phuse·virtual

Single Day Event

CDISC Open Rules Engine

Everything new is well-forgotten old

Dmitry Kolosov
Parexel

Disclaimer

- *The views and opinions expressed in this presentation are those of the author(s) and do not necessarily reflect the official policy or position of CDISC, Microsoft or any other company participating in the development of CORE.*

Conformance Rules

Standards are now a vital part of our industry. And conformance rules are an integral part of standards.

Why do we need conformance rules?

- Improve Quality
- Verify consistency
- Speed up development
- Streamline acceptance

Conformance Rule Challenges

"In theory, theory and practice are the same.
In practice, they are not."

- Rules and implementation are written by different people
- Different implementations might have different results
- Implementation can happen long after a standard is released
- Rule writing process does not include automated testing
- It can be challenging to update a rule even if it is wrong

What to do if a standard is followed but a validator triggers an issue?
What should be commented, rule or implementation?

Rule Requirements

- Developed together with standards
- Governed by CDISC/Community/Health Authorities
- Open and available for everyone
- Metadata driven
- Machine and human-readable
- Executable

Rule Governance

- Developed together with standards
- Governed by CDISC/Community/Health Authorities
- Open and available for everyone
- Metadata driven
- Machine and human-readable
- Executable

Rule Governance

Standard Conformance Rules are developed by CDISC teams

- SEND - SEND Conformance Rules v4.0 (411 rules)
- SDTM - Conformance Rules v1.1 for SDTMIG v3.2 and v3.3 (477 rules)
- ADaM -ADaM Conformance Rules v3.0 (525 rules)
- Define-XML - Conformance Rules for Define-XML v2.1 (225 rules)
- And more

Define-XML 2.1 Rules Example

Rule Identif	Plain Text Rule	Rule	Rule Message	Applicable Versions	Source Type	XPaths	Element	Attribute
38	No more than one child element def:AnnotatedCRF must be provided for the MetaDataVersion element.	No more than one //MetaDataVersion/def:AnnotatedCRF must be provided.	Child element def:AnnotatedCRF is provided more than once for element MetaDataVersion.	Both 2.0 and 2.1	Schema	/ODM/Study/MetaDataVersion/def:AnnotatedCRF	def:AnnotatedCRF	
39	Child element def:DocumentRef must be provided for the def:AnnotatedCRF element.	//def:AnnotatedCRF/def:DocumentRef must be provided.	Child element def:DocumentRef is missing for element def:AnnotatedCRF.	Both 2.0 and 2.1	Schema	/ODM/Study/MetaDataVersion/def:AnnotatedCRF/def:DocumentRef	def:AnnotatedCRF/def:DocumentRef	
42	Attribute leafID must be provided for the def:DocumentRef element.	@leafID must be provided.	Attribute leafID is missing for element def:DocumentRef.	Both 2.0 and 2.1	Schema	/ODM/Study/MetaDataVersion/def:SupplementalDoc/def:DocumentRef	def:DocumentRef	leafID

Pros

- Explains the rule in a simple language
- Contains a formal algorithm
- Includes a rule message
- Can be grouped by element and attribute
- Distinction between 2.0 and 2.1
- A lot of rules are already implemented in a schema

Cons

- Not executable/automated
- Requires implementation for full conformance

Single Source of Truth

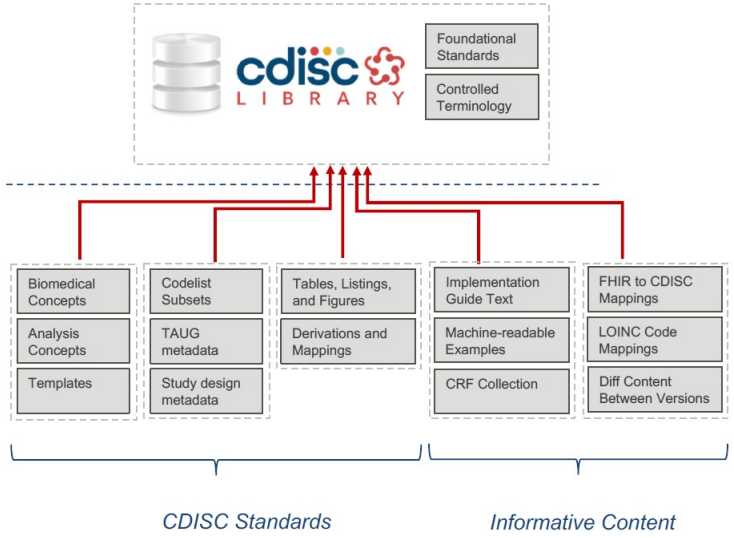
- Developed together with standards
- Governed by CDISC/Community/Health Authorities
- Open and available for everyone
- Metadata driven
- Machine and human-readable
- Executable

Single Source of Truth

How to make rules open and available for everyone?

CDISC Library

- Standards
- Codelists
- Conformance Rules
- And more



Machine and human-readable

- Developed together with standards
- Governed by CDISC/Community/Health Authorities
- Open and available for everyone
- Metadata driven
- Machine and human-readable
- Executable

YAML

Yet Another Markup Language

JSON

```json
{
  "event": "Phuse SDE",
  "publisher": "Dmitry",
  "slides": [{
    "name": "YAML",
    "page": "11",
    "description":
    "YAML is yet another markup language"
    }]
}
```

XML

```xml
<presentation>
  <event>Phuse SDE</event>
  <publisher>Dmitry Kolosov</publisher>
  <slide>
    <name>YAML</name>
    <page>11</page>
    <description>
      YAML is yet another markup language
    </description>
  </slide>
</presentation>
```

YAML

```yaml
presentation:
  event: Phuse SDE
  publisher: Dmitry
  slides:
    - name: YAML
      page: '11'
      description: >
        YAML is yet another
        markup language
```

Rule Example

```yaml
CoreId: VariablePresenceExample
Version: "1"
Authority:
  Organization: CDISC
Reference:
  Origin: SDTM Conformance Rules
  Version: "1.1"
  Id: CG0058
Description: Raise an error when --ENRTPT exists in a dataset, --ENTPT does not exist.
Sensitivity: Dataset
Scopes:
  Standards:
    - Name: SDTMIG
      Version: "3.3"
  Classes:
    Include:
      - All
  Domains:
    Include:
      - All
Rule Type:
  Variable Presence:
    Check:
      all:
        - name: "--ENRTPT"
          operator: "exists"
        - name: "-ENTPT"
          operator: "not_exists"
Outcome:
  Message: --ENRTPT exists in a dataset, but --ENTPT does not exist.
Citations:
  - Document: SDTM v1.4
    Section: 2.2.5 Timing Variables for All Classes
    Cited Guidance: "(--ENTPT) Description or date/time in ISO 8601 or other character format of the sponsor-defined reference point referred to by --ENRTPT."
```

Description

Scope

Algorithm

Message

Description

Executable

- Developed together with standards
- Governed by CDISC/Community/Health Authorities
- Open and available for everyone
- Metadata driven
- Machine and human-readable
- Executable

CORE Application Structure

CDISC Open Rule Engine (OSS)

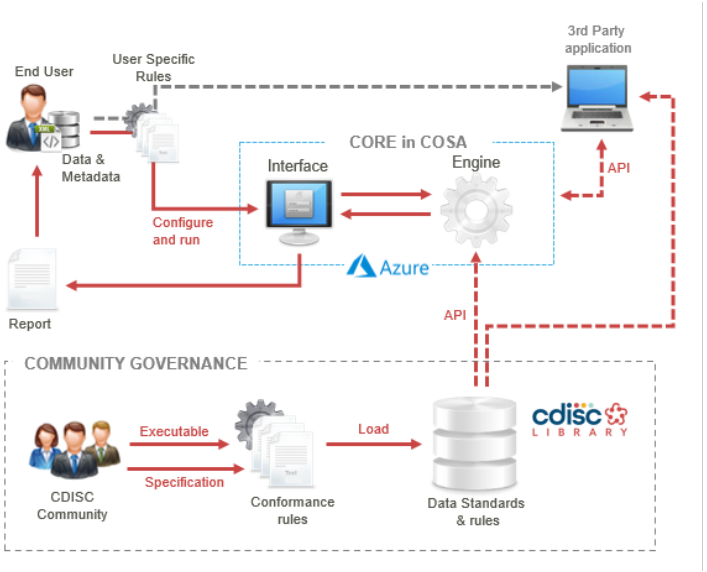
Runs rules and creates a validation report

CORE Backend

Stores data, study information and reports

User Interface (OSS)

Allows user to interact with the backend and rules engine, review reports



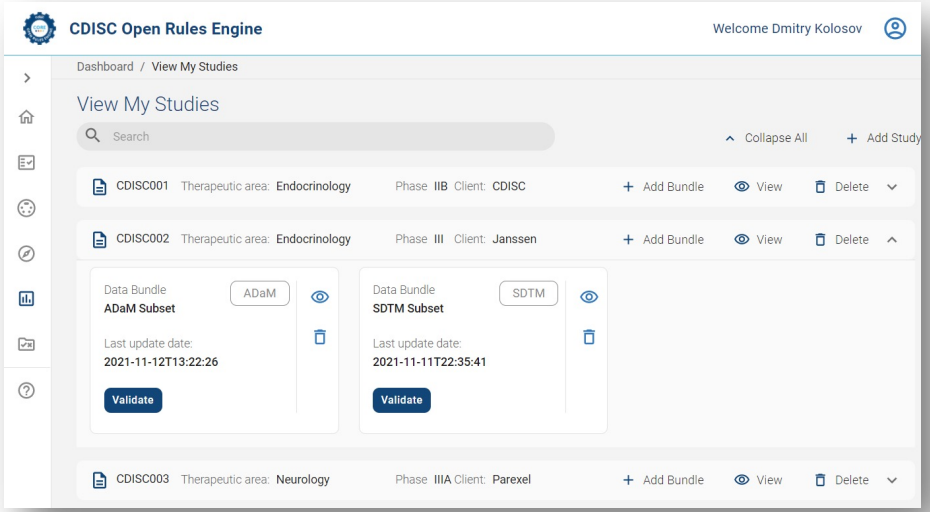
CDISC Open Rule Engine

CORE is an engine which allows to process rules

- Open-Source
- Written in Python
- Can be integrated with other systems
- Cross-platform
- Not a commercial project

User Interface

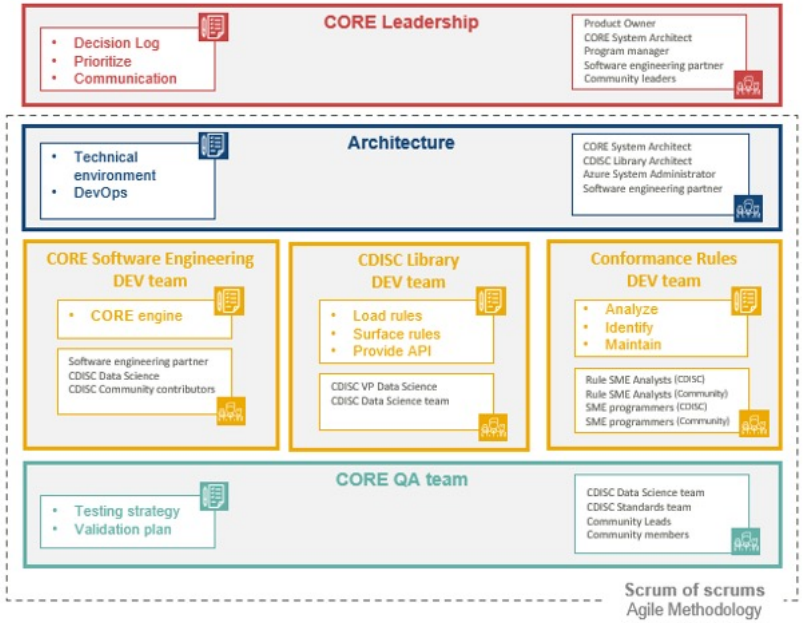
- Written in TypeScript
- Open-Source
- API and Authentication functionality is implemented in separate classes
- Report review
- Issue annotations



CORE Team Structure

Teams

- Stakeholders
- Architecture
- Software Engineering
- Library Integration Team
- Conformance Rule Development Team
- QA / Testing Team

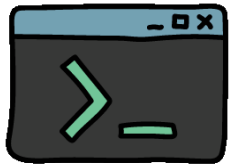


CORE is developed by the community. Everyone can participate.
Details at https://www.cdisc.org/core

Future Possibilities

Open-source license advantages:

- Users can submit pull requests for open-source components
- MIT allows both free and commercial usage



Ideas of how CORE and UI can be further developed by a community:

- CLI – call CORE from R/SAS code
- Desktop App – reuse UI and CORE with Electron