

MTCONNECT

VERSION 2.0

POWERED BY  **AMT**

MTCONNECT VALIDATION
MODEL BASED STANDARD VALIDATION USING GRAPHS
April 16, 2026

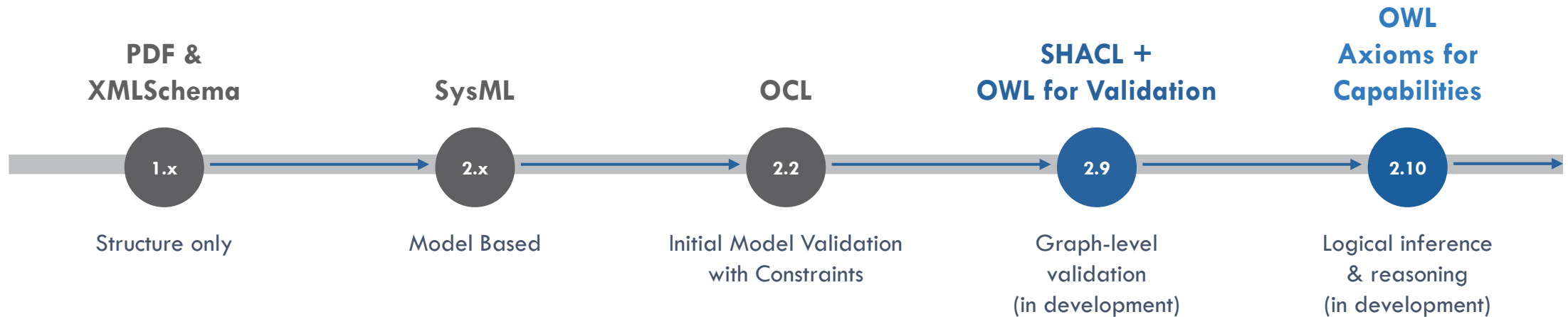
William Sobel

Co-Founder – Metalogi

Chief Architect - MTConnect Institute

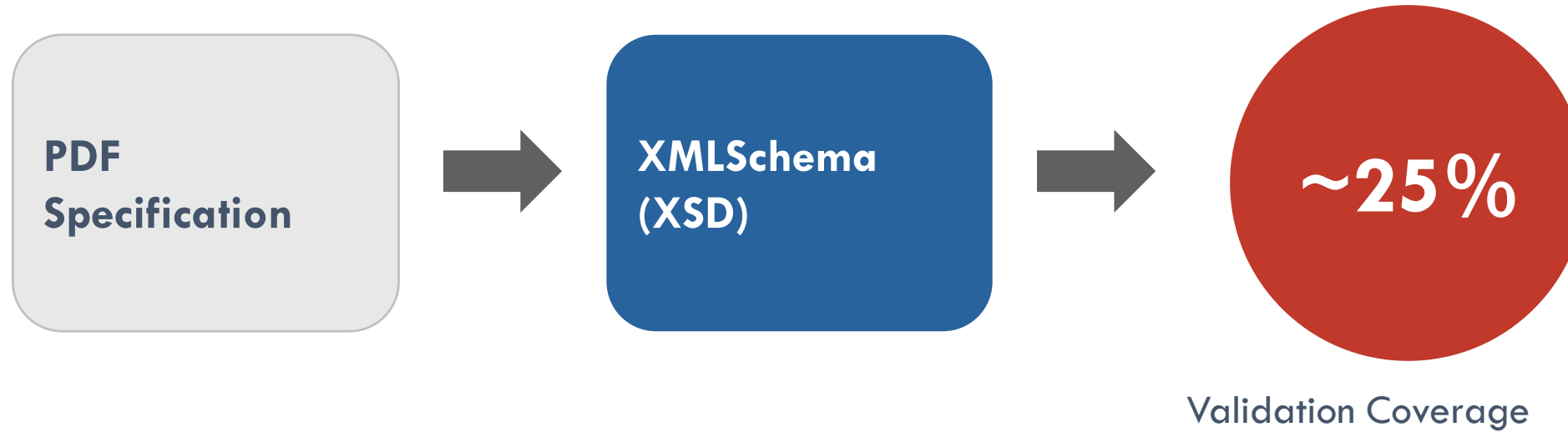
Chair: Architect WG – Industrial Ontologies Foundry

The Journey of MTConnect Validation



Increasing validation coverage and model fidelity →

The Early Days: PDF & XMLSchema



Prose requirements • Tree-structured validation • No cross-entity rules

Moving to SysML

**SysML Model
(Cameo)**

Can Validate

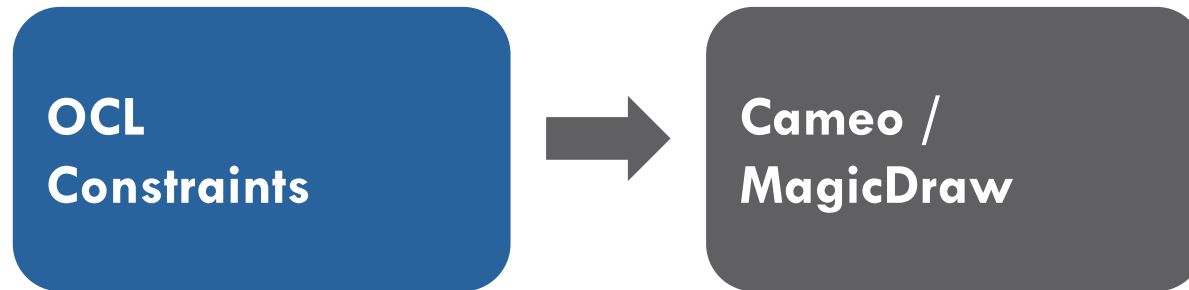
- ✓ Property Types
- ✓ Cardinality
- ✓ Inheritance

Cannot Validate

- ✗ Business Rules
- ✗ Cross-Entity
- ✗ Runtime Data

Better model definition — but still poor validation coverage

Adding OCL Constraints



```

Components::Devices::Device::allInstances()->iterate(device;devicecount:Real=0|
if device.observes->size() >= 3
then
  if device.observes->iterate(av;avail:Real=0|if av.ocAsType(Dataltems::"Dataltem Types)::Event).type =
DataTypes::EventEnum::AVAILABILITY then avail+1 else avail+0 endif) = 1
  then if device.observes->iterate(ac;assetc:Real=0|if ac.ocAsType(Dataltems::"Dataltem Types)::Event).type =
DataTypes::EventEnum::ASSET_CHANGED then assetc+1 else assetc+0 endif) = 1
  then if device.observes->iterate(ar;asetr:Real=0|if ar.ocAsType(Dataltems::"Dataltem Types)::Event).type =
DataTypes::EventEnum::ASSET_REMOVED then asettr+1 else asettr+0 endif) = 1
    then devicecount + 1
    else devicecount + 0
    endif
  else devicecount + 0
  endif
else devicecount + 0
endif
endif) = hasDevice->size()
  
```

Complex Syntax

Vendor Locked

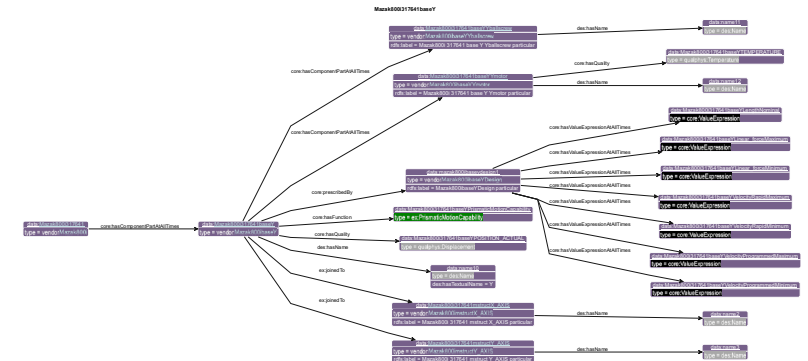
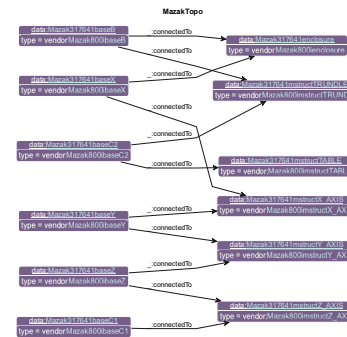
Few Tools

Expressive power — but locked in a single tool with complex syntax

Models Are Graphs – Why not use Graph tools?

Information models are inherently graphs of interconnected entities. Not trees. Not flat schemas. Validation needs to work at the graph level.

The Graph extends beyond single documents.



Enter SHACL: Graph-Level Validation

Open
Standard

SHACL

Graph-Level
Validation

W3C Recommendation (2017)

Multiple
Implementations

```
val:ControllerWithControllerMode
a sh:NodeShape ;
sh:class mt:Controller ;
sh:property [
  sh:path mt:observes ;
  sh:qualifiedValueShape [
    sh:class mt:ControllerModeDataItem
  ] ;
  sh:qualifiedMinCount 1 ;
] .
```

SPARQL
Integration

```
val:DeviceShouldHaveController
a sh:NodeShape ;
sh:targetClass mt:Device ;
sh:property [
  sh:severity sh:Warning ;
  sh:message "Device should have a
    Controller with ControllerMode" ;
  sh:path [
    sh:oneOrMorePath mt:hasComponent
  ] ;
  sh:qualifiedValueShape val:ControllerWithControllerMode ;
  sh:qualifiedMinCount 1 ;
] .
```

Shapes Constraint Language — validating the entire graph

Converting the Model to OWL



Built on W3C Open Standards

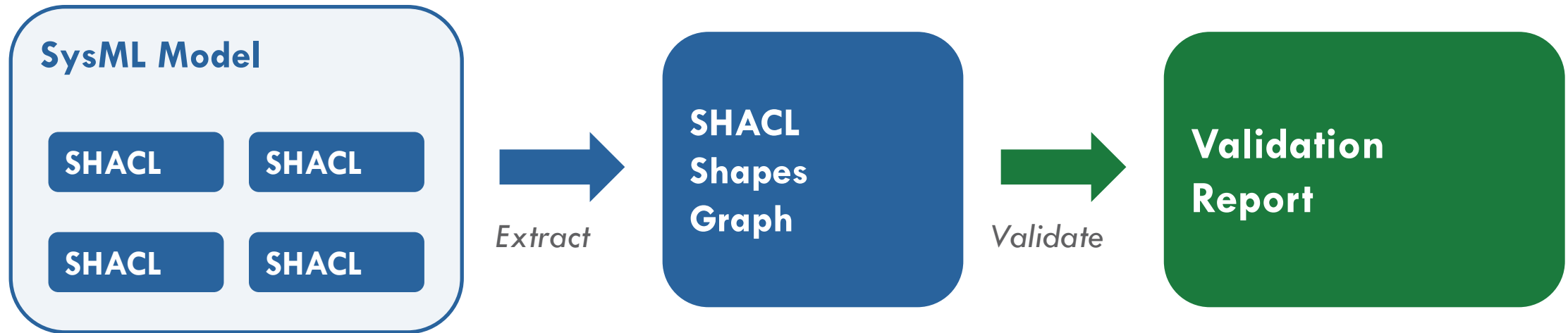
RDF

RDFS

OWL

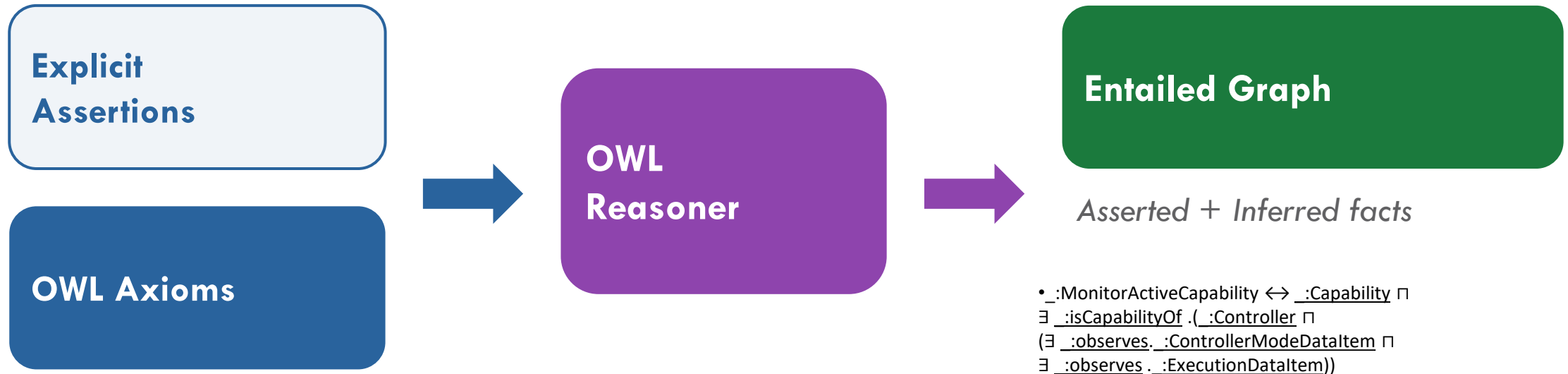
SPARQL

Embedding SHACL in the Model



Constraints live with the model — extracted and validated automatically

OWL Axioms: Logical Constraints & Inference



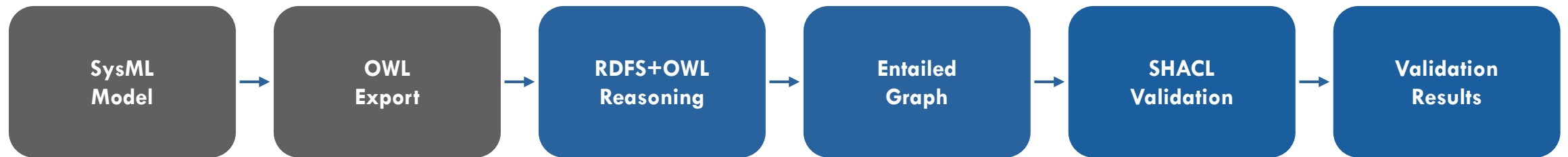
Infer capabilities from structure — classify by what devices can do using use cases

Disjointness

Equivalence

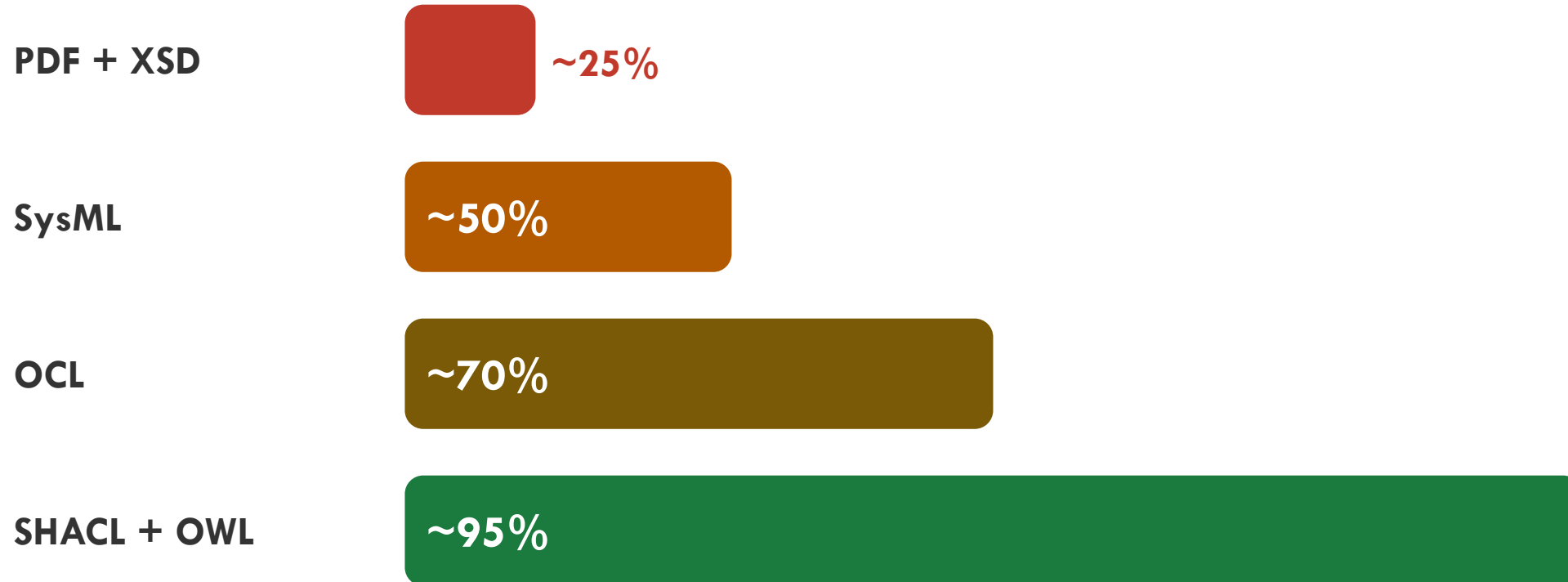
Restrictions

The Complete Validation Pipeline



SHACL constraints embedded in SysML model → extracted during OWL export → fed to validator

Where We Are Today



A model that validates itself — constraints are part of the specification

Questions?

William Sobel

Metalogi.io

will@metalogi.io

+1-510-684-6400

