

Towards automating FAIR dataset record deployment as a continuous integration process

Use of GitHub Actions (GHA) on RDA maDMP metadata

CRIS2026 — Paper #23

M.-A. Sicilia^{id}, A. Ballesteros-Rodríguez^{id}, E. García-Barriocanal^{id},
M. Mora-Cantallops^{id}, A. Nogales^{id}

University of Alcalá | IRyCIS (Instituto Ramón y Cajal de Investigación Sanitaria)

May 20–22, 2026

Motivation

Approach

Prototype

Conclusions

FAIR principles (Wilkinson et al. 2016)

require:

- ▶ Machine-actionable, discoverable outputs
- ▶ Rich, standardised metadata
- ▶ Clear licensing & provenance
- ▶ Consistent quality & versioning

The gap: principles are widely adopted, but
consistent implementation is manual and
error-prone

Key insight

Software engineering solved the same problem with
CI/CD — why not apply it to metadata?

Underwood (2023): “continuous metadata” embedded in CI/CD

Suetake et al. (2023): automated validation improves reusability

224,580 Zenodo records, 493 research institutions — 92.0% fail at least one rule

Rule	% failing
Version string present	82.6%
Subject keywords present	45.3%
License present	35.4%
Creator ORCID present	33.5%
Description ≥ 100 ch	13.4%
Access rights stated	0.0%

Modal failure: **2–3 rules simultaneously**

No correlation with institution size

Trend 2015–2026: two very different stories

Platform made mandatory $\Rightarrow$ compliance followed:

- ▶ ORCID: 98.0% $\rightarrow$ 13.7% non-compliant
- ▶ License: 79.4% $\rightarrow$ 8.0% non-compliant

Requires deliberate depositor input $\Rightarrow$ no improvement:

- ▶ Version: 98.9% $\rightarrow$ 80.6% (barely moved in 11 years)
- ▶ Keywords: 7.4% $\rightarrow$ 62.7% — regressed

Takeaway

Without a policy gate, fields that require human judgement do not improve — and may get worse.

Metadata for datasets originates from:

- ▶ Project DMPs (Data Management Plans)
- ▶ Operational data from REDCap & other systems

Internal metadata catalog: **OpenMetadata** (institution-wide)

Target: publish validated records to **external FAIR repositories**

Key design choice

GitHub repositories = **source of truth** for versioned, publication-ready metadata records

Individual components (GHA, maDMP schema, Zenodo API) already exist.

Proposed approach: the **organisational automation pattern**

1. Policy enforcement via .github repository

Organisation-wide metadata quality rules applied to every dataset

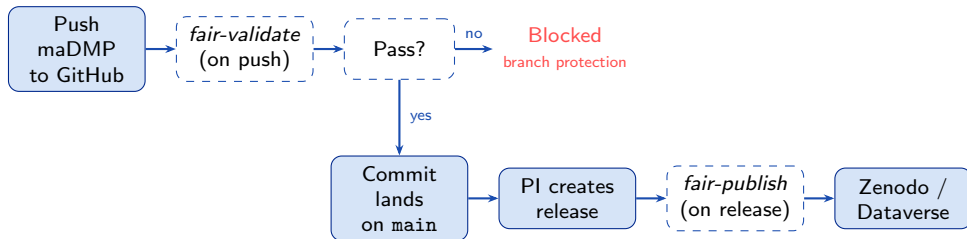
2. Semantic versioning for dataset records

Each version bump triggers the full pipeline

3. Chained, auditable workflow

maDMP validation → FAIR metric checks → multi-repo deployment — all triggered by a single release event

Distinguishes the approach from ad-hoc upload scripts or single-repository integrations



Benefits:

- ▶ **Consistency** — same metadata to all targets
- ▶ **Auditability** — GHA execution history
- ▶ **Efficiency** — one release, many endpoints
- ▶ **Flexibility** — add/remove repos in config

`.github` (governance)

Org-level special repo.

`.github/workflows/`

- ▶ `fair-validate.yml`
- ▶ `fair-publish.yml`

`.github/policies/`

- ▶ `fair-policy.yml`

*Owned by institution.
Policies propagate to
all project repos.*

`fair-publish` (engine)

Institution-agnostic
validation & publishing
package. GHA steps call:

```
fair-publish validate  
fair-publish publish
```

Installed at runtime via
`pip install git+...`

*Institution-specific rules
live in `.github`, not here.*

`demo-dataset` (project)

One PI's dataset repo.

- ▶ `dmp.json` — the maDMP
- ▶ `zenodo_state.json`
- ▶ `.github/workflows/`
 - `validate.yml`
 - `publish.yml`

*PI pushes `dmp.json`.
Everything else is
inherited from org.*

Code available at <https://github.com/in-silico-institute>

Six built-in FAIR rules applied to every dataset:

- ▶ Creator ORCID present *F1, A1*
- ▶ License present & recognized *R1.1*
- ▶ At least one keyword *F2*
- ▶ Description $\geq$ N chars *F2, R1*
- ▶ Access rights stated *A1*
- ▶ Dataset identifier present *F1*

Policy file lives in `.github/policies/`;
automatically inherited by all project repos.

Listing 1: `fair-policy.yml` (in-silico-institute/.github)

```
version: "1.0"

rules:
  - name: Creator ORCID present
    severity: error
  - name: License present and recognized
    severity: error
  - name: At least one keyword
    severity: error
  - name: Description length
    severity: warning
    config:
      min_chars: 150
  - name: Access rights stated
    severity: error
  - name: Dataset identifier present
    severity: error

targets:
  - zenodo
```

Goal: enforce that every dataset carries at least one keyword drawn from a recognised biomedical ontology (SNOMED CT, LOINC, or HPO).

Two files to add — nothing else changes:

1. Declare the rule in `fair-policy.yml`
2. Drop a Python file into `custom_rules/`

Listing 2: `fair-policy.yml` — add one entry

```
rules:
  # ... existing rules ...
  - name: Biomedical keyword
    severity: error
    config:
      ontologies: [SNOMEDCT, LOINC, HPO]
```

DMP keywords annotated as "SNOMEDCT:38341003", "LOINC:55284-4", "HPO:0000822", ...

Listing 3: `custom_rules/biomedical_keyword.py`

```
PREFIXES = {"SNOMEDCT", "LOINC", "HPO"}

def rule_biomedical_keyword(
    dataset: dict, config: dict) -> bool:
    allowed = set(config.get("ontologies", []))
    return any(
        kw.split(":")[0].upper() in allowed
        for kw in dataset.get("keyword", [])
        if isinstance(kw, str) and ":" in kw
    )
```

Key point

The core package is not modified. Institutions ship their own rule sets; the GHA workflow picks them up automatically.

- ▶ GHA is a viable CI/CD backbone for FAIR metadata publishing
- ▶ Policy-guided, organisation-wide quality enforcement
- ▶ Low technical entry barrier — replicable without custom software
- ▶ Directly connects to CRIS infrastructure as upstream layer

Current limitations

- ▶ Public metadata only
- ▶ No automated error notifications
- ▶ OpenMetadata integration not yet automated
- ▶ Policy evolution not yet automated
- ▶ Quality gate requires branch protection
rules configured at org level

Takeaway

The contribution is an **organisational automation pattern**: treat metadata publication as a software release process.

Thank you!

Questions welcome

`msicilia@uah.es`

`https://github.com/in-silico-institute`

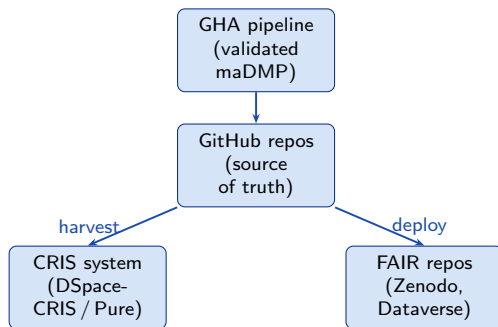
Paper #23 — CRIS2026, University of the Azores, Ponta Delgada, May 20–22, 2026

GitHub repositories with validated maDMP metadata act as a **harvestable data layer** for institutional CRIS systems:

- ▶ DSpace-CRIS, Pure, Converis . . .
- ▶ Ingest via APIs or standard protocols
- ▶ No CRIS-level re-validation needed (already enforced upstream)

Positioning

A **complementary upstream layer** to CRIS—not a replacement. Feeds validated, version-controlled dataset records into the broader research information ecosystem.



Trigger configuration:

```
on:
  push:
    branches: [main]
    paths: ["dmp.json"]
  pull_request:
    paths: ["dmp.json"]

jobs:
  validate:
    uses: in-silico-institute/.github/
      .github/workflows/
      fair-validate.yml@main
    with:
      dmp_path: dmp.json
    # policy fetched automatically
    # from org .github repo
```

Actual run output:

```
DMP: Longitudinal cohort study of metabolic biomarkers
Datasets: 1 Errors: 0 Warnings: 0
```

```
Dataset: Metabolic biomarker dataset
[F1, A1] Creator ORCID present OK
[R1.1] License present & recognized OK
[F2] At least one keyword OK
[F2, R1] Description length OK
[A1] Access rights stated OK
[F1] Dataset identifier present OK
```

```
PASSED
```

All 6 rules pass. Exit code 0.

A failing rule exits 1 — blocks the commit from reaching main and therefore prevents any release.

On release, Zenodo publication and PI e-mail notification both happen automatically.

What the pattern assumes

1. **External authoring front-end**

maDMP is produced by a researcher-friendly DMP tool; this pipeline is the back-end only

2. **Institutional repository ownership**

Repos are owned by the organisation, not individual PIs — prevents state inconsistencies from forking or deletion

3. **Branch protection configured at org level**

Validation only acts as a hard gate if GitHub branch protection rules *require* it to pass before a release is permitted

Assumption 1 is deferred to future work; assumptions 2 and 3 are governance decisions, not technical ones

Workflow 1 — `fair-validate.yml`

Trigger: *push / pull request (paths: dmp.json)*

Two sequential **jobs** visible in GHA graph:

1. **Schema check** — structural validity via `madmpy` (RDA maDMP Common Standard)
2. **Policy check** — institutional FAIR rules; only runs if schema check passes

Governance requirement

Branch protection must require this check to pass — otherwise a PI can create a release without validation

Workflow 2 — `fair-publish.yml`

Trigger: *GitHub release published*

- ▶ New dataset → create new Zenodo draft
- ▶ Known dataset → new version (via `zenodo_state.json`)
- ▶ Commits updated state file back to repo
- ▶ Notifies PI by email on success

Why split?

Validation gives fast feedback on every push. Publishing only happens at a deliberate version event. The gate cannot be bypassed.

Trigger: GitHub release event

```
on:
  release:
    types: [published]
jobs:
  publish:
    uses: in-silico-institute/.github/
      .github/workflows/
      fair-publish.yml@main
    with:
      dmp_path: dmp.json
      sandbox: true
      secrets: inherit
```

ZENODO_TOKEN stored as org-level secret —
PI never handles credentials directly.

Actual run output:

```
DMP validated!
[draft created] Metabolic biomarker dataset
id=1234567 doi=10.5072/zenodo.1234567
policy=1.0
Published 1 dataset(s).
[new] metabolic-biomarker-cohort-wave1-5
-> record=1234567 policy=1.0
[main] chore: update zenodo_state.json after v1.0.0
PI notified by email.
```

Zenodo sandbox draft created. State file committed back to
repo. PI receives institutional email with record details.