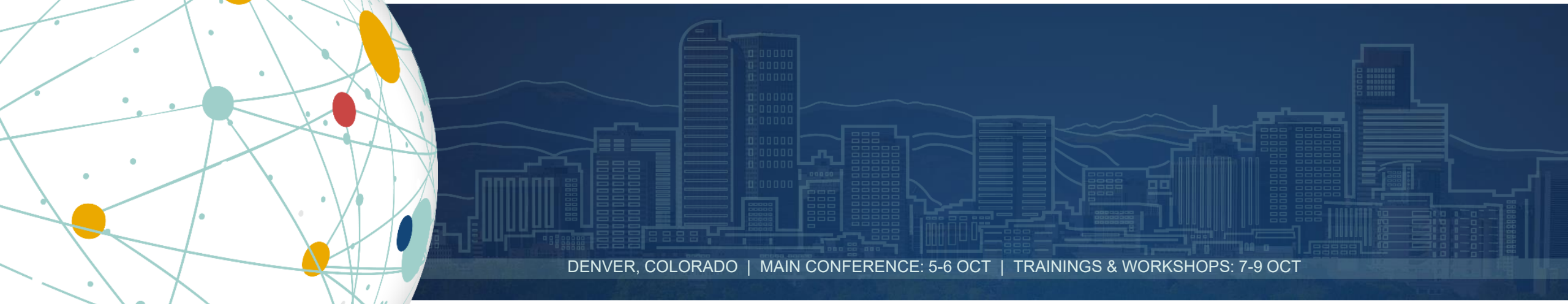


cdisc®

2026 US Interchange

THE FUTURE IS CONNECTED: STANDARDS AND AI POWERING DIGITAL TRANSFORMATION



From Human-Readable Mappings to Machine-Verifiable SDTM Transformation Specifications with AI-Assisted Authoring

Rostislav Markov, Amazon Web Services, Inc. | Janet Low, Merck & Co., Inc.

2026 CDISC US Interchange, Denver, Colorado

What this talk covers

1

Make the specification machine-verifiable

Why mapping spreadsheets cannot be checked by software, and what a machine-verifiable specification adds.

2

Agree on the specification, not the engine

How a deterministic assembly step keeps one specification portable across SAS, R, Python, and partner organizations.

3

Let AI draft; approved software decides

Where AI fits safely: AI drafts, an approval gate decides, and nothing runs without validation and approval.

Merck's Clinical Data Layer (CDL) and the pattern underneath it

≈4,000
SDTM updates per year

60–80%
cross-study mapping reuse baseline, standardized studies



CDL TODAY: IN PRODUCTION

What CDL does today

- Serverless on AWS; capabilities are Python functions
- Specifications authored in a web UI, stored as versioned JSON
- Registry of capability metadata for discovery and reuse
- Software assembles the workflow from the specification; every run is traced end to end
- AI-suggested mappings appear as drafts for programmer review

THE PATTERN UNDERNEATH: THIS TALK

What the pattern makes possible

- The machine-verifiable specification is the shared agreement; engines and languages stay open
- One specification across SAS, R, Python, vendor platforms
- A shareable machine-readable asset across companies
- AI drafts within a control boundary; approved software decides

CDL keeps the two apart: the specification says what a transformation should do (the intent); approved capabilities say how it is coded. The rest of this talk describes that pattern, independent of any platform.

Before CDL: the specification as a set of spreadsheets

	A	B	C	D
	Source dataset	**Source variable**	**Target variable**	**Mapping / derivation logic (free text)**
1	AE_RAW	AESTDT	AE.AESTDTC	Convert to ISO 8601; keep partial dates
2	AE_RAW	AETERM	AE.AEDECOD	Code to MedDRA preferred term
3	AE_RAW	AESEV	AE.AESEV	Map to controlled terminology (AESEV)
4	DM	RFSTDTC	AE.AESTDY	Study day from AESTDTC and DM.RFSTDTC

Mapping_AE | Reference: terminology, codelists | Metadata: domains, variables

Source-to-target mappings without code, plus reference and metadata spreadsheets: useful collaboration artifacts, written for people.

Written for people to interpret
The transformation logic lives in free text; its meaning depends on the reader.

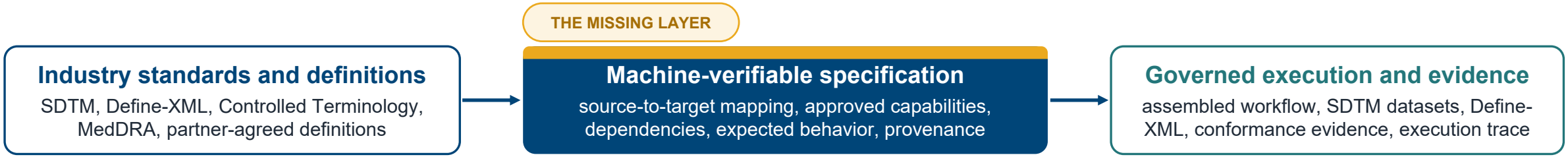
Cannot be checked before code runs
Software cannot confirm that what it describes is executable, complete, or approved.

Re-implemented by every team
Each sponsor, partner, or language team (SAS, R, Python) re-codes the same intent.

No safe handoff for AI
An AI-drafted spreadsheet cannot be validated by software, so AI stays informal or becomes a prompt.

A spreadsheet is machine-readable; the logic inside it is not. Column D is free text: software can open the file but cannot check what it says. Machine-verifiable means the same content is structured so software can validate it before anything runs and any engine (SAS, R, or Python) can execute it.

Design principle: agree on the specification, not the engine



AGREE ON

What the specification carries

- source and target references
- transformation and derivation rules
- dependencies and ordering
- approved capabilities and parameters
- expected behavior and constraints
- version, provenance, approval state

KEEP OPEN

Implementation remains open

- SAS, Python, R, SQL, or other languages
- sponsor, vendor, or open-source tooling
- workflow engine and orchestration technology
- registry implementation
- cloud or on-premises deployment
- AI-assisted or human authoring

Same specification, any engine: **SAS** **R** **Python** **SQL**

Interoperability comes from shared definitions and shared validation rules, not from a mandated runtime.

The specification references industry standards and definitions; it does not depend on any single standard.

A machine-verifiable transformation specification: the same content, in a form software can check

Illustrative structured specification

```
study: ABC-123
source: AE_RAW
target: SDTM.AE

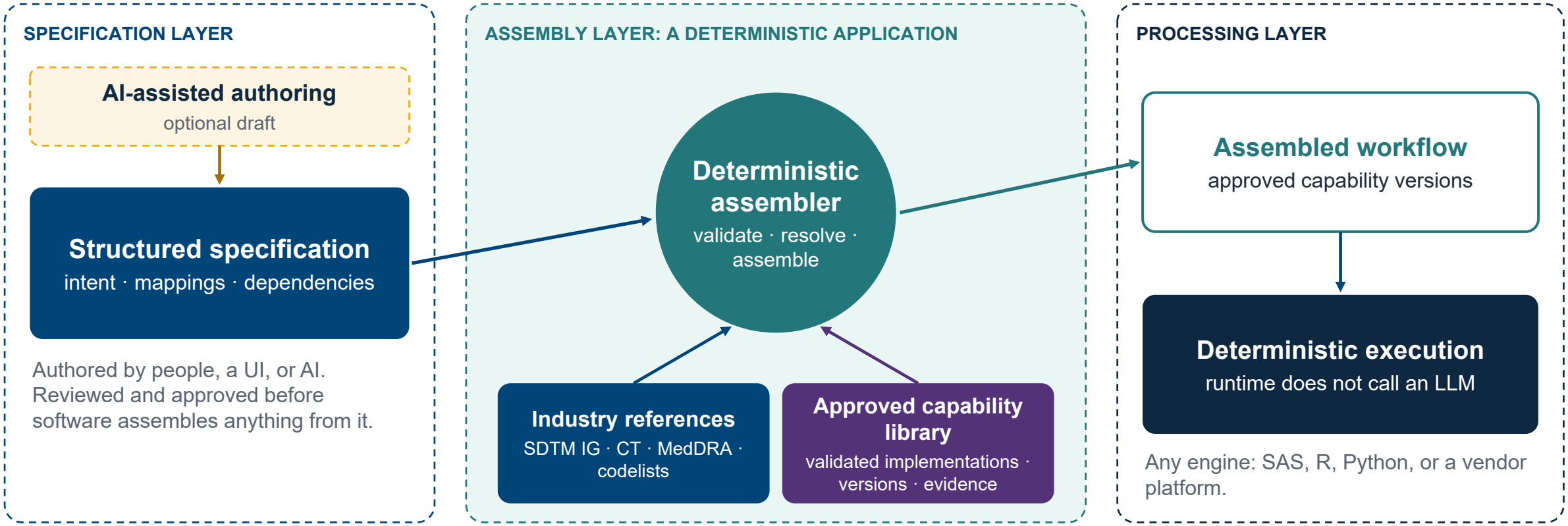
steps:
  - transform: format_date
      input: AESTDTC
      output: AESTDTC
  - transform: map_term
      input: AETERM
      terminology: MedDRA

depends_on: [DM]

policy: approved-only
```

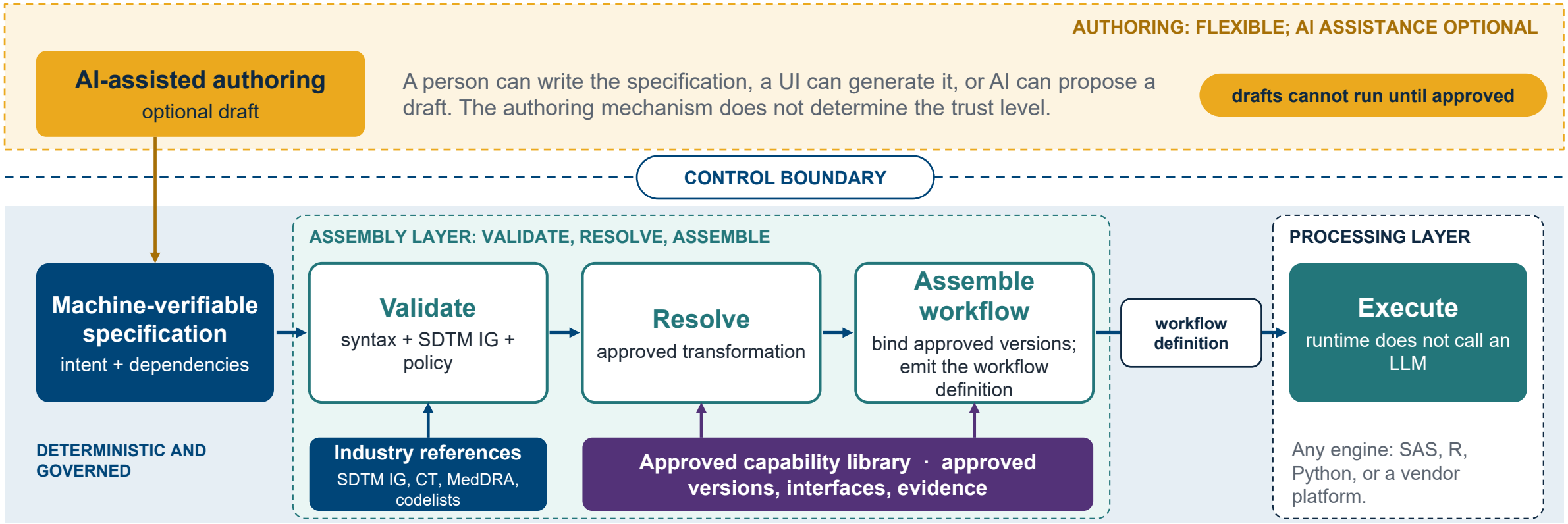
1. **Intent**
   Named source + target elements and expected transformation.
2. **Transformation rule**
   Which approved transformation applies, with its parameters and expected behavior.
3. **Dependencies**
   Required domains, metadata, terminology, and ordering constraints.
4. **Governance**
   Version, provenance, approval state, and execution eligibility.

The pattern: the specification says what, approved building blocks say how, software assembles the workflow



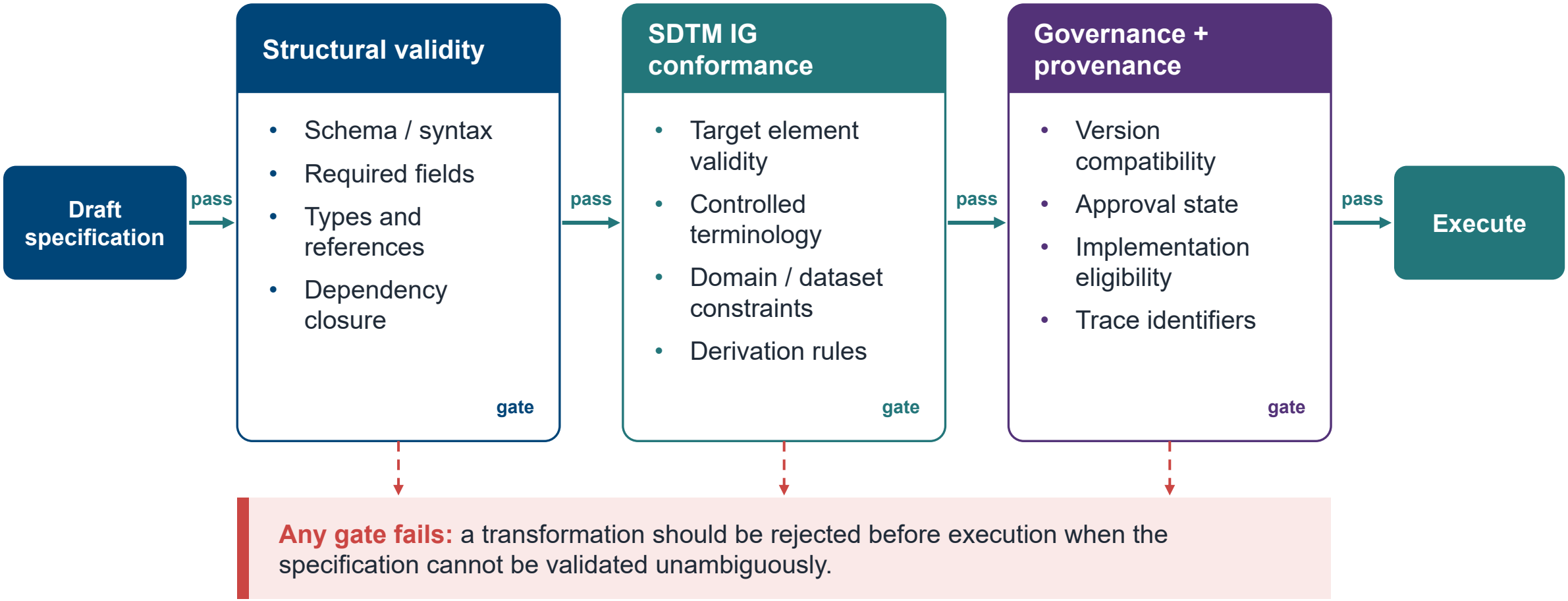
This pattern is Specification-Driven Composition: the specification is the input, approved transformation functions are resolved from a governed library, and a deterministic step assembles the workflow. Merck's Clinical Data Layer runs it in production.

A clear control boundary enables controlled AI adoption in regulated workflows



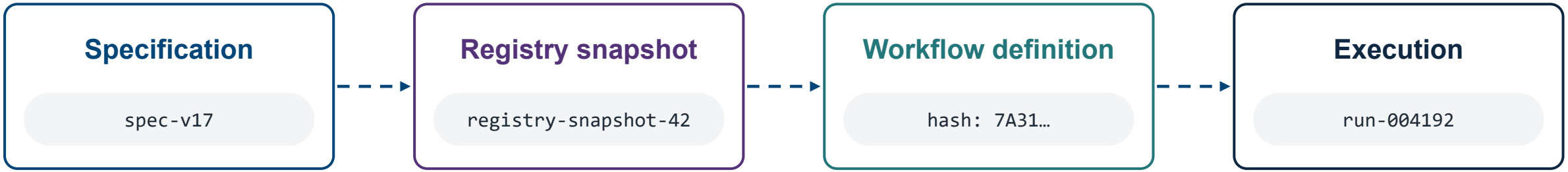
AI may draft; only approved specifications run. The specification is a governed input to software, not a prompt to a model: the assembly step validates, resolves, and assembles, and never interprets.

Pre-execution validation creates a control point before anything runs



Every execution traces back to the approved records that authorized it

EXPLAINABILITY FROM APPROVED RECORDS: THE MINIMUM REPRODUCIBLE TRACE



Record
the approved specification version and the exact capability-registry snapshot used for assembly.

Persist
validation results and the generated workflow definition or its canonical hash.

Link
execution logs and outputs back to those records for review and reproducibility.

Deterministic assembly: the same approved specification and the same registry snapshot produce the same workflow definition, so every run can be explained and reproduced from governed records. Data outputs can still differ if source data changes.

What the specification and the assembly step each deliver

Machine-verifiable specification

- **Reviewable specification:** people still review it; software can validate it.
- **Errors caught before execution:** ambiguous or unapproved specifications are rejected at the gate, not in production.
- **Controlled AI adoption:** AI drafts inside a control boundary; nothing runs without validation and approval.
- **Governed:** version, provenance, and approval state travel with the specification.
- **Portable:** one specification across SAS, R, and Python; shareable with partners and joint ventures.

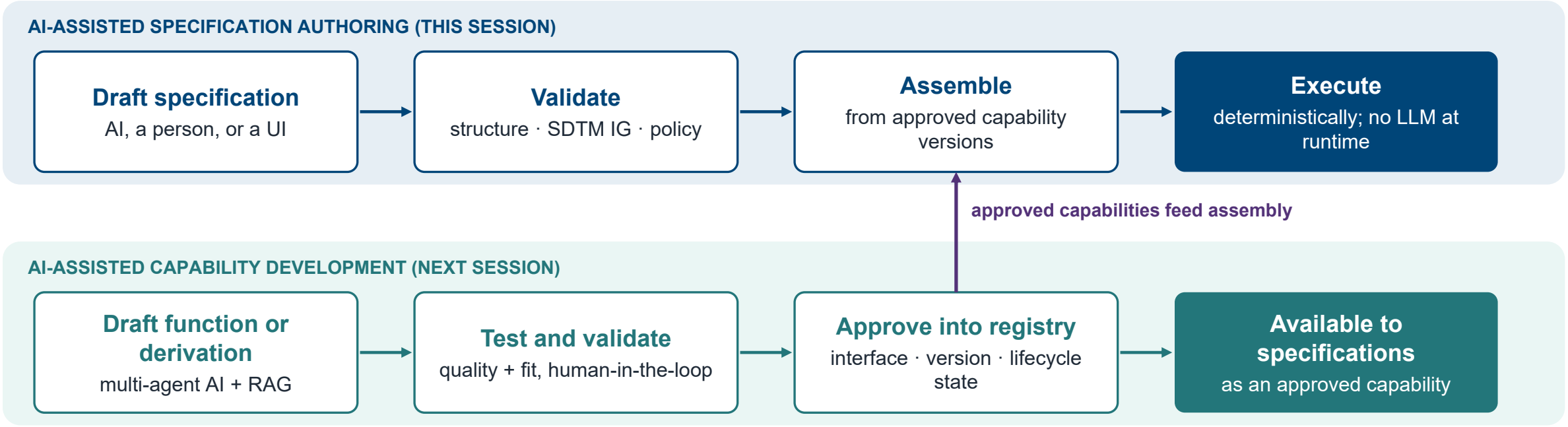
generalization

Deterministic assembly step

- **Reproducible:** same specification + same registry snapshot = same workflow; no LLM in the runtime path.
- **Governed reuse:** only approved capability versions run; capabilities are managed independently of studies.
- **Traceable and explainable:** every run links back to the records that authorized it.
- **Faster study setup:** select and configure approved capabilities instead of re-coding them.
- **Change isolation:** capabilities evolve independently; studies pin the approved versions they need.

All three objectives in one place: a specification software can check, one specification for any engine and any partner, and AI that drafts behind an approval gate.

AI can draft specifications and new capabilities; approval gates decide what runs



Same control principle in both lanes: AI drafts, an approval gate decides, and only approved specifications and capabilities run.

14:00 | Proma Debnath and Srinivas Malipeddi, Merck & Co., Inc. *"Accelerating SDTM Engine Development and Adoption with a Multi-Agent AI Framework and RAG-Based Assistant"*

Three takeaways

All three come from one pattern: Specification-Driven Composition.

1

Make the specification machine-verifiable

Keep the content people already write in mapping spreadsheets, but structure it so software can validate it before anything runs.

2

Agree on the specification, not the engine

Make the rules, dependencies, expected behavior, and provenance portable while SAS, R, Python, and platform choices stay open.

3

Let AI draft; approved software decides

AI can propose specifications and capabilities, but validation, assembly, and execution stay deterministic and governed.

Questions?

Human-readable mappings can become machine-verifiable specifications without giving up review by people.

Made machine-verifiable, the specification is portable, executable, and governed before anything runs.

UP NEXT AT 14:00
Proma Debnath and Srinivas Malipeddi, Merck & Co., Inc.: Accelerating SDTM Engine Development and Adoption with a Multi-Agent AI Framework and RAG-Based Assistant

Rostislav Markov
Amazon Web Services, Inc.
rmarkov@amazon.com

Janet Low
Merck & Co., Inc.
janet_low@merck.com

The pattern, in depth:
AWS Architecture Blog



Appendix: scope and positioning

An architecture pattern, not a proposed standard

Specification-Driven Composition is an architecture pattern: the specification carries what a transformation should do, governed capabilities carry how it is coded, and a deterministic assembly step sits between them. Merck's Clinical Data Layer is a production use case of it in clinical data processing. This is not a proposal for a new CDISC standard.

Definitions from established standards

The specification references SDTM, Define-XML, Controlled Terminology, dictionaries such as MedDRA, and sponsor or partner-agreed definitions. It does not depend on any emerging standard, and no endorsement of a specific standard is implied.

Shared definitions, open runtime

The pattern deliberately separates shared definitions from runtime implementation. The specification should not mandate SAS, Python, R, a cloud platform, or a specific workflow engine.

AI-assisted authoring is optional

AI-assisted authoring is optional. The core idea is a machine-verifiable specification that can be validated, governed, resolved to approved capabilities, and deterministically assembled before execution.