cdisc®

2026 Europe Interchange

THE FUTURE IS CONNECTED: STANDARDS AND AI POWERING DIGITAL TRANSFORMATION

MILAN, ITALY | MAIN CONFERENCE: 20-21 MAY | TRAININGS & WORKSHOPS: 18, 19, & 22 MAY

Toward a Richer Machine-Readable Representation of CDISC Standards and CORE Rules

Thierry Lambert, CEO, AdClin

Agenda

1. Lessons learned from experience
2. Summary
3. Ideas for evolving: DSL examples
4. Conclusion

Where We Come From

We have a long track in computerizing the standards

- convert PDF to annotated HTML
- integrate the CT & the codetables, feedback to CT Team
- build a global model SDTM + SDTMIGs + CT/codetables/relationships + FDA TCG + knowledge from the text:
  - "the values can be Y or null"
    - codelist NY_Y
  - "would not generally be used"
    - variable.use = add_with_care
    - Scale: allowed, conditional, add_with_care, discouraged, forbidden
    - Only "conditional" needs rules

cdisc®

The CDISC Library

When the library was made available, we already had our own model, but we looked deeply into it

- conclusion : our model was richer, although covering only SDTM for clinical trials

The Rule Engine

When the CORE became available, we tried it (on April 2024) and compared to P21

- conclusion:
  - wrong messages (MHENDTC cannot be after RFSTDTC)
  - lots of missing issues seen by P21
  - youth problems?

The CORE Rules

- When we found the CORE rules in the library, we downloaded them and tried to process them
  - i.e., put them in an object model
- Main issues:
  - citation model unnecessarily complex
  - extremely difficult to find the model for BaseCheck, the elementary component of the “Check” key

A Rule Browser

- To better understand, we created a rule browser that presents the rules in a more readable way, linked to the source rules.
- You can find it at www.adclin.com/rules

cdisc®

Rule Browser: An Example

CORE-000001 ← CG0176, TIG0405

IGs	SDTMIG 3.2, 3.3, 3.4; TIG 1.0
Description	Raise an error when IECAT is equal to 'INCLUSION' and IEORRES is not equal to 'N'.
Scope	IE
Check	`IECAT == "INCLUSION" && IEORRES != "N"`
Outcome	IEORRES is not equal to 'N' when IECAT equals 'INCLUSION'.
Rule Type	Record Data
Sensitivity	Record
Executability	Fully Executable
Citations	doc: SDTM v3.4 — section: 6.3.4 doc: TIG 1.0 — section: 2.8.10.13 — item: Assumption 1 > The intent of the domain model is to collect responses to only those criteria that the subject did not meet, and not the responses to all criteria. doc: IG v3.2 — section: 6.3 — item: Assumption 2 doc: IG v3.3 — section: 6.3.4 — item: Assumption 1 > The intent of the domain model is to collect response to only those criteria that the subject did not meet, and not the responses to all criteria.

Diving into the CORE Code

When we did not understand a rule code, we went to the CORE code

- Surprise: no real class “Rule”, the operational parts of the engine get their parameters directly out of the rule JSON/YAML dictionaries.
- Extremely difficult to understand how a parameter (e.g., “wildcard” or “child” in MatchDataset) modifies the behavior of the engine.

Our Diagnosis

- The way the rules actually work is too intertwined with the engine code
- It is too difficult to understand what a rule actually checks and how it works

cdisc®

Our Proposal

- Create a DSL for rule specifications
  - backed by a richer, well-defined model of the standards
  - as easy as possible to understand at first sight
  - that does not ignore performance issues for an engine
- Specify clearly how an engine must behave with these specifications, and what information it has from the standards.
- The engine itself should be implementable in any language: ruby, python, java, rust, etc.

cdisc®

What we have started

- Write an object model for the standards and a submission
- Review all existing rules, and rewrite them with a DSL, with specifications on how the engine should understand them
  - Far less operators (from 92 to about 25)
  - A few functions: length, slice, upcase
  - Limited agregate functions: min, max, any, record_count

cdisc®

Example of a DSL: The Scope

Today, in YAML:

- Classes:
  - Include: (list) noted “ci”
  - Exclude: (list) noted “ce”
- Domains:
  - Include: (list) noted “di”
  - Exclude: (list) noted “de”
  - include_split_datasets: true/false

cdisc®

Scope DSL Proposed

- di=ALL; ce=RELATIONSHIP
  - `all - relationship`
- ce=RELATIONSHIP; de=AP--
  - `all - relationship ap_all`
- ci=FINDINGS; di=IE
  - `IE`
- ci=ALL; ce=SUPP--, AP--; include_split_datasets=true
  - `all:split - SUPP--:split ap_all:split`
- ci=EVENTS; di=MH; include_split_datasets=true
  - `MH MH:split`

Scope DSL: Surprises

- DSL : RELSUB
  - ci=ALL; di=RELSUB
  - ci=RELATIONSHIP; di=RELSUB
  - ci=SPECIAL PURPOSE; di=RELSUB
- DSL : DM
  - ci=ALL; di=DM
  - ci=SPECIAL PURPOSE; di=DM
  - ci=INTERVENTIONS, SPECIAL PURPOSE; di=DM
    - maybe “interventions DM”?
  - ci=RELATIONSHIP, SPECIAL PURPOSE; di=DM
    - maybe “relationships DM”?

Rule Review Results

- Only rules that apply to an SDTMIG for now
- Experience: the DSL allows to understand the rules much better
- Result so far
  - 590 input rules (724 total)
  - About 130 rules dropped:
    - Wrong rules
    - Duplicate rules
    - Covered by another, possibly refactored rule
    - Covered by the generic rule “the variable must follow its type/codelist”

An Example: CORE-000001

Part of YAML code:

```yaml
Description: >-
  Raise an error when IECAT is equal to 'INCLUSION'
  and IEORRES is not equal to 'N'.
Scope:
  Classes:
    Include:
    - FINDINGS
  Domains:
    Include:
    - IE
Check:
  all:
  - name: IECAT
    operator: equal_to
    value: INCLUSION
    value_is_literal: true
  - name: IEORRES
    operator: not_equal_to
    value: "N"
    value_is_literal: true
Outcome:
  Message: IEORRES is not equal to 'N' when IECAT equals
    'INCLUSION'.
```

With DSL:

```
description:
  when IECAT is "INCLUSION",
  IEORRES must be "N"

scope: IE

when: IECAT == "INCLUSION"

ok: IEORRES == "N"  # or fail: IEORRES != "N"

# message: the description by default
```

But CORE-000001 Is Not Needed

In the standards:

- IEORRES is required
  - so it cannot be null, general rule for all required variables
- IEORRES where IECAT == "INCLUSION" has codelist NY_N
  - this is the subcodelist of NY with only N
  - general rule: “a variable must follow its codelist” (null is allowed, but see previous rule)

Comparing datetimes

- Current rule CORE-000714
  - scope: DM
  - fail: RFXSTDTC != null & RFXENDTC != null & RFXSTDTC date_gt RFXENDTC
- Issue:
  - RFXSTDTC = 2016-05-20T08:00
  - RFXENDTC = 2016-05-20
  - String comparison: fail (the engine seems to shorten the 1st)

The Solution

- Date comparisons must use “certainly” or “possibly”
  - if any datetime is missing, everything is possible and nothing is certain
- Rewritten rule:
  - scope: DM
  - fail: RFXSTDTC certainly > RFXENDTC
- Universal logic, won’t fail on the example

Conclusion

- The way the engine works must be crystal clear
- To foster community participation, the rules must be
  - easily accessible on the net, with a query mechanism (“which rules deal with --ENRTPT?”)
  - easily readable and understandable
  - easy to write (after understanding the base logic)
  - easy to test (without constructing whole datasets: these are integration tests)

What's Next

We will open a repository on Github, proposing:

- An object model fed by the standards, the CT and typing information
- A clear engine specification
- Rewritten rules, easier to understand and manage, with an issue tracker to allow discussing them
- A proof-of-concept rule engine (in Ruby)

Grazie!

Thank You!

Domande?

Questions?