Semantic UMD product ID cards

- Problems with the existing way of doing things
- Organising information semantically
- Choosing a standard schema
- Example implementation
- Questions

What problem are we solving? - cognitive load

- Currently, information about the state of the product is kept in various places, depending on the product
  - The repo (spec, changelog, git history, etc)
  - The product web page
- No common metadata schema for products – you are liable to discover different things about products from themselves
- What is the **canonical** place to find metadata? A wiki.
  - This in itself is a problem – too easy for human error to creep in
  - Impossible to maintain provenance
  - Rollback is even more difficult.

Other problems

- No way to express machine-readable dependencies between products
  - Makes testing, integration and delivery difficult
- **No links between different entities**
  - Which **people** are responsible for this software?
  - What **roles** do those people have?
  - What **platforms** is this software for?

How do we address this problem?

- Some discussion started a while back - https://community.egi.eu/t/package-json-for-umd-products/203
- Separate the model from the view
- All UMD products have an implicit data model – the table on the wiki. However:
  - the table conflates the model with the view
  - Impossible to lint or check
  - too easy to write inconsistent information
  - Impossible to “back-propagate” to the actual product
- The product needs to keep the model with itself, and propagate it, so that *any* view can be constructed.

We need two things:

A schema

- We are describing a bunch of *things* which need to maintain a consistent *meaning*

Linked Data

- We need to have semantic *relationships* between *things*

One more thing:


KISS

There is a solution!
Schema.org + JSON-LD

- Schema.org – describes **things** on the internet
- JSON-LD – lightweight alternative to RDF for **linked data**
- But will it work for UMD?

Schema.org Terms vs UMD terms

- The Schema.org vocabulary has been extended for research software by Codemeta
- Adopting the Codemeta vocabulary, the only term missing from what UMD needed was IPV6 Readiness - https://community.egi.eu/t/comparing-umd-product-cards-with-schema-org/207
- So we added that https://github.com/codemeta/codemeta/pull/194

Implementation

- Well-defined, standard schema (Codemeta) ✅
- Simple language (JSON) ✅
- Ability to link terms (JSON-LD)✅
- Machine-readable? ✅
- Solves our problem?

e.g. BDII

- Initial proposal : put it in the actual repo of the actual product:
  - https://github.com/EGI-Foundation/bdii/pull/9
- But some products do not have have a single canonical repo.
  - Put all the products in a massive umd.json ?
- Maybe need some tooling to merge various products... it's JSON after all

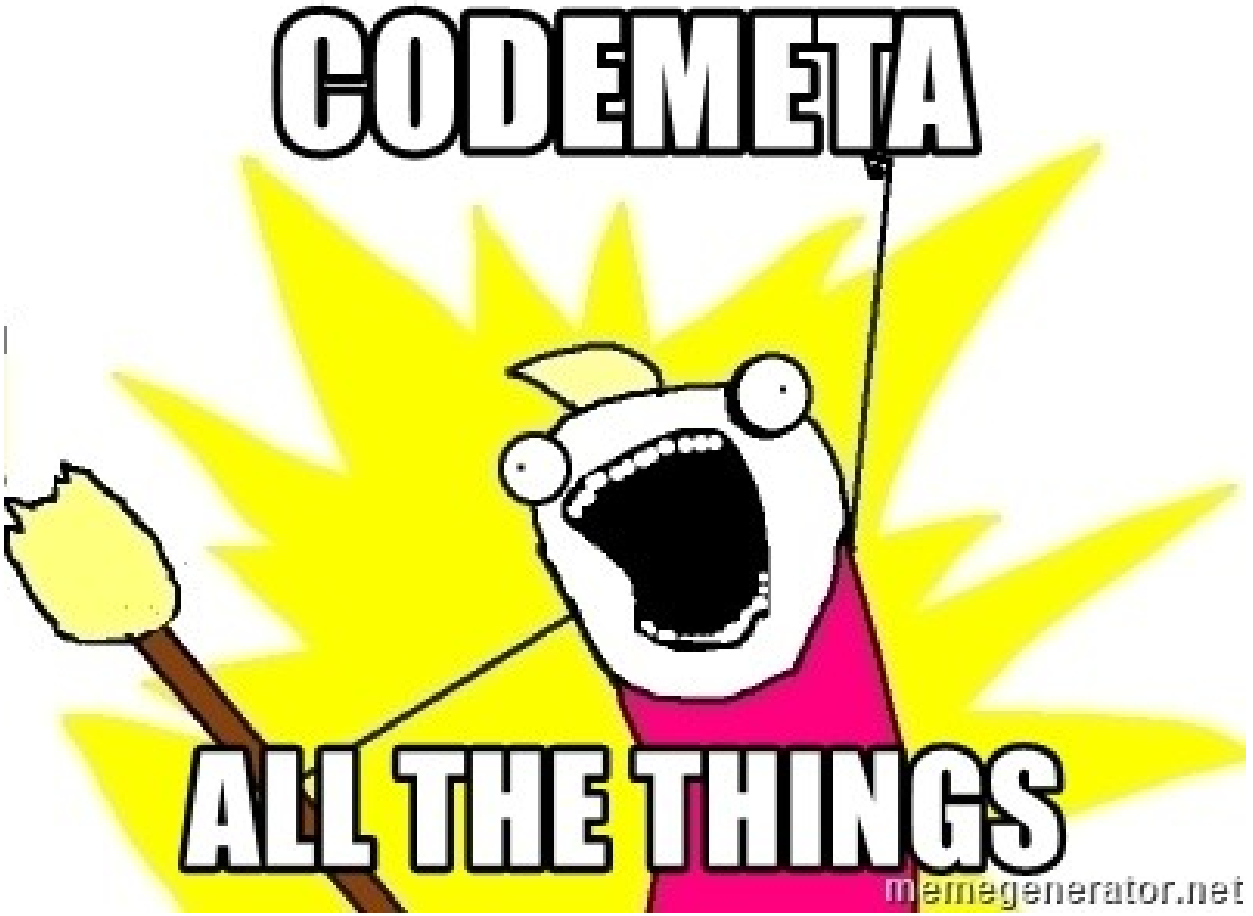
Information is easily parseable


```
{
  "@context": "https://doi.org/10.5063/schema/codemeta-2.0",
  "@type": "Code",
  "name": "Berkeley Database Information Index (BDII)",
  "description": "The BDII implementation of the
information",
  "provider": {
    "@type": "Organization",
    "name": "",
    "url": ""
  },
```

Who the maintainer?
Who is the security contact?

```
"maintainer": [
    {
      "@type": "Person",
      "@id": "https://orcid.org/0000-
0002-5686-3193",
      "name": " Baptiste",
      "familyName": "Grenier",
      "affiliation": {
        "@type": "Organization",
        "name": "EGI Foundation",
        "url": "http://www.egi.eu"
      }
    },
...
],
```

```
{
 "@type": "Role",
    "roleName":
  "Security and
Vulnerability",
    "url": "
}
```


CODEMETA
ALL THE THINGS
memegenerator.net

Easy, Tiger

This has to work with the AppDB

- AppDB interoperates with OpenAIRE, has an XML data exchange format
- UMD products should not be “special” things in the AppDB, but treated just the same as all the other applications.
- However, having an ID card *in the repo* with a *standard schema* makes QA and continuous delivery to AppDB far more feasible.