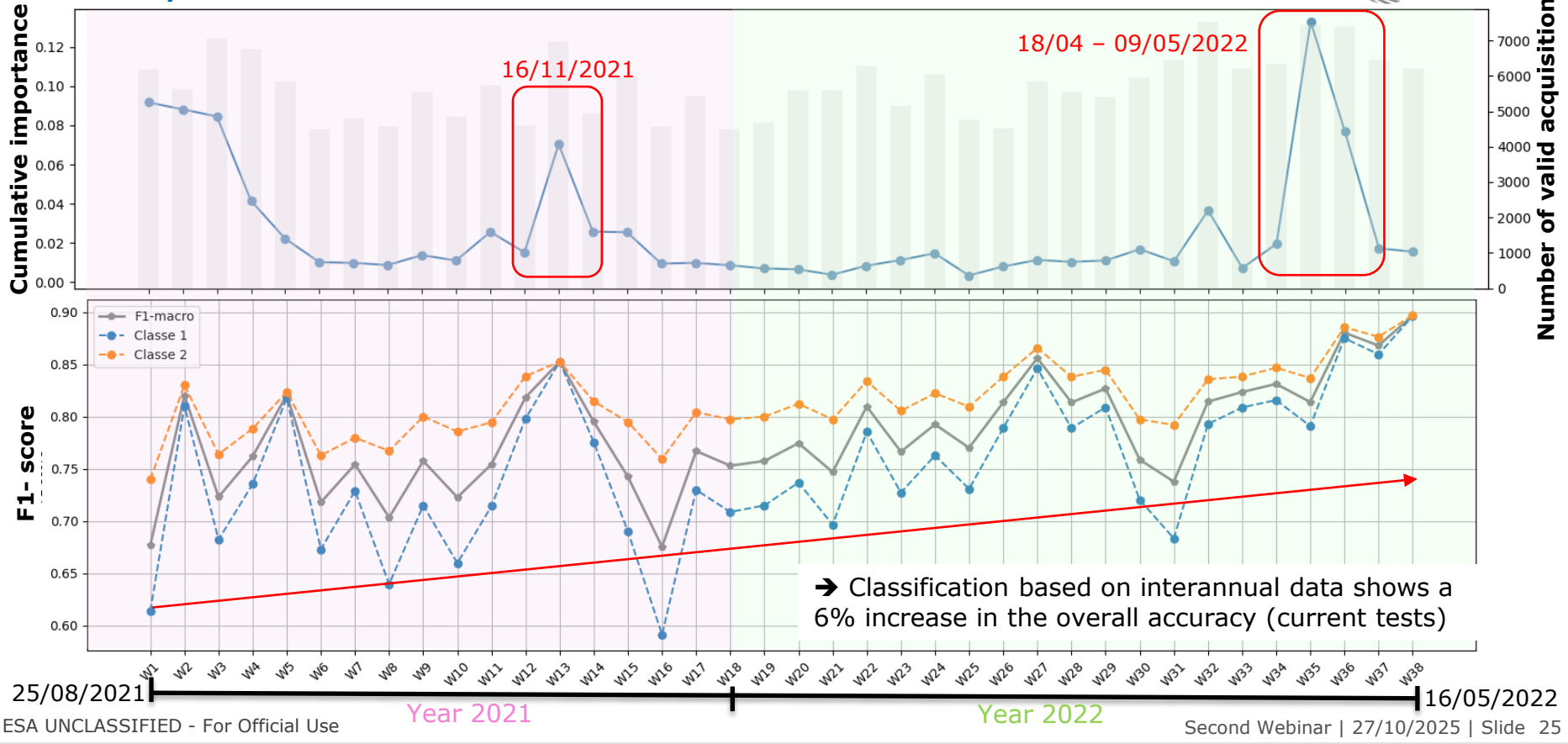
Input format for importing into the system:
- CSV file
  - Min 2 mandatory attributes

Producing method of the CSV:
- External code to be run by the user

WCC processor: two scenarios for in-situ data acquisition

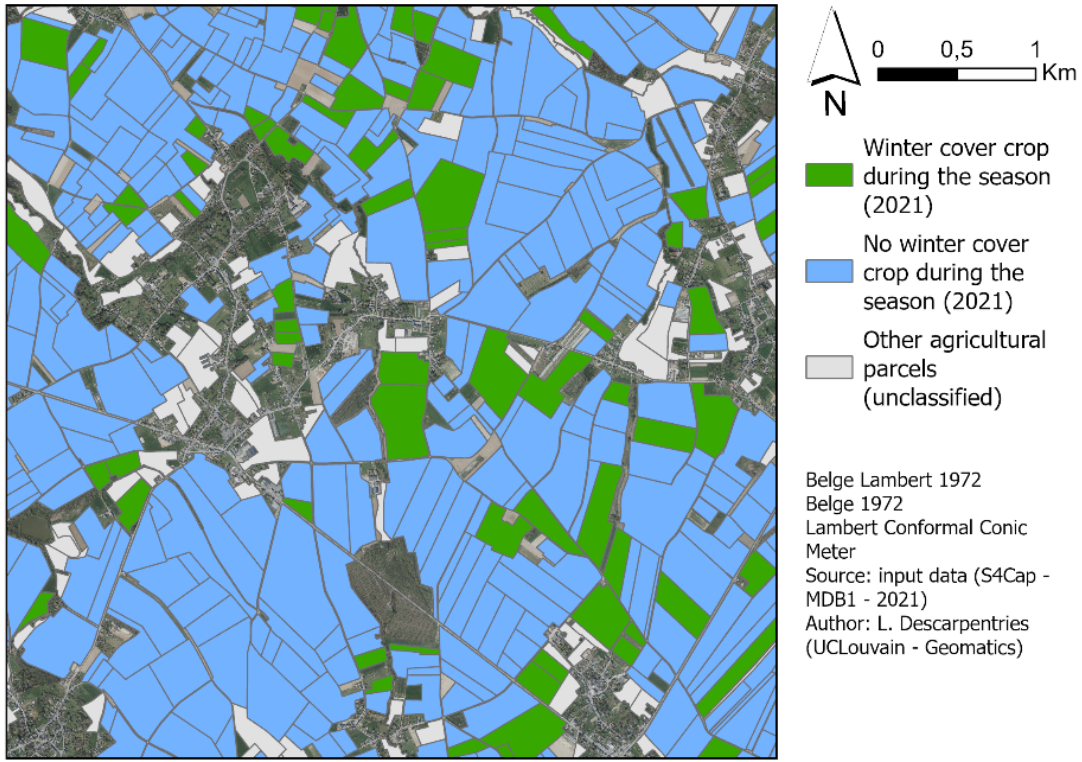


WCC: inter-annual time series importance for classification accuracy



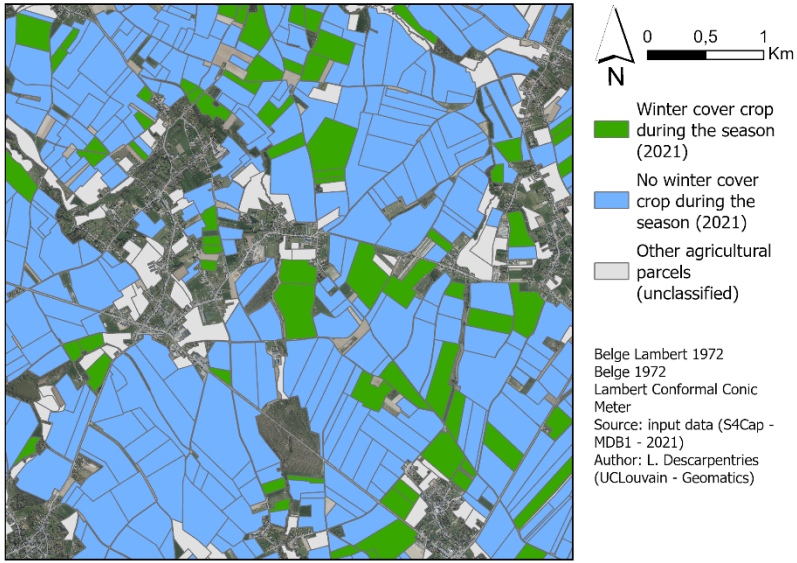
WCC processor expected outputs: CSV layer and geopackage

Binary classification (WCC/NWCC) in gpkg format



WCC processor expected outputs: CSV layer and geopackage

Binary classification (WCC/NWCC) in gpkg format



Attribute table of the gpkg file also in CSV format

NewID	Pred_class	Conf_pred1	Conf_pred2
	1 or 2	[0;1]	[0;1]

Unique identifier for agricultural parcels

Predicted class:
- 1: WCC during the season
- 2 :NWCC during the season

Prediction confidence level for class 1

Prediction confidence level for class 2

WCC processor: Emergence and termination detection – first step for WCC characteristics assessment

A within-season approach for detecting early growth stages in corn and soybean using high temporal and spatial resolution imagery

Feng Gao [a], Martha Anderson [a], Craig Daughtry [a], Arnon Karnieli [b], Dean Hively [c], William Kustas [a]

Emergence detection (WISE)

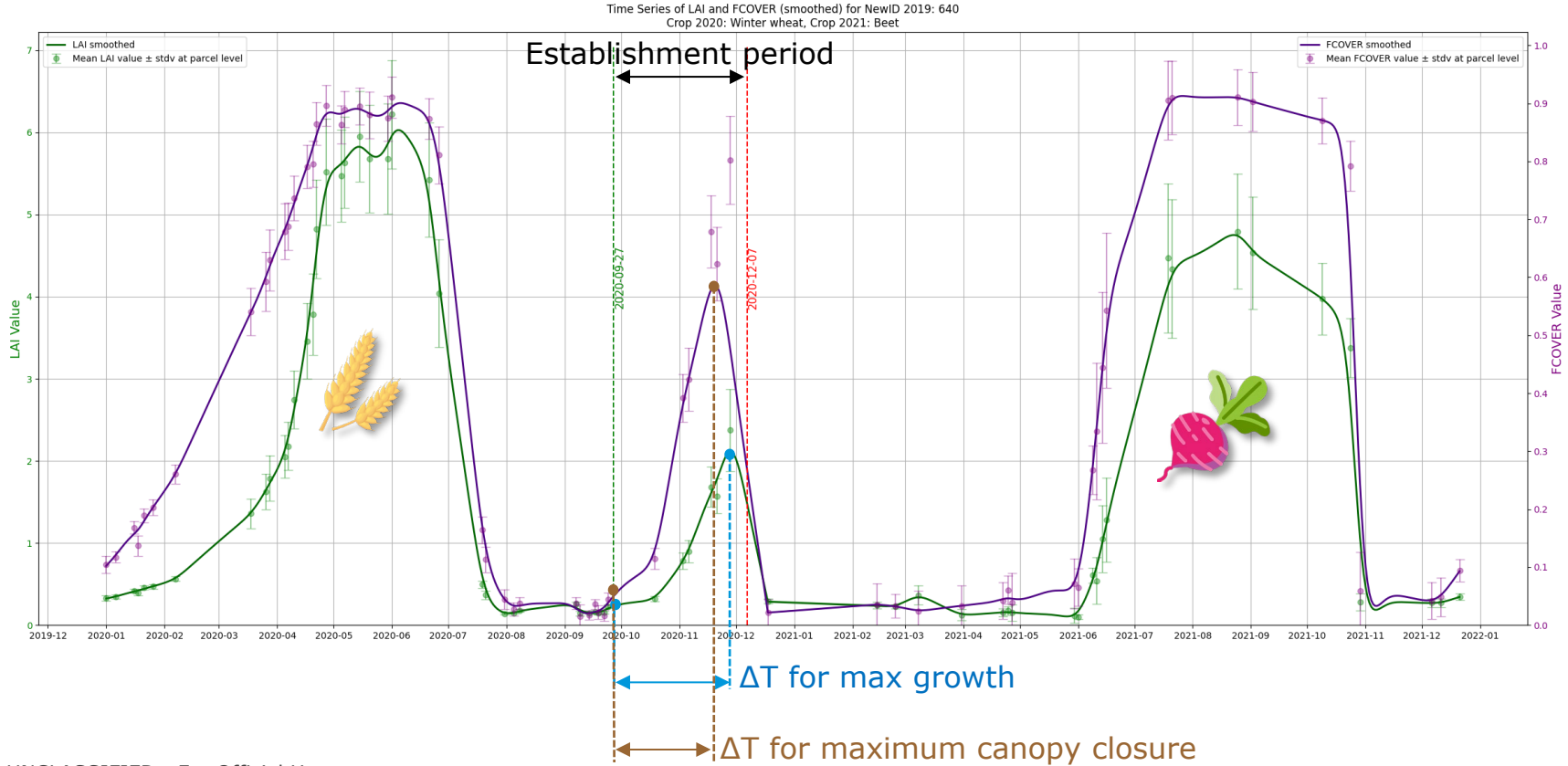
Near real-time detection of winter cover crop termination using harmonized Landsat and Sentinel-2 (HLS) to support ecosystem assessment

Feng Gao [a], Jyoti Jennewein [b], W. Dean Hively [c], Alexander Soroka [d], Alison Thieme [b], Dawn Bradley [e], Jason Keppler [e], Steven Mirsky [b], Uvirkaa Akumaga [a]

Termination detection (WIST)

- Modified WISE and WIST algorithms (adjusted thresholds and date filtering method)
- Algorithms currently calibrated and validated on HRL VLLC SCE layer
  - Need for real in-situ data on emergence and termination to improve robustness

WCC processor: WCC characteristics extraction



WCC processor: WCC characteristics output csv

Output: CSV table

NewID	Em_d	Ter_d	S_length	Uncert_em	Uncert_ter	M_gro_d	M_gro_val	M_gro_t	M_cov_d	M_co_val	M_co_t
int	date	date	days	days	days	date	LAI val [-]	days	date	FCOVER val [-]	days

Emergence date

Termination date

WCC season length

Uncertainty in days linked to emergence and termination detection

Date for max LAI after emergence

Value at max LAI

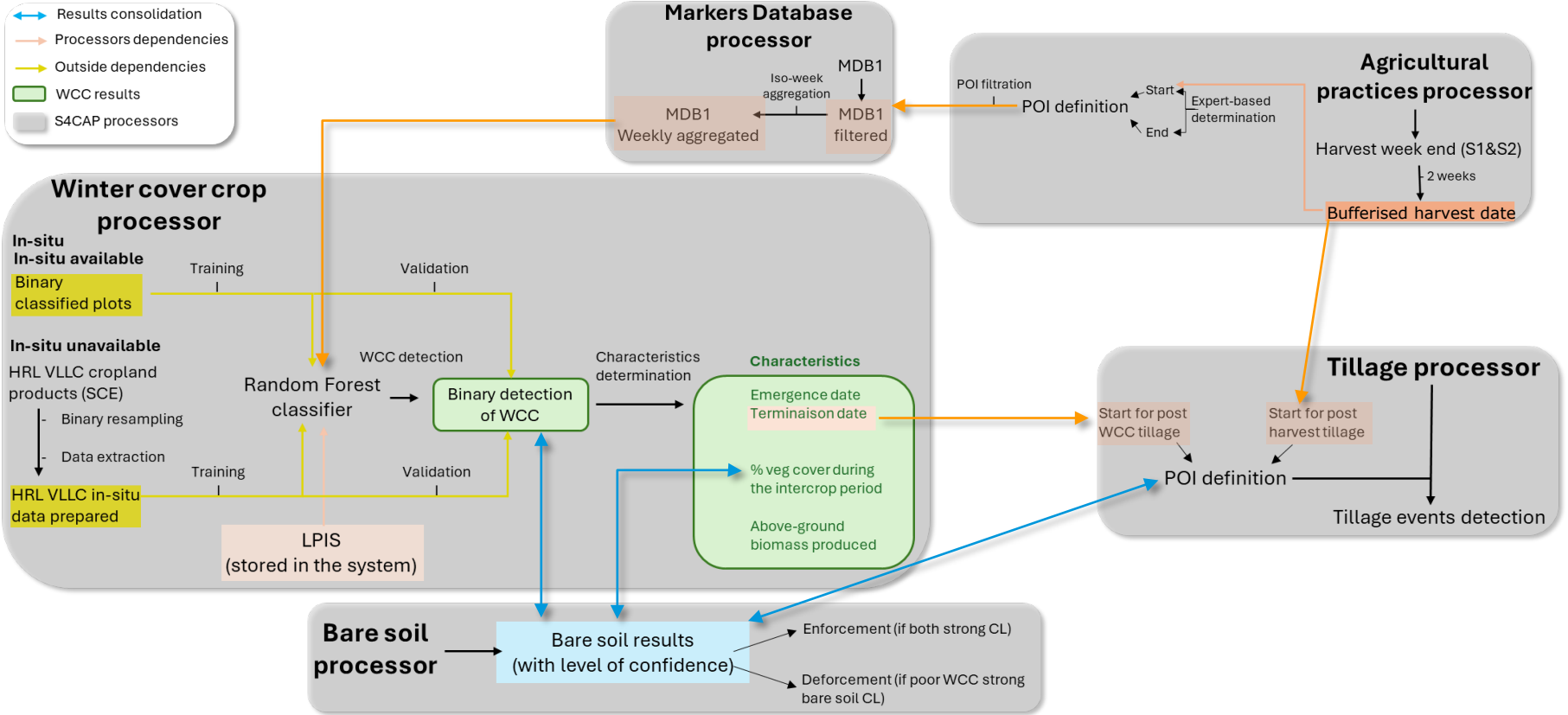
ΔT between emergence and max LAI

Date for max FCOVER after emergence

Value at max FCOVER

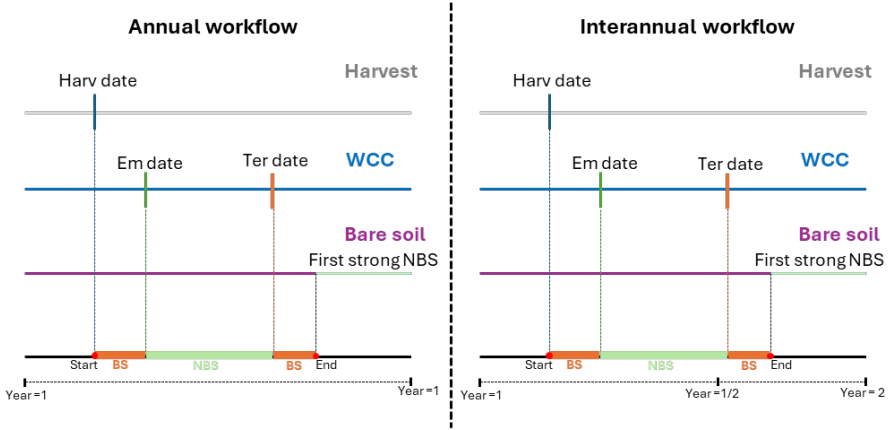
ΔT between emergence and max FCOVER

WCC processor: overall dependencies within the system for WCC characteristics assessment

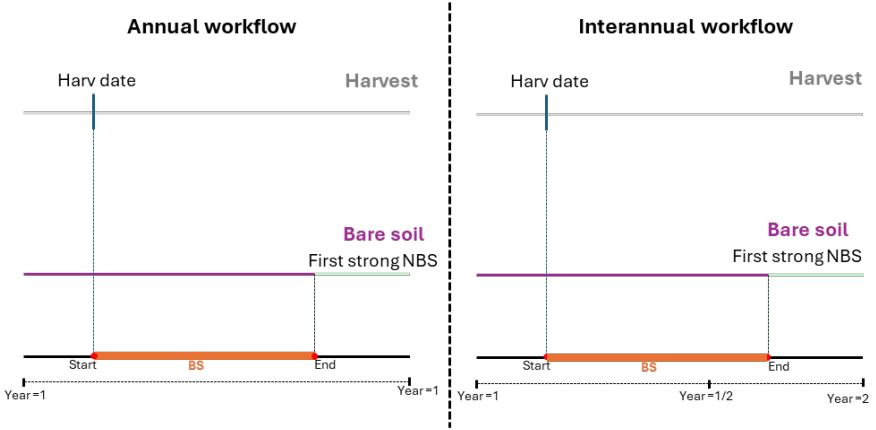


WCC processor: %BS and NBS within the intercropping period

WCC scenario



NWCC scenario



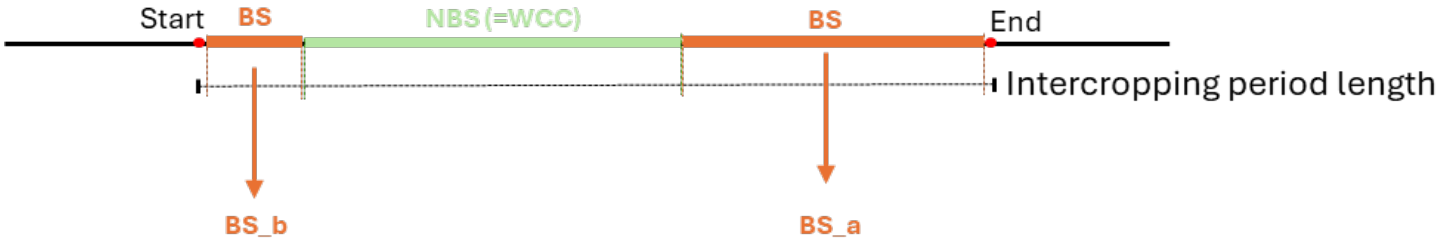
Inputs:

- Start of harvest week from L4C processor
- Emergence and termination date from WCC processor
- BS/NBS detection from bare soil processor

Inputs:

- Start of harvest week from L4C processor
- BS/NBS detection from bare soil processor

WCC processor: intercropping period analysis - expected outputs



Output: CSV table

NewID	Inter_d	BS_d	NBS_d	BS_b	BS_a	BS_per	NBS_per	BS_b_per	BS_a_per
Id of the field	Length of the intercropping period [days]	BS period length within the intercropping [days]	NBS period length within the intercropping [days]	BS period before WCC emergence [days]	BS period after WCC termination [days]	% of BS time over the entire intercrop period	% of NBS time over the entire intercrop period	% of BS before WCC emergence over the entire intercrop period	% of BS after WCC termination over the entire intercrop period

WCC processor next steps

- Further developments and consolidation

- Integration of bare soil data for detection consolidation (in progess)
- Emergence and termination detection consolidation with S1 VH, VV, rVH/VV metrics
- Development of the workflow for above-ground biomass estimation

- In situ data concerns:

- Need for calibration and validation data for the different analyses
    - Presence/absence of WCC
    - Emergence/termination dates of WCC
    - WCC above-ground biomass measurements
    - WCC species

TD processor is in development

Which information can be used to reduce false positives?

- Proxies of residue cover?
- Correct effect of precipitation?

Objective: Generalizable threshold-based algorithm

Need for **validation data**, feel free to reach out!

THANK YOU!