Language engineering for the masses: business rules for the Digital COVID-19 Certificate

LangDev Meetup, September 26th 2022

Meinte Boersma

DSL Consultancy

The EU DCC

Quick facts

What & Why

A system
provided through the
EU eHealth Network (eHN)
to issue and verify digital proofs of
vaccination, test, or recovery,
to facilitate freedom of movement during the COVID-19 pandemic
in a GDPR-compliant way.

Introduced July 1st 2021
Countries participating ~80 (EU MS + EFTA + “Third Countries”)
Number issued ~6 billion
Bigger than (WHO/ICAO), or compatible with (DIVOC) similar standards

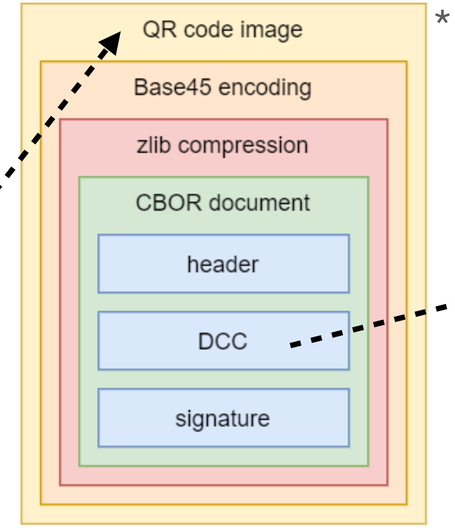
DSL Consultancy

The EU DCC

What's in it

*

QR code



*

structure


```
{
  "ver": "1.3.0",
  "nam": {
    "fn": "Achternaam",
    "fnt": "ACHTERNAAM",
    "gn": "Voornaam",
    "gnt": "VOORNAAM"
  },
  "dob": "1963",
  "v": [
    {
      "tg": "840539006",
      "vp": "1119305005",
      "mp": "CVnCoV",
      "ma": "ORG-100032020",
      "dn": 1,
      "sd": 6,
      "dt": "2021-02-18",
      "co": "GR",
      "is": "Ministry of Health Welfare and Sport",
      "ci": "urn:uvci:01:NL:74827831729545bba1c279f592f2488a"
    }
  ]
}
```


DCC JSON payload

*) Images reproduced from https://www.bartwolff.com/Blog/2021/08/08/decoding-the-eu-digital-covid-certificate-qr-code

DSL Consultancy

The EU DCC

Decoding

Decode a DCC with e.g. https://floysh.github.io/DCC-green-pass-decoder/

More info in this blog: https://www.bartwolff.com/Blog/2021/08/08/decoding-the-eu-digital-covid-certificate-qr-code

The EU DCC

What's to verify

Technically valid Signature verifies through DSC + JSON up-to-spec

Fit-for-entry Is the DCC acceptable for its holder to enter a Country of Arrival regarding its entry regulations?

...or *validation rules*, or *conditions*, or *constraints*...

Examples of business logic as *business rules*:

- The result of a test certificate must be negative.
- A first vaccination with the Janssen vaccine must be at least 28 days old.
- A second vaccination with Pfizer must be at most 270 days old
  ...but minors are exempted!

DSL Consultancy

The EU DCC

How to determine fit-for-entry

Sovereignty implies:
Every participating country can have their own entry regulations ⇔ business rules

Problem: Determine fit-for-entry *upfront*

Did someone say "DSL"?!

Solution: Publish business rules prescribed in an *exchangeable, executable format* on EU DCC Gateway

Design decision: Must be a JSON format

DSL Consultancy

The EU DCC

Software systems involved

- EU provides a central **EU DCC Gateway** server to publish DSCs to verify signatures
- Every participating country is responsible for building their own:

  1) Verifiers (apps) - open-source reference implementations are available
  2) National Backend in-between verifiers apps and Gateway
  3) Issuance infrastructure

- Development environments and infrastructure differs wildly across all participating countries

DSL Consultancy

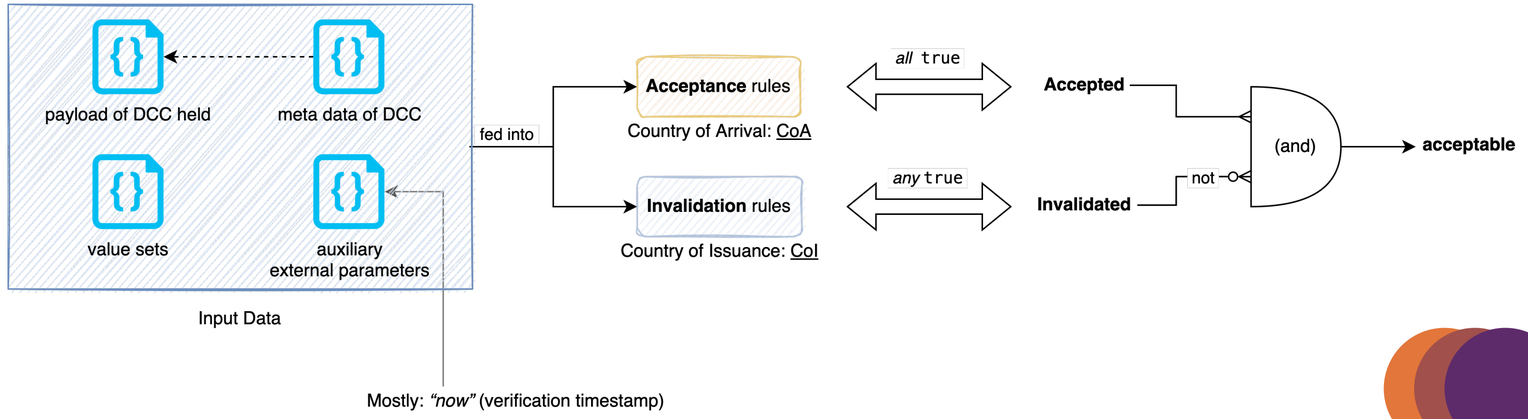
The EU DCC

Validation framework

See: https://ec.europa.eu/health/sites/default/files/ehealth/docs/eu-dcc_validation-rules_en.pdf

Determines how each participating country:

- Should *publish* their business rules (in which format)
- Should *run* business rules when verifying a DCC



DSL Consultancy

Prescribing business rules

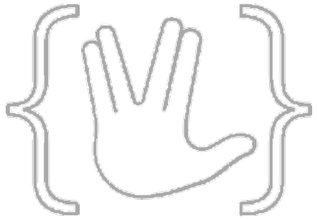
Why a JSON format?

1) Logic as JSON is “just data” ⇔ e.g. compliant with Apple's bytecode policy

2) JSON is well-supported across many platforms

3) No need to write a parser for a textual DSL (for many platforms)

4) E.g. JsonLogic already somewhat known, and allegedly “human-readable”, JSON format for expressing business logic



DSL Consultancy

Prescribing business rules

Why not use JsonLogic?

1) Not small: lots of operations, some with multiple variants (for convenience)

2) Behavior of implementations differ (⇔ no specification)

3) Custom operations are needed for EU DCC;
   e.g.: working with (partial) dates (YYYY, YYYY-MM),
   and UCIs

The solution: **Cert**ificate**Logic**

DSL Consultancy

Prescribing business rules

What is *CertLogic?*

A DSL?
Probably not...

CertLogic is:

- A small (minimal) subset of JsonLogic (on which it's compatible), with a couple of *domain-specific* operations added
- Defined by: a specification of syntax + semantics, backed by a test suite
- On GitHub: https://github.com/ehn-dcc-development/eu-dcc-business-rules/tree/main/certlogic

DSL Consultancy

CertLogic

What is *CertLogic?*

A CertLogic expression *evaluates* (or: “is interpreted”) against given data - typically, the DCC's payload + external parameters + value sets.

As a function:

Haskell people can read this “x” as “→”

`evaluate`: 《*expr*》 x 《*data*》 → 《*result*》 | *Error*

《*expr*》 and 《*data*》 are in JSON format

《*result*》 is usually JSON, but can contain `Date` objects

An error is thrown if the expression is invalid,
or a type incompatibility is encountered.

DSL Consultancy

CertLogic

The “grammar”

Valid CertLogic expressions are:

- a simple literal: a 《*boolean*》, an 《*integer*》, or a "《*string*》"
- an *operation* of the form

  `{ "《operation》": [ 《operand₁》, 《operand₂》, ... ] }`

- an array of CertLogic expressions:

  `[ 《expr₁》, 《expr₂》, ... ]`

DSL Consultancy

CertLogic

Operations (1/3)

- data access: `{ "var": "`《*path*》`" }`
  Semantics: e.g. path = `"v.0.f"` evaluates to 1 on
  `{ "v": [ { "f": 1 } ] }`
- if: `{ "if": [ `《*guard*》`, `《*then*》`, `《*else*》` ] }`
- and: `{ "and": [ `《*operand*$_1$》`, `《*operand*$_2$》`, ... ] }`
- not: `{ "!": [ `《*operand*》` ] }`
- reduce: `{ "reduce": [ `《*operand*》`, `《*lambda*》`, `《*initial*》` ] }`

DSL Consultancy

CertLogic

Operations (2/3)

- equality: `{ "===": [ 《operand₁》, 《operand₂》 ] }`
- membership: `{ "in": [ 《operand》, 《array》 ] }`
- integer and date comparisons:
  { "《operator》": [《operand₁》, 《operand₂》⟦, operand₃⟧] }
- integer plus: `{ "+": [ 《operand₁》, 《operand₂》 ] }`

DSL Consultancy

CertLogic
Operations (3/3)

- working with dates:
  - `{ "plusTime": [ 《operand》, 《amount》, 《unit》 ] }`
    Semantics: e.g. `"2022-04-01" + 713` <u>days</u> `= 2023-03-15`
  - `{ "dccDateOfBirth": [ 《operand》 ] }`
    Semantics: “round up” a partial DOB YYYY⟦-MM⟧ to latest possible date, e.g. `"2002" → 2002-12-31`, and `"2004-02" → 2004-02-29`
- `{ "extractFromUVCI": [ 《operand》, 《index》 ] }`
  Semantics: `("URN:UCI:01:NL:M6B3Y3663FA6REKP6KRL42#9", 2) → "M6B3Y3663FA6REKP6KRL42"`

DSL Consultancy

CertLogic

Operations (4/3)

“Where's my **OR**?!”

Desugaring to the rescue:

$$\{ \texttt{"or"}: [\ \langle\!\langle expr_1 \rangle\!\rangle,\ \langle\!\langle expr_2 \rangle\!\rangle\ ]\ \}$$

$$\equiv \{ \texttt{"if"}: [\ \langle\!\langle expr_1 \rangle\!\rangle,\ \langle\!\langle expr_1 \rangle\!\rangle,\ \langle\!\langle expr_2 \rangle\!\rangle\ ]\ \}$$

CertLogic
Tooling

CertLogic-Fiddle is a minimalistic “IDE”. Features:

- Input: CertLogic expression and data
- Output: validation of expression, evaluation result, compact notation
- Can share examples through URLs, such as this one

CertLogic

Why not make it a “real” DSL (1/2)

Real, as in:

- nice, human-readable syntax (not JSON...)
- editor
- type system (basically JSON Schema)
- IDE
- define and run tests

CertLogic

Why not make it a “real” DSL (2/2)

Reasons:

- Lack of time
- Wildly differing developer environments
- Strong network through EU's eHN
- Debugging using CertLogic Fiddle worked really well
- Set of template rules was provided

CertLogic
Partial evaluation

Goal

Determine automatically which vaccines are accepted by a country, from their business rules

Idea

Mark certain values (`mp`, `dt`) in the ⟪*data*⟫ as `Unknown`

Modify `evaluate` function so it doesn't reduce an ⟪*`expr`*⟫

that would produce `Unknown` (or any value that's not a CertLogic expression)

Usage

Partially evaluate <u>and</u>(⟪*all Acceptance rules of a country*⟫) against a DCC

payload with `dt = Unknown` to derive which vaccines are accepted,
and what their *validity ranges* are

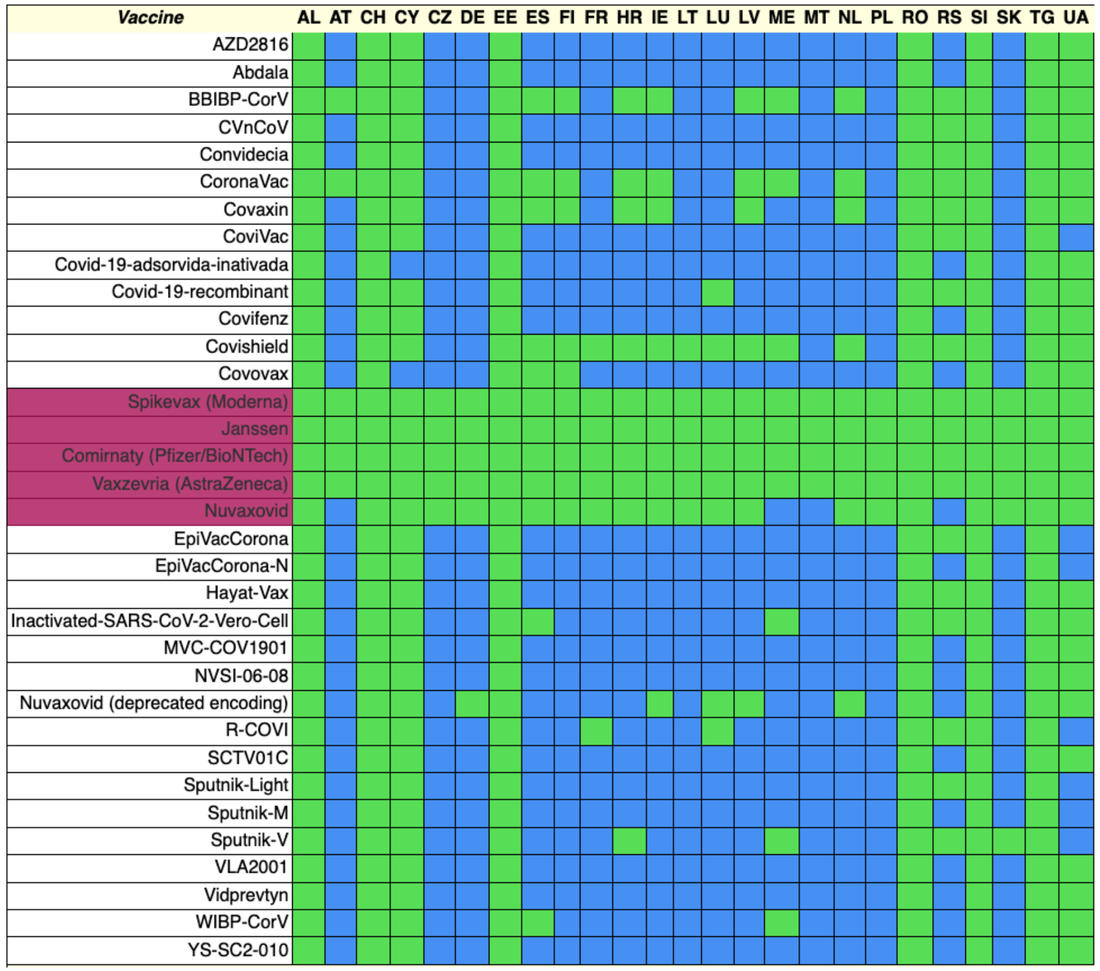
Problem

First have to make `evaluate` endomorphic by extending CertLogic a bit

DSL Consultancy

Analysing business rules

Using partial evaluation



Accepted vaccines	1/1	2/2	2/1	3/3
Luxembourg (LU -)	regs. on Re-open EU		(regs. on	
Covid-19-recombinant, Covishield, Spikevax (Moderna), Janssen, Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Nuvaxovid, Nuvaxovid (deprecated encoding), R-COVI	14-270	0-270	0-366	0-366
Latvia (LV -)	regs. on Re-open EU		(regs. on	
BBIBP-CorV, CoronaVac, Covaxin, Covishield, Spikevax (Moderna), Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Nuvaxovid, Nuvaxovid (deprecated encoding)	15-270	15-270	0-	0-
Janssen	15-270	0-	0-	0-
Montenegro (ME -)	regs. on Re-open EU		(regs. on	
BBIBP-CorV, CoronaVac, Covishield, Spikevax (Moderna), Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Inactivated-SARS-CoV-2-Vero-Cell, Sputnik-V, WIBP-CorV	0-	0-180	0-	0-
Janssen	0-180	0-180	0-	0-
Malta (MT -)	regs. on Re-open EU		(regs. on	
Spikevax (Moderna), Janssen, Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca)	0-	0-270	0-	0-
Netherlands (NL -)	regs. on Re-open EU		(regs. on	
BBIBP-CorV, CoronaVac, Covaxin, Covishield, Spikevax (Moderna), Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Nuvaxovid, Nuvaxovid (deprecated encoding)	14-270	14-270	0-	0-
Janssen	28-270	0-	0-	0-

https://github.com/ehn-dcc-development/eu-dcc-business-rules-analysis

DSL Consultancy

Prescribing business rules

Things that went well

1) **Short time-to-market** In ~2 months from 0 to:
   - CertLogic spec + implementations
   - validation framework - how business rules should be published and run
   - business rules published on EU DCC Gateway by ~25 countries
   - implemented in many verifier apps
2) **Small spec** (and keeping it that way) Allowed quick implementation and controlled evolution, but flexible enough to adapt to changing requirements
3) **Analysis** Analysed rules using language engineering techniques *(partial evaluation)*
4) **Versioning** Versioned specification and implementations independently

DSL Consultancy

Prescribing business rules

Things that could have gone better

1) **Limited scope**
   Only small part of entry regulations “fit” in validation framework, which was hard to extend.

2) **Adoption** Not all countries participating in the EU DCC share their entry regulations using the validation framework. Reasons:

   i) Only small part of entry regulations covered - fear of “false positives”

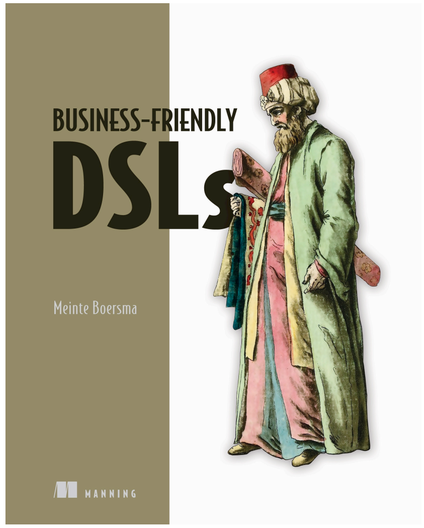
   ii) The *developer experience* (DX) is not so good

DSL Consultancy

Speaker's links

Email: meinte.boersma@gmail.com

Book: https://www.manning.com/books/business-friendly-dsls



GitHub: https://github.com/dslmeinte/

DSL Consultancy