MTCONNECT

VERSION 2.0

POWERED BY AMT

MTCONNECT VALIDATION & CAPABILITIES

April 29, 2026

William Sobel

Chief Architect - MTConnect Institute (AMT)

Chair: Architect WG – Industrial Ontologies Foundry

Co-Founder – Metalogi

William Sobel

William Sobel has been architecting complex systems across various industries for over 35 years. For the last 19 years, Mr. Sobel has served as the Chief Architect and chair of the Executive Committee of the MTConnect Standard, an international model-based standard for manufacturing equipment and the IIoT. Additionally, he leads the Industrial Ontologies Foundry Architecture Working Group, which advances support for cross-domain industrial knowledge and explainable AI. While leading the architecture of the MTConnect standard, Mr. Sobel was the CEO of VIMANA, a leading industrial analytics platform. Before VIMANA, he worked as a visiting lecturer at UC Berkeley, teaching agile web development and data center optimization using machine learning at RadLAB. Before UC Berkeley, Mr. Sobel worked in the financial industry for 17 years. He served as VP and Chief Architect at MSCI-Barra, leading a team that was architecting the industry's first and leading SaaS financial risk management software.

Information Model Standards

- Purpose: Communicate information (data) from one system to another
- Provide minimal domain semantics to tag and interpret data
- Do this efficiently to minimize bandwidth and complexity
- They have issues:
  - Trustworthiness: Documents are considered correct if they conform to a schema; they do not verify the information makes sense
  - Usability: The implementation does not include the axioms in the model
  - Commitment: They lack the ability to express any commitment to any theory

Standards and Interoperability

- *Statistics: "All models are wrong, but some are useful" – George Box*
- *Information Models: "All models are underspecified, but some are useful"*
- Standards under-commit leading to ambiguity & interoperability issues
- Ontologies provide semantics
- Exploring two areas to gain traction for ontologies:
  - *Validation*
  - *Capabilities*

Validation

Verifying the model and the data

(Information models only focus on data)

The Journey of MTConnect Validation

PDF & XMLSchema	**SysML**	**OCL**	**SHACL + OWL for Validation**	**OWL Axioms**
1.x	2.x	2.2	2.9	2.10
Structure only	Model Based	Initial Model Validation with Constraints	Graph-level validation (in development)	Logical inference & reasoning (in development)

Increasing validation coverage and model fidelity →

The Early Days: PDF & XMLSchema

PDF Specification → **XMLSchema (XSD)** → **~20%**

Validation Coverage

- *Prose requirements*
- *Tree-structured validation*
- *No cross-entity rules*
- *No cross-attribute constraints*
- *Limited taxonomy*
- *No Commitment*

- *Structure*
- *Data Format / Controlled Vocabulary*
- *Cardinality*
- *ID Uniqueness*
- *ID References (in same doc)*

Hit a wall

- Could not model types and type relationships
- Could not express relations across between documents (models)
- Could not express the property type
- Model definitions were split between XSD and PDF
  - Neither contained the entire model
- We were constrained by what we could express in XSD
- We could only validate structure and format of documents
- XSD is NOT a modeling language

Transition to SysML and Model Normative

SysML Model (Cameo)

Can Validate	Cannot Validate
✓ Data Types	✗ Mereology
✓ Cardinality	✗ Transitivity
✓ Specialization	✗ Instance Data
✓ Behavior	✗ Metaproperties

Better model definition

All documentation lives in the model (no more PDF); Model is machine readable

Still needs additional levels of validation

Adding OCL Constraints

OCL Constraints → **Cameo / MagicDraw**

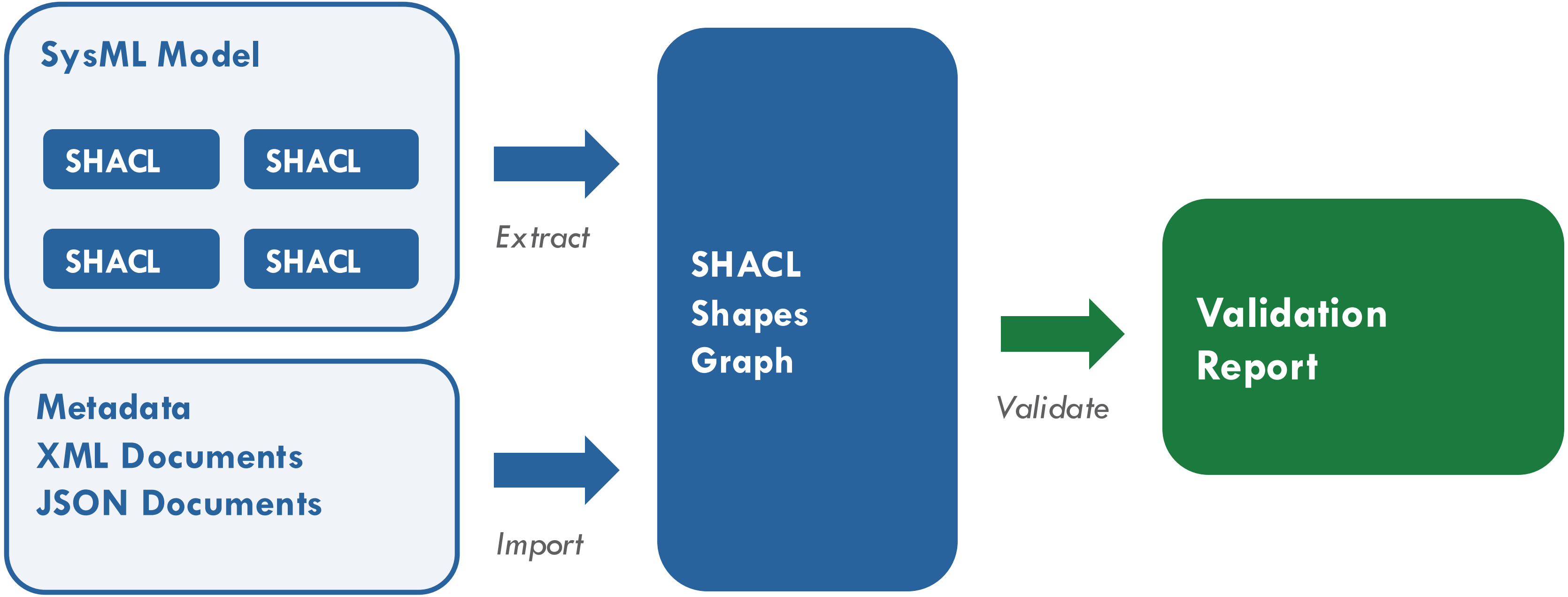
- **Complex Syntax**
- **Vendor Locked**
- **Few Tools**

Expressive power — but locked in a single tool with complex syntax
Individuals do no live in the SysML model

Hit a wall with SysML

- We could not validate conformance of the model to style requirements
- Could not express
  - Theory
  - Disjointedness/Equivalence
  - Sufficient Conditions
  - Shapes
  - Property subsumption
- There are no particulars!
  - SysML does not make importing particulars easy
- **We cannot validate the model (does it make sense?)**

Phase 1: RDFS w/ Embedding SHACL in the Model



Constraints live with the model — extracted and validated automatically
Instance data can be validated using information models

Validating the Document & Validating the Model

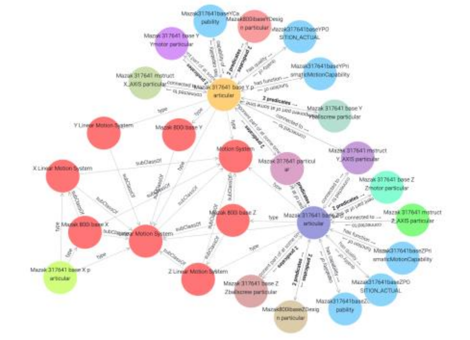
- Ontology validation
  - Style and rule constraints for definitions
  - Conformance and consistency for construct naming
  - Etc…
- Particular validation
  - Component mereology and topology are valid
  - Component names provided and are correct (make sense)
  - Etc…

MTCONNECT

Phase 2: Translating the Model to OWL

SysML Model → **OWL Ontology** →

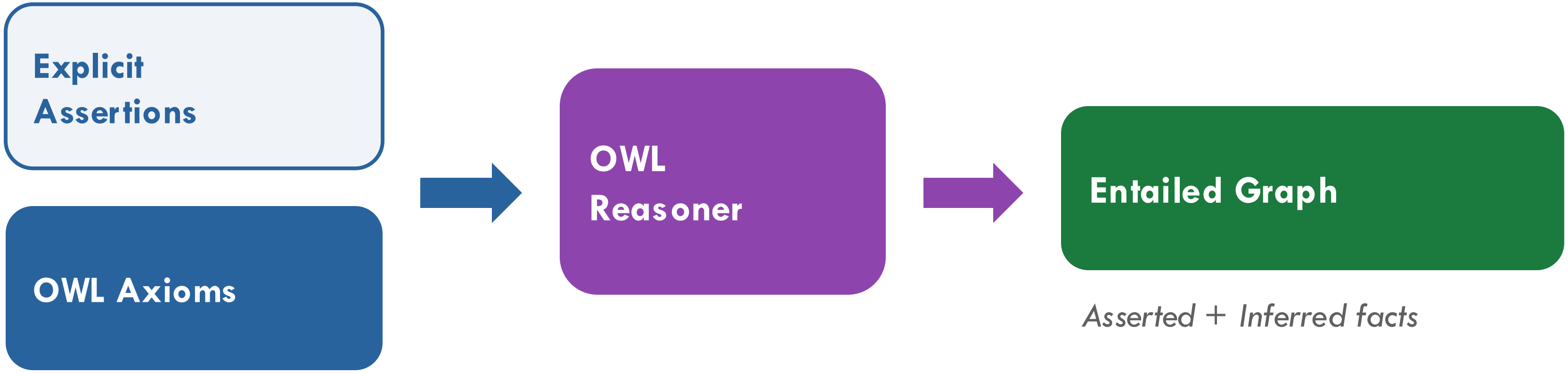
RDF Graph



Mapping the MTConnect model to IOF
BFO as TLO and IOF as MLO including other ontologies like SE, PPS, Maintenance, etc.

Do not Replace the Information Model with the Ontology
Axioms and Constraints built into the SysML model

OWL Axioms: Logical Constraints & Inference

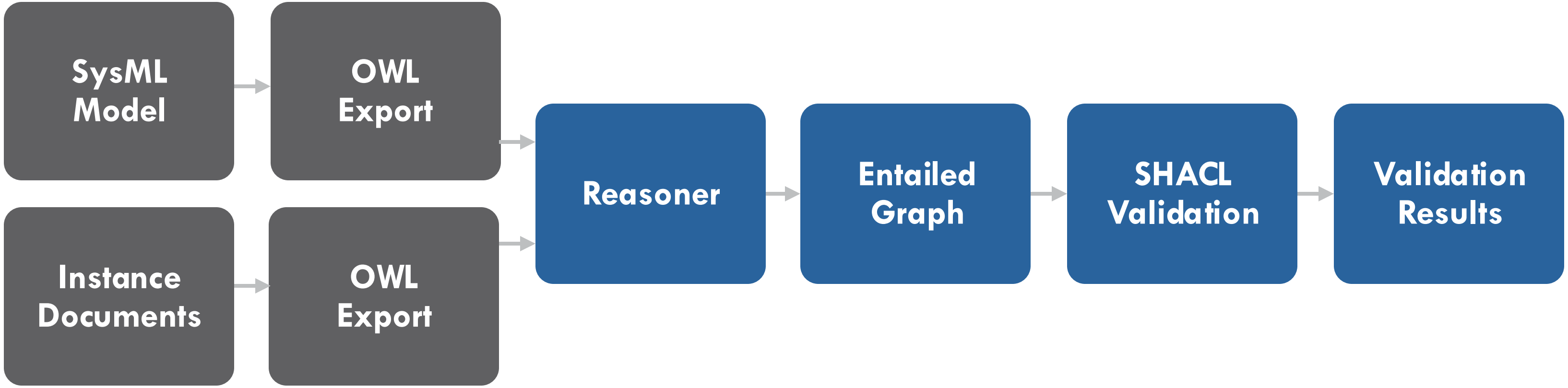


Disjointness

Equivalence

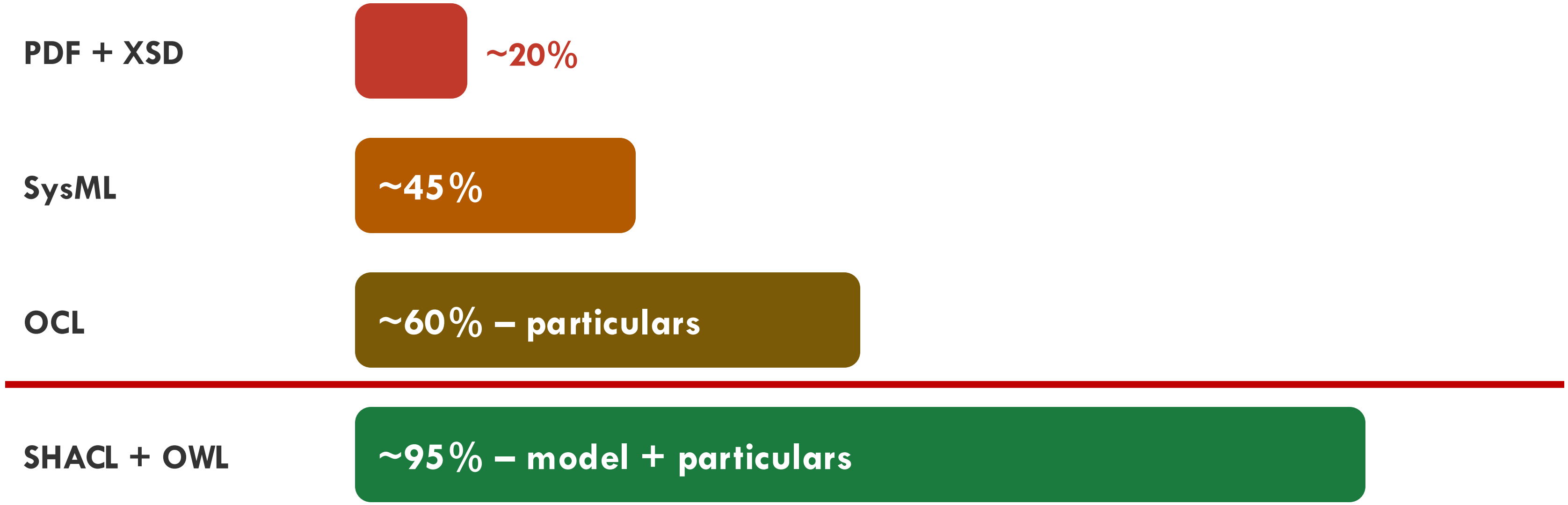
Restrictions

The Model Validation Pipeline



SHACL constraints embedded in SysML model → extracted during OWL export → fed to reasoner
Reasoner checks for logical inconsistency and violations
Run SHACL on entailed graph to check additional type and particular constraints

MTC Validation Today (Only particulars)



A model that validates itself. The constraints are part of the specification
Currently WIP for RDFS export and SHACL constraints

Interoperability and the Digital Thread

- A mapping $M : A \rightarrow B$ between conceptual models inherits Inconsistencies(A) ∪ Inconsistencies(B) ∪ Inconsistencies(M).
- Where A lacks identity criteria, M must invent them.
- And, where we have two data sources: S and T from different providers
  - S and T interpret the identity criteria differently
- Where B permits non-equivalent representations of the same proposition, M must pick one.
  - S uses prop U, and T uses prop V
- Where either schema underspecifies, M must add axioms neither author endorsed.
- The composed mapping is therefore not a translation between two ontologies; it is a *third* ontology, usually undocumented, whose commitments are the union of three sources of error.

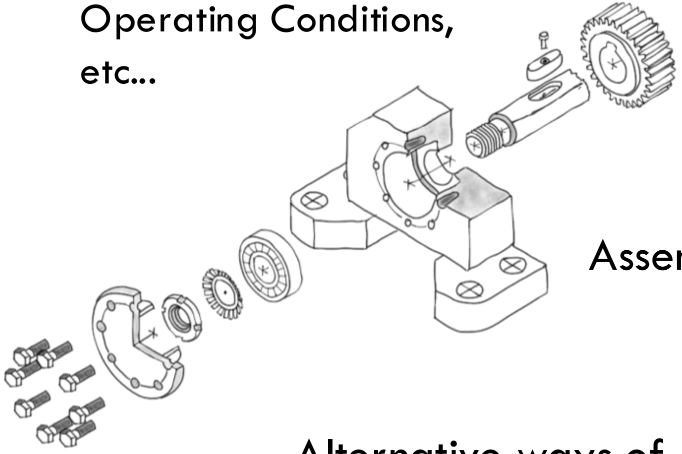
Use Case: Capabilities

Agile manufacturing and the digital thread

Agile Manufacturing

Product Requirements

Function, Geometry,
Operating Conditions,
etc...



Assembly

Mfg Requirements

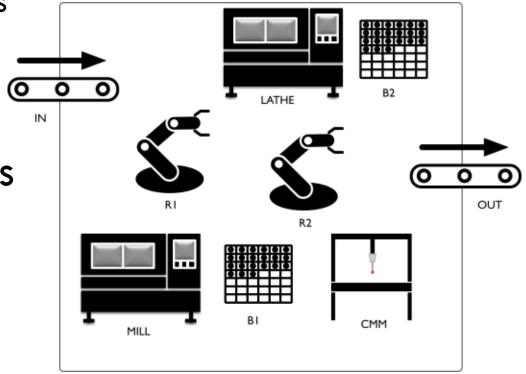
Mass, Material,
Tools, Load, HP,
etc…

requires

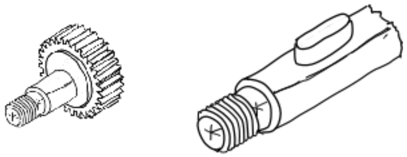
Capabilities

matches

Has Capabilities



Alternative ways of manufacturing parts



Capabilities Across the Enterprise

Capabilities live at every level

Enterprise Capabilities
Emergent from shops and business systems

Aggregated capabilities at the shop or enterprise level: "Can we satisfy the requirements for this DOD aerospace product?"

Shop Capabilities
Emergent from machines, personnel, tools, and technology

Transitive relationship to equipment that matches business need: "Can engineer, machine, and inspect titanium aerospace parts"

Individual Machine
Emergent from parts

Realized capabilities: what this configured device can actually do given its current condition: "Can perform 5 axis machining on titanium"

Component / Part
Asserted Capabilities

Asserted at the part/component level: "Can perform continuous revolute motion with 1,500 Nm of force." "Coolant through the tool at 80 bar."

Capabilities are emergent through transitive relationships of systems to their parts

Digital Thread is a synthesis of business, engineering, manufacturing, inspection, use, maintenance, and reuse/recycle systems

Simple Example for a Machine Tool

mtc:System ⊑ ∃ constr:hasComponentPartAtSomeTime .bfo:independent_continuant
mtc:System ⊑ ∃ constr:hasCapability .constr:Capability

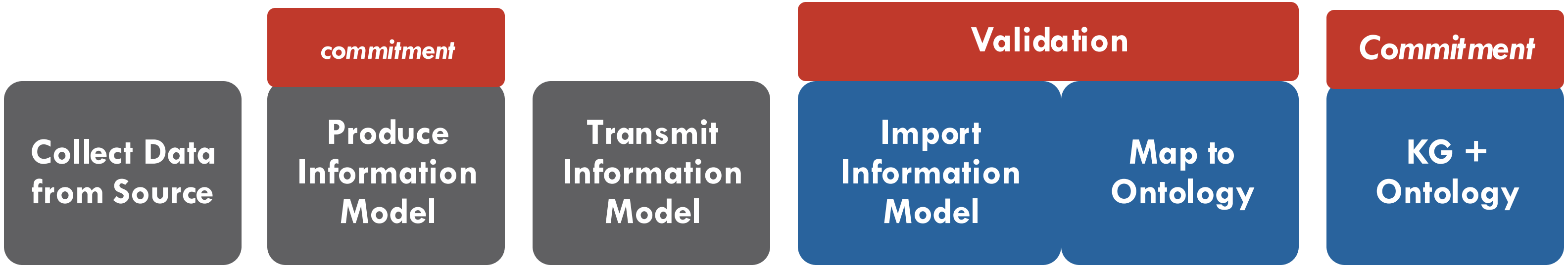
mtc:LinearMotionSystem ⊑ ∃ constr:hasCapability .mtc:PrismaticMotionCapability

mtc:ThreeAxisMotionCapability ↔ constr:Capability ⊓ ∃ constr:capabilityOf .(mtc:Machine ⊓ ∃ mtc:hasComponent .mtc:XLinearMotionSystem ⊓ ∃ mtc:hasComponent .mtc:YLinearMotionSystem ⊓ ∃ mtc:hasComponent .mtc:ZLinearMotionSystem ⊓ ∃ mtc:hasComponent .(mtc:CRotaryMotionSystem ⊓ ∃ constr:hasFunction .mtc:ContinuousRevoluteCapability))

mtc:FourAxisMotionCapability ↔ mtc:ThreeAxisMotionCapability ⊓ ∃ constr:capabilityOf .(mtc:Machine ⊓ ∃ mtc:hasComponent .(mtc:RotaryMotionSystem ⊓ ∃ constr:hasFunction .mtc:IndexedRevoluteCapability))
mtc:FourAxisMotionCapability ⊑ mtc:ThreeAxisMotionCapability

mtc:Cell ⊑ bfo:material_entity
mtc:Cell ⊑ ∃ constr:hasRole .mtc:SystemRole

The Complete Pipeline



Model definitions enhanced to remove ambiguity
Perform minimal commitment when creating information models
Communicate data using information models
Define transformations from Information Model to Ontology
Transformation become normative

Where do we go from here…

- Can't and won't replace information models
- First, provide model-based standards for information models
  - Maps to document structure and provides intermediate definition
- Sell why an ontology is necessary
  - Focus on concerns like validation
  - Find use cases to elucidate concerns like capabilities
- Standards organizations create principled ontologies that commit to a well-defined theory
- They will need lots of help
- AI will be the driver. AI cannot resolve ambiguity.

Questions?

William Sobel

MTConnect Institute

wsobel@amtonline.com

+1-510-684-6400