

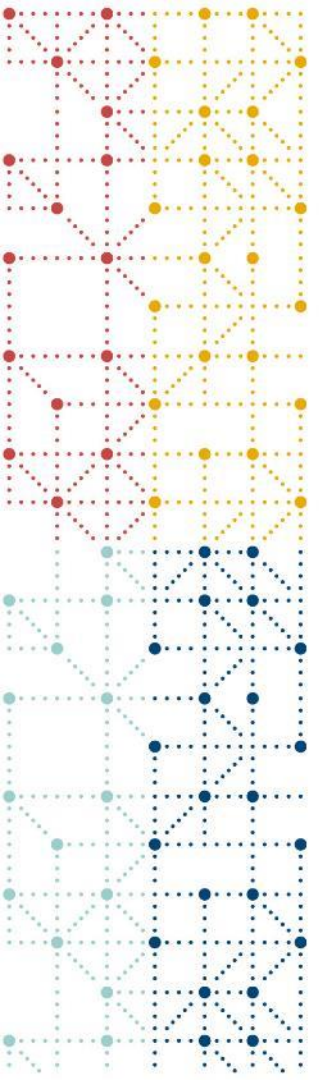
Launching CDISC CORE v1.0: A New Standard for Automated Study Conformance

Tomás Sabat Stöfzel, Verisian, CEO

Nick De Donder, CDISC, CDISC Open Rules Program Manager

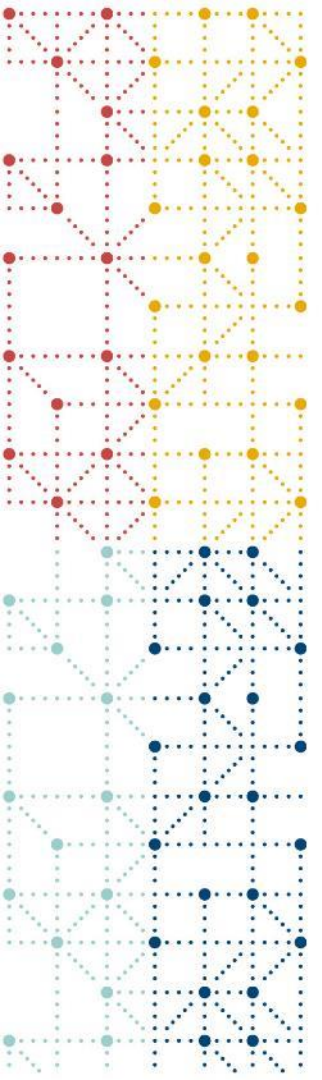
17 November 2025





Agenda

1. Overview & Opportunities
2. V1.0 Scope & Status
3. CORE Refactor Plan
4. Rule Authoring
5. Conclusion



Overview & Opportunities for Improvement



CDISC Open Rules Engine (CORE)

Purpose

- Open-source
- Reference implementation
- Execute conformance rules

Key Features

- Authoring, testing, and execution
- Standardized, transparent validation
- Open-source access to the global clinical research community
- Adapt to evolving regulatory and research needs



CDISC Open Rules Engine (CORE)

Architecture

- Written Python, rules authored in YAML
- Rules engine, command-line interface, and testing framework
- Built on a lightweight adaptation of Venmo engine

Future Vision

- Embedding established query languages (e.g. SQL & JSONata)
- Enhance scalability to support new standards
- Easier rule creation, testing, and maintenance

Key Opportunities & Impact

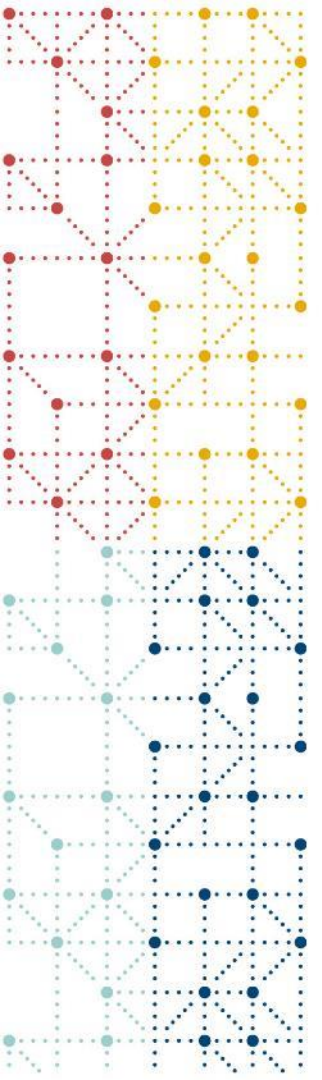
**Reduced Engineering
Dependency**

Improved Scalability

**Fostering
Innovation**

**Increased Flexibility for
Rule Writers**

**Reduced Maintenance
Burden**



CORE V1.0 Scope & Status

Open Rules v1.0 Scope

Standard	# Rules	# Complete	% Complete
SDTMIG v3.3 and 3.4	505	453	89.7%
SENDIG v3.1.1	312	218	69.9%
FDA Business Rules	291	179	61.5%
TIG v1.0	484	406	83.9%
USDM v3.0 and v4.0	259	113	43.6%
Totals	1851	1369	74.0%

Engine

- Support for the rules
- Initial support for ADaM rules
- Refactoring for performance & flexibility

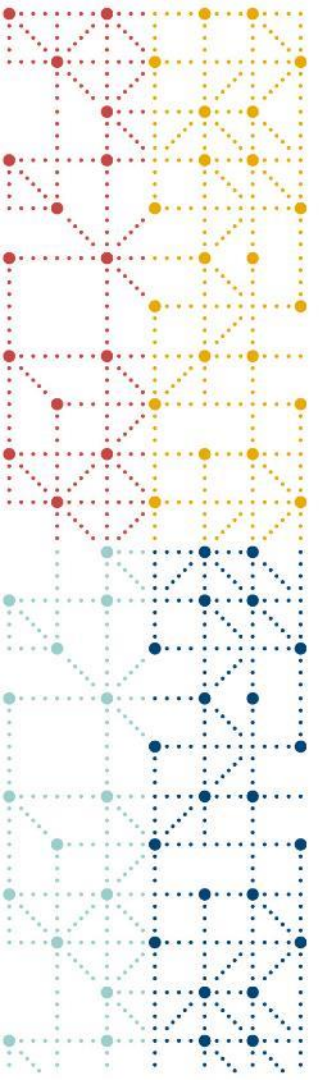
What Follows v1.0

Rules in support of

- All SDTM and SEND versions
- ADaM
- Define-XML
- PMDA Rules
- Biomedical Concepts

Drivers

- Potential AI-Assisted Rule Authoring UI
- Other?



CORE Refactor Plan

A Scalable Path to Smarter, More Maintainable Conformance Rules

We tested CORE on real study data

Running one 12.5gb dataset resulted in...

...CORE failing to run

What's going on?

Over time, the codebase has grown

Creating layers of complexity

Technical debt

Leading to an architecture unsuitable for production-use

The issue of memory management

CORE starts by loading an entire dataset into memory

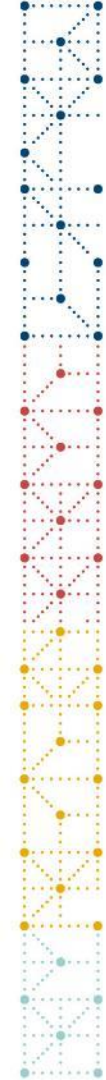
Then keeps creating copies as operators execute

The issue of memory management

CORE starts by loading an entire dataset into memory

Then keeps creating copies as operators execute

So that 12.5gb dataset is multiplied many times over in memory



The issue of memory management

CORE starts by loading an entire dataset into memory

Then keeps creating copies as operators execute

So that 12.5gb dataset is multiplied many times over in memory

You just can't fit that much into memory

A better way to scale

3.5k+ direct references to pandas objects throughout operators

Data is being manipulated in multiple disconnected places

There is no separation between data operations and business logic

A better way to scale

3.5k+ direct references to pandas objects throughout operators

Data is being manipulated in multiple disconnected places

There is no separation between data operations and business logic

Result is redundant processing, high memory use, and slower execution

Issues of abstractions

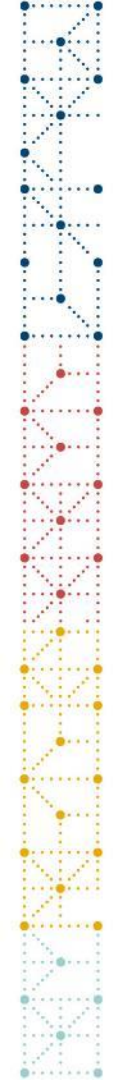
Data and business logic need to be separated

- Fetching data should be in one layer
- Business rules like validation checks need to live in another

Without this separation, reusing components, testing logic, and scaling performance become almost impossible

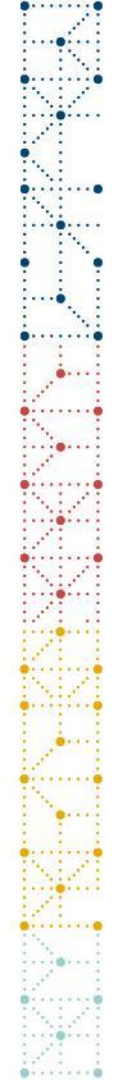
Great software isn't built once, it's rewritten continuously

How could we at Verisian help?



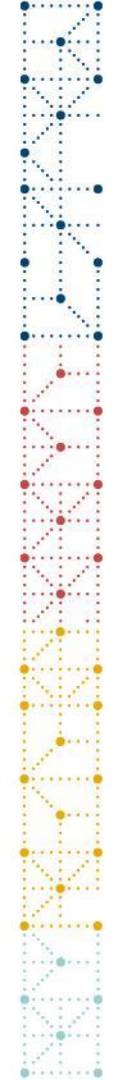
How could we at Verisian help?

Let's do the rewrite together!



The goal

Build a production-ready engine



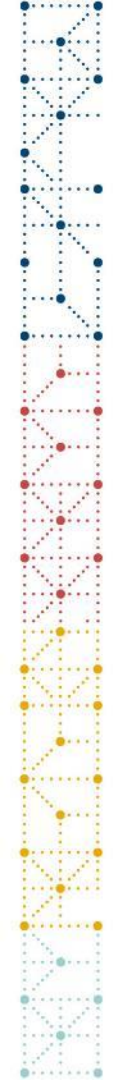
The refactor

Decouple business and data logic

Modularize the architecture

Move all database operations into SQL

One unified SQL layer to handle all the database work



No more loading entire datasets into memory, multiple times

Re-implemented all the rule operators within a modular SQL structure

Consolidated redundant functions into a clean, standardized set

Created the foundations flexible enough to support future operations and use cases

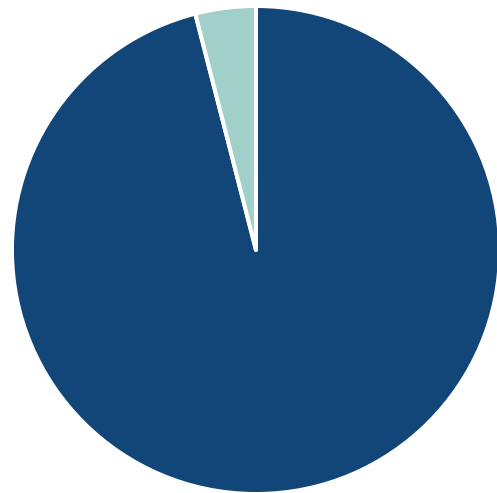
Regression testing

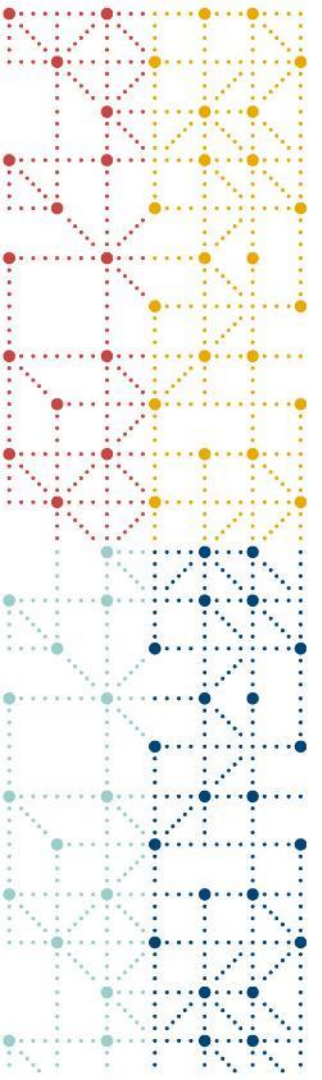
Creating a validation and testing foundation with regression tests to ensure behavior stays stable over time

Discovered a published rule with a deprecated check operator

Status

- New engine running 96% of existing rules successfully
- Matching the old engine expected November-end
- Full integration testing on large studies coming soon





Rule Authoring

We need your help

Only 56 rules from finishing SDTM

Join as a volunteer as we enter the final sprint...

Demystifying rule authoring

When DSCAT = 'PROTOCOL MILESTONE', DSTERM and DSDECOD will contain the same value drawn from the sponsor's controlled terminology.

(SDTMIG v3.2, Section 6.2, Assumption 3d)

Demystifying rule authoring

If DSCAT = "PROTOCOL MILESTONE"

Then DSTERM must equal DSDECOD

Demystifying rule authoring

If **DSCAT** = "PROTOCOL MILESTONE"

Then DSTERM must equal DSDECOD

```
Check:  
  all:  
    - name: DSCAT
```

Demystifying rule authoring

If DSCAT = "PROTOCOL MILESTONE"

Then DSTERM must equal DSDECOD

```
Check:
  all:
    - name: DSCAT
      operator: equal to
```

Demystifying rule authoring

If DSCAT = "PROTOCOL MILESTONE"

Then DSTERM must equal DSDECOD

```
Check:
  all:
    - name: DSCAT
      operator: equal_to
      value: PROTOCOL MILESTONE
```

Demystifying rule authoring

If DSCAT = “PROTOCOL MILESTONE”

Then **DSTERM** must equal DSDECOD

```
Check:
  all:
    - name: DSCAT
      operator: equal_to
      value: PROTOCOL MILESTONE
    - name: DSTERM
```

Demystifying rule authoring

If DSCAT = “PROTOCOL MILESTONE”

Then DSTERM **must equal** DSDECOD

```
Check:
  all:
    - name: DSCAT
      operator: equal_to
      value: PROTOCOL MILESTONE
    - name: DSTERM
      operator: not equal to
      value: DSDECOD
```

Demystifying rule authoring

If DSCAT = "PROTOCOL MILESTONE"

Then DSTERM must equal **DSDECOD**

```
Check:
  all:
    - name: DSCAT
      operator: equal_to
      value: PROTOCOL MILESTONE
    - name: DSTERM
      operator: not_equal_to
      value: DSDECOD
```

Demystifying rule authoring

If DSCAT = "PROTOCOL MILESTONE"

Then DSTERM must equal DSDECOD

```
Check:
  all:
    - name: DSCAT
      operator: equal_to
      value: PROTOCOL MILESTONE
    - name: DSTERM
      operator: not_equal_to
      value: DSDECOD
```

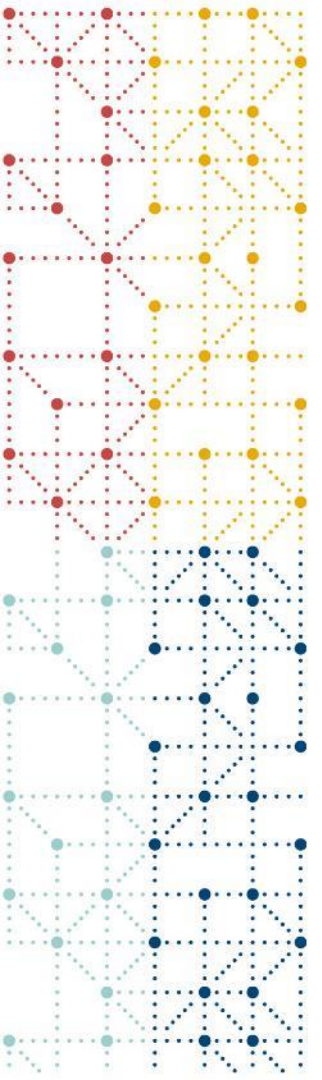
Could this be any easier?

Demystifying rule authoring

Message: DSCAT equals PROTOCOL MILESTONE but DSTERM is not equal to DSDECOD

Demystifying rule authoring

```
4 Authorities:
5   - Organization: CDISC
6   Standards:
7     - Name: SDTMIG
8     Version: '3.2'
9   References:
10    - Origin: SDTM and SDTMIG Conformance Rules
11    Rule Identifier:
12      Id: CG0066
13      Version: '1'
14      Version: '2.0'
15    Citations:
16      - Cited Guidance: d. When DSCAT='PROTOCOL MILESTONE',
17        DSTERM and DSDECOD
18        will contain the same value drawn from the sponsor's
19        controlled
20        terminology.
21        Document: IG v3.2
22        Item: Assumption 3d
23        Section: '6.2'
24 Check:
25   all:
26     - name: DSCAT
27       operator: equal_to
28       value: PROTOCOL MILESTONE
29       value_is_literal: true
30     - name: DSTERM
31       operator: not_equal_to
32       value: DSDECOD
33 Core:
34   Id: CORE-000322
35   Status: Published
36   Version: '1'
37 Description: When DSCAT equals PROTOCOL MILESTONE, then DSTERM should be
38 equal to
39 DSDECOD
40 Executability: Fully Executable
41 Outcome:
42   Message: DSCAT equals PROTOCOL MILESTONE but DSTERM is not equal to
43   DSDECOD
44   Output Variables:
45     - DSTERM
46     - DSDECOD
47 Rule Type: Record Data
48 Scope:
49   Classes:
50     Include:
51       - EVENTS
52   Domains:
53     Include:
54       - DS
55 Sensitivity: Record
```



Conclusion

Conclusion

- LinkedIn group: CDISC CORE Collaboration
- Volunteer



LinkedIn Group



Volunteer Form





Thank You!

tomas@verisian.com

ndedonder@cdisc.org

