

Plans for spec, core and ocrd_all

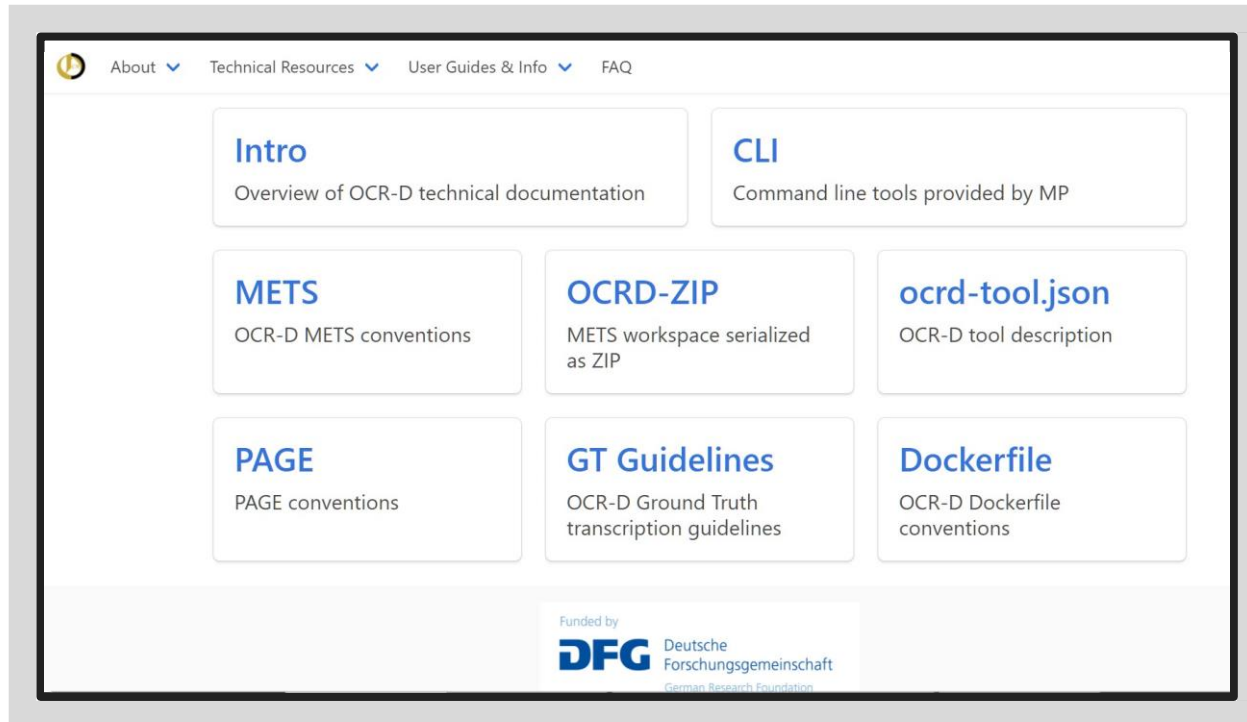
Konstantin Baierer

OCR-D Phase 3 Kick-Off Workshop

2021-07-30

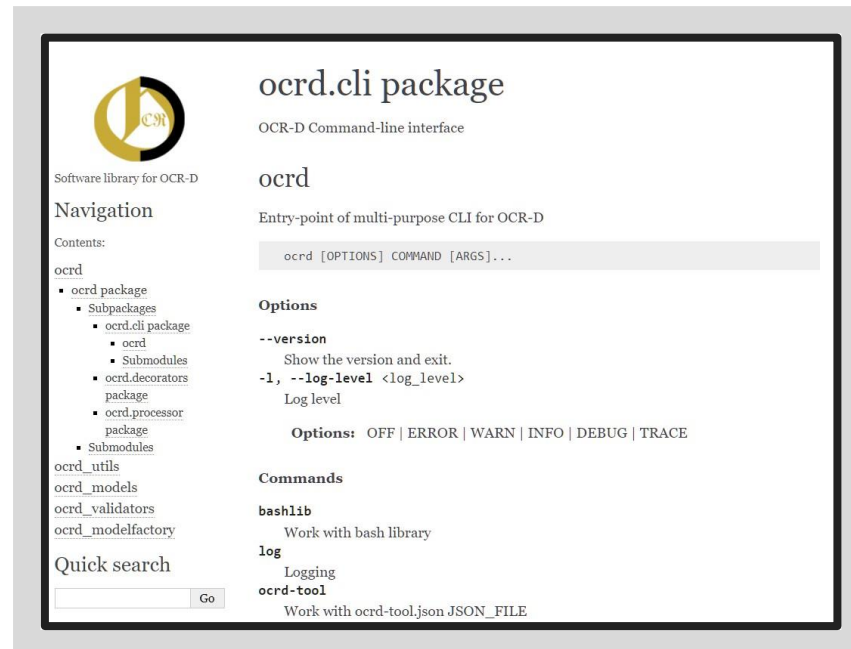
Recap - spec

- Specifications that define the mechanisms, conventions and standards we use in OCR-D



Recap core

- Python reference implementation of the specs
- Command line tools for various tasks
- APIs for PAGE-XML, METS (workspace), basic workflows (ocrd process)

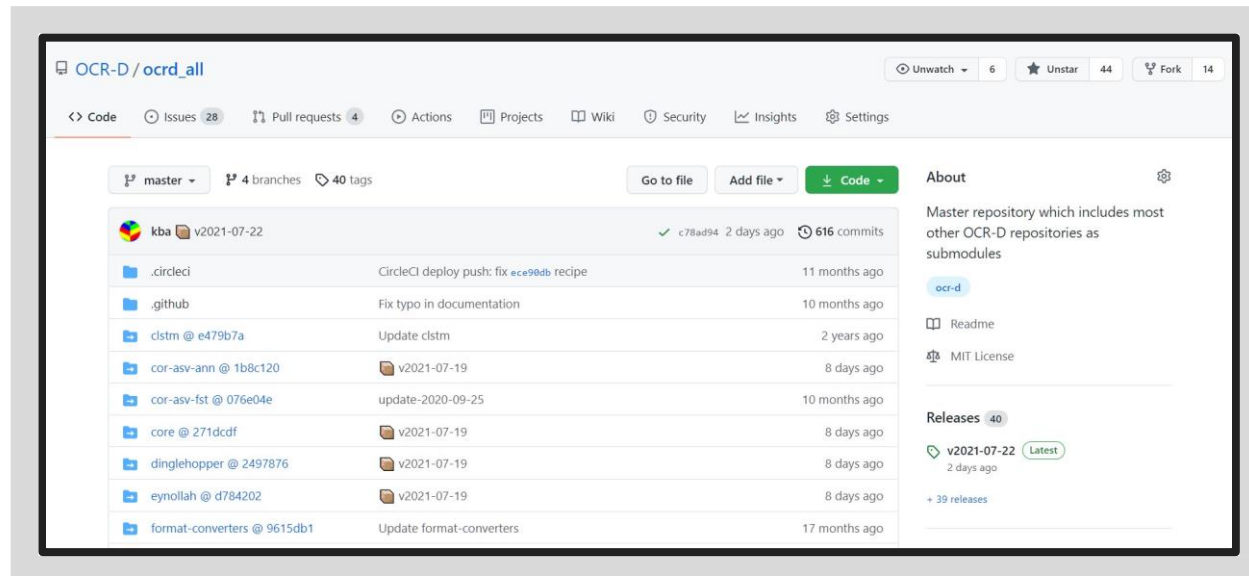


The screenshot shows the documentation for the `ocrd.cli` package. On the left, there is a navigation sidebar with a logo (a yellow circle with 'OCR-D' inside) and the text 'Software library for OCR-D'. The sidebar lists contents including 'ocrd' (with subpackages like 'ocrd.cli package', 'ocrd.decorators package', and 'ocrd.processor package') and 'ocrd_utils', 'ocrd_models', 'ocrd_validators', and 'ocrd_modelfactory'. There is also a 'Quick search' field with a 'Go' button.

The main content area is titled 'ocrd.cli package' and describes it as the 'OCR-D Command-line interface'. It lists the 'ocrd' entry-point as the 'Entry-point of multi-purpose CLI for OCR-D' and shows the command syntax: `ocrd [OPTIONS] COMMAND [ARGS]...`. Below this, it lists 'Options' such as `--version` (to show version and exit) and `-l, --log-level <log_level>` (to set log level). The available log levels are listed as 'Options: OFF | ERROR | WARN | INFO | DEBUG | TRACE'. It also lists 'Commands' including `bashlib` (Work with bash library), `log` (Logging), and `ocrd-tool` (Work with ocrd-tool.json JSON_FILE).

Recap ocrd_all

- Meta-project containing all processors as git submodules
- Building docker containers
- Native installation



Plans for spec

Formalize concepts from OCR- D / core

- Workspace
- Processor
- Handling of `pc:AlternativeImage`

Revise ocrd-tool.json

- Drop `input_file_grp/output_file_grp`
- Show what kind of image transformations the processor supports instead
- Allow defining [resources](#)

**Quality estimation processors /
evaluators**

Common format for (dynamic) workflows

- Going beyond ocrd process
- Descriptive syntax, e.g. YAML

Metadata and exchange format for models

- Engine-independent metadata for OCR models
- Single-file format for Calamari checkpoints

Web API

See next talk :-)

Plans for core

Page-wise processing

- `process` -> `process_page`
- Introduce `setup` method

Improved logging and error handling

- measuring throughput
- actionable logging for intelligent workflows

Managing computing resources

Efficiently use GPU, CPU cores and RAM

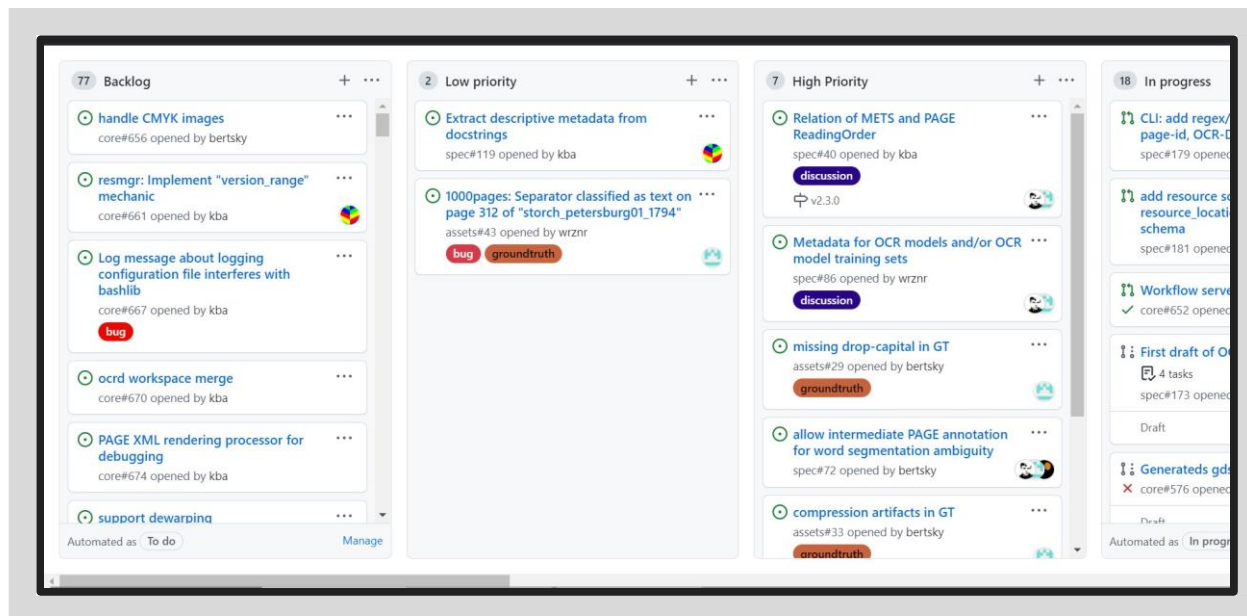
Networked components for scalable workflows

[Workflow/Processing Server \(White Paper\)](#)

Test data and infrastructure for evaluating workflows

PAGE API: Scaling and dewarping

Reduce our issue/PR backlog



Plans for ocrd_all

Transition to slim containers

- https://github.com/OCR-D/ocrd_all/issues/69
- <https://github.com/qurator-spk/ocrd-galley>

CUDA support, Docker and native

Better dependency management

Use our own package infrastructure
with prebuilt tensorflow, tesseract, OLENA

Better CI/CD

- Additional artifacts beyond DockerHub
- Test with meaningful (workflow) test data

Thank you!