DMCommunity Challenge Apr-2026 Submissions Review

Perspective: Decision Management and AI Governance Practitioner

Prepared by: Jeremiah Connelly, Novus-Forge
Date: May 2026
Challenge: https://dmcommunity.org/challenge-apr-2026/

Framing the Review

The April 2026 challenge sits at an intersection that is increasingly relevant to anyone building production systems with AI: the point where probabilistic language models meet deterministic decision logic. The three required services (creatinine clearance, medication selection, and drug interaction checking) are textbook structured decisions. Finite inputs, explicit conditions, deterministic outputs. What the challenge asks, implicitly, is whether an agentic system can be trusted to orchestrate those decisions correctly, consistently, and transparently.

This review evaluates the five submissions from the DMCommunity April Challenge, through a combined lens. The primary frame is decision management discipline: rules externalization, standards alignment, model transparency, hit policy rigor, and explain-ability. Part of each entry assessment is a secondary frame drawn from agentic architecture: how sequencing is enforced, how observable the system's behavior is, and where trust is placed between the LLM and the deterministic layer. The two frames are not in tension; they are asking the same question from different directions.

The Architecture Spectrum

Entry	Rules Format	Standards Alignment	LLM Role	Sequencing Enforcement
Aletyx	DMN 1.6 / FEEL	Native DMN	Author, not runtime executor	DMN engine (structural)
Trisotech	DMN + BPMN/CMMN	DMN + OMG standards	Conversational interface	Soft (advocates BPMN)
OpenRules	Decision glossaries + tables	Proprietary (DMN-adjacent)	Orchestrator and interface	Soft (schema-inferred)
Novus-Forge	Python rule agents	None formal	Governed executor	Manifest (structural)
Gene Weng	Python + SKILL.md	None formal	Orchestrator	Soft (skill-inferred)

A pattern worth noting before going entry by entry: the two approaches with the strongest decision management standards alignment (Aletyx and Trisotech) are also the two that most clearly constrain or eliminate LLM involvement in the execution path, that correlation is not coincidental.

Entry Reviews

OpenRules: Dr. Jacob Feldman

Decision management perspective:

OpenRules demonstrates the most mature deployment posture of the five entries. The decision glossary pattern, where each service publishes its own schema and natural-language description, is a practical operationalization of what the DM community has long advocated: decision services as first-class, independently deployable artifacts. The services are self-describing, which means a new LLM or orchestration layer can consume them without any custom integration work.

The hit policy choices reflect proper decision modeling discipline. The `DetermineMedication` table uses multi-hit, correctly acknowledging that more than one condition may apply to a given patient. The `DetermineDosing` table uses single-hit, correctly enforcing that only one dosing outcome is valid. The `BigDecisionTable` for drug interactions treats data as a lookup structure rather than nested rule conditions, which is architecturally sound: reference data that changes independently of business logic should not be embedded in rule tables.

Agentic dimension:

The sequencing dependency between services is the main agentic concern. The challenge logic requires creatinine clearance to be calculated before dosing can be determined. In a DMN model, this dependency would be explicit as a directed edge in the DRD. Here, it is implicit in the service descriptions and inferred by the LLM at runtime. In the happy path, a well-prompted LLM gets this right consistently. In edge cases, unusual inputs, or a partially degraded service environment, there is no structural enforcement to fall back on. For regulated clinical deployment, undeclared sequencing dependencies are a reliability risk worth addressing explicitly, either with a formal process model or with explicit dependency declarations in the service schemas.

The AWS Lambda deployment is worth acknowledging: these are real callable endpoints, not simulated tools. The LLM is invoking production-grade services and receiving structured JSON. That is a complete and deployable agentic loop.

Aletyx: Alex Porcelli

Decision management perspective:

This is the submission that I feel will generate the most discussion among decision management practitioners, and for good reason. The claim that the AI Assistant generated the entire DMN model without manual intervention deserves careful attention, because it represents a genuinely new capability: *AI as decision model author*.

The DRD structure the AI produced appears proper and dependencies flow upward cleanly: `Creatinine Level`, `Age`, `Weight`, and `Is Penicillin Allergic` are input nodes. `Creatinine Clearance` and `Medication` are intermediate decisions. `Dosing` depends on both. `Conflicting Medications` stands as a reference structure. `Drug Interaction Warnings` iterates over the conflict table using a FEEL list comprehension. `Treatment Recommendation` assembles all outputs via a Context node. This is how a trained DMN modeler would structure it.

The hit policy choices are worth examining individually:

- **Medication table (F-hit):** First applicable rule wins. Correct: penicillin allergy takes priority over age-based selection, and F-hit enforces that ordering explicitly.
- **Dosing table (F-hit):** Again correct. Renal impairment overrides age-band defaults. Priority ordering is the right semantics here.
- **Conflicting Medications (C7 / collect-count):** This is the most interesting choice. C7 collects all matching rows and returns the count. Practically, this means the model returns the number of interactions found, not the list of interactions. A DM expert would likely prefer C- (collect all, no operator) to return the full interaction objects for clinical presentation. Whether the AI chose C7 intentionally or as a shortcut is an open question worth exploring with Aletyx.

The FEEL expression for drug interactions, `for conflict in Conflicting Medications return if list contains(split(Active Medications, ","), conflict.Medication2) then {...} else null`, correctly iterates over the interaction table and filters by active medications. This is idiomatic FEEL. That an AI produced it without additional prompting is notable.

DM critique: The creatinine clearance FEEL expression (`decimal(((140 − Age) ∗ Weight) / (Creatinine Level ∗ 72), 2)`) omits the female sex correction factor (multiply by 0.85 for female patients). This gap is shared by several entries in this challenge, including my own, but it is worth flagging here because a DMN tool with a validation layer should ideally catch formula completeness issues. A possible question for the community:

"Should AI-generated DMN models be automatically validated against a clinical knowledge base before being published?"

The bigger question this raises: If AI can correctly author DMN models, the implication for DM tooling is significant. Aletyx is essentially demonstrating that their AI can read a decision specification and produce a standards-compliant model. The next frontier is whether it can validate that model against domain constraints, clinical, regulatory, or otherwise.

Agentic dimension:

This entry largely sidesteps the runtime agentic problem, and that is precisely what makes it interesting from a governance standpoint. By using the LLM to author a deterministic DMN model rather than to orchestrate at runtime, Aletyx produces a system with zero LLM involvement at decision time. No hallucination risk at execution, no sequencing nondeterminism, no LLM observability overhead. The DRD encodes the execution dependency graph structurally: `Dosing` cannot execute before `Creatinine Clearance` because the DMN engine enforces that dependency. This is a fundamentally different answer to the agentic governance problem from the other entries.

Trisotech: Dr. John Svirbely, MD

Decision management perspective:

Trisotech's submission is the most aligned with where the decision management discipline says it wants to go, even while acknowledging it does not get all the way there in this entry.

The core architectural principle is cleanly stated: LLMs handle interaction, DMN models own the decisions, BPMN/CMMN govern execution sequences. This is the pattern that OMG standards are designed to support. The submission correctly identifies that prompt-based instruction is not a substitute for model-governed execution: *"these controls belong in models, not prompts. Period."* This is a statement of DM orthodoxy that the community should take seriously as it evaluates LLM-integrated architectures.

The use of decomposed intermediate decisions for traceability is sound DMN practice. Rather than collapsing medication selection and dosing into a single complex table, the model separates them, making each step independently auditable. A regulator or clinical reviewer can inspect the antibiotic selection table independently of the dose adjustment table. That separation is not just aesthetically clean; it is a governance asset.

Agentic dimension:

This entry makes the most important agentic architecture observation of the five submissions: instruction-based governance is not sufficient. Telling an LLM to "always call creatinine clearance before dosing" is a soft constraint. A BPMN sequence flow that requires a successful CCr result before the dosing service is reachable is a hard constraint. The difference between soft and hard enforcement matters enormously in production clinical systems, and Trisotech correctly names it.

The MCP-based service exposure creates a clean trust boundary between the LLM layer and the deterministic decision layer. The LLM cannot inspect the decision logic; it can only call the service and receive a result. This is good agentic hygiene: the LLM's probabilistic reasoning is confined to the interaction layer, where uncertainty is acceptable, and excluded from the decision layer, where it is not.

Combined critique: The submission stops short of building the full governed architecture it advocates for. It demonstrates the LLM-plus-DMN half of the pattern without the BPMN/CMMN enforcement half. That is a reasonable scope for a community challenge entry, but it leaves practitioners without a complete reference implementation. Trisotech is well-positioned to close this gap in a future submission.

Novus-Forge: Jeremiah Connelly (Self-Review)

Decision management perspective:

From a DM standards perspective, Novus-Forge is the weakest entry. The business rules live in Python: readable, testable, and correct, but outside any formal decision notation. There is no DRD, no hit policy declaration, no FEEL. A decision management toolset cannot introspect, validate, or simulate these rules without running the code. If this system were deployed in a regulated clinical environment, the rules would need to be re-expressed in a notation that compliance tooling can process.

The `rules_export_agent` tool call partially addresses this gap by generating markdown and HTML representations of the rule logic for business analyst review. It is a practical workaround, but not a substitute for formally modeled decisions. The rules are human-readable in the export, but they are not machine-readable in any standards-compliant sense.

Agentic dimension:

This is where Novus-Forge makes its most distinctive contribution. The manifest-driven pipeline trades LLM autonomy for execution reliability. The sequence (creatinine, then medication, then interactions, then rules export) is structural, not inferred. The LLM cannot accidentally skip a step or call them out of order, regardless of how the physician describes the patient. This is the BPMN-like control Trisotech correctly advocates for, implemented through a different mechanism.

The PII interception layer demonstrates a key principle in agentic system design: trust boundaries must be enforced at ingress, not at the LLM layer. A physician submitting a raw clinical note with a Social Security Number or Patient Medical ID embedded does not require the physician to understand the governance rules. The system parks the transaction before any clinical agent sees the data. The LLM never had the opportunity to process or echo the PHI because the governance layer ran first.

The per-step telemetry with cryptographic hashing addresses the observability gap that most agentic architectures leave open. Every pipeline step produces a structured, tamper-evident record: what input was received, what service was called, the LLM "thought process or CoT", what was returned, and how long it took. This is full agent observability persisted immutably.

Combined view: Novus-Forge is strong on the agentic governance dimensions (sequencing, observability, PHI handling) and weak on the decision management standards dimensions (no DMN, no formal notation). A mature production architecture would combine the governance infrastructure of Novus-Forge with the standards-compliant decision modeling of Aletyx or Trisotech.

Gene Weng: Claude Skills and Python

Decision management perspective:

This entry shares the same DM standards limitation as Novus-Forge: deterministic Python with no formal notation. The `SKILL.md` files are an interesting analog to decision glossaries. They document intent in natural language alongside deterministic code, but they are not machine-readable by DM tooling and cannot be introspected or simulated by a decision management platform.

What Gene Weng's submission demonstrates well is rule isolation. Each service is independently testable via pytest, which addresses one of the core DM concerns about LLM-integrated systems: that the decision logic can be validated separately from the conversational layer. The test suite covers boundary conditions (CCr threshold at exactly 50, age boundary at 15 and 60) in the way that a trained decision analyst specifying acceptance criteria would. The `SKILL.md` pattern, where intent is documented alongside code, also serves a function similar to a decision table narrative: it is reviewable by non-technical stakeholders even if it is not standards-compliant.

Agentic dimension:

The skills architecture is the most minimal agentic implementation of the five entries: Claude's own built-in agent loop handles sequencing and result assembly, no additional SDK or orchestration harness needed, which makes this genuinely lean. The skill-loading mechanism gives Claude awareness of what tools are available and how to invoke them, and each skill returns structured output that the orchestrating skill assembles into a coherent therapy plan.

The sequencing concern present in OpenRules and Trisotech applies here as well. Claude infers the correct order (CCr before dosing) from the skill descriptions, which works reliably on the happy path but is soft-enforced. For a production system, this would need to be hardened. The approach also has no PII detection layer: whatever the physician includes in the patient scenario text passes through Claude before reaching the Python scripts.

The pytest coverage earns specific recognition from an agentic architecture standpoint. In any system where an LLM invokes tools, those tools must be independently testable. A bug in the creatinine agent should be catchable in a unit test, not discovered in a patient case. Gene Weng builds this discipline in from the start.

Cross-Cutting Observations

Standards Adoption and Its Agentic Implication

Two of five entries use DMN formally (Aletyx, Trisotech). Three implement equivalent logic in code. The standards-aligned entries are also the ones where the LLM's role in the execution path is most constrained.

This is not a coincidence: when decision logic lives in a formal DMN model, the model itself becomes the execution engine and the LLM's job is reduced to interpretation and interface. When decision logic lives in code or skill files, the LLM must reason about sequencing and interpretation at runtime, which increases the surface area for variability.

The Sequencing Enforcement Gap

Across all five entries, the dependency that CCr must be calculated before dosing is determined is handled differently: by the DMN engine (Aletyx), by BPMN advocacy without implementation (Trisotech), by LLM inference from service descriptions (OpenRules, Gene Weng), or by a fixed manifest (Novus-Forge). None of the entries except Aletyx and Novus-Forge enforce this dependency with a structural artifact that governance tooling can independently verify. This is the gap Trisotech correctly names, and it is the right target for a possible future challenge: *a complete demonstration of LLM-plus-process-model orchestration where sequencing is enforced by the model, not by the LLM.*

Hit Policy Rigor as a Governance Signal

The hit policies used in the DMN entries carry real governance, meaning this deserves explicit attention. F-hit (first match wins) is appropriate for decisions where priority ordering determines the outcome. C7 (collect-count) quantifies matching rows but discards the content. C- (collect all) returns every matching rule, which is valuable for audit trails and explainability. Aletyx's use of C7 for the conflicting medications table means the output tells a clinician how many interactions exist, not what they are. A C- policy would return the full interaction objects for display and logging. Future challenge entries should be asked to justify hit policy choices explicitly, because those choices encode governance assumptions.

Explainability Requires Both Standards and Observability

DMN's strongest argument in the DM community is that decision tables can be reviewed by business analysts without reading code. Aletyx and Trisotech deliver on this; their models are inspectable by any decision professional with DMN literacy. OpenRules delivers it via auto-generated service descriptions. Novus-Forge partially delivers via the rules-export-agent. Gene Weng delivers via SKILL.md.

The agentic complement to model explain-ability is runtime observability: can a reviewer trace exactly what the agent did, in what sequence, with what inputs and outputs? Novus-Forge builds this into the governance layer. The others largely leave the agent's reasoning implicit. For regulated environments, both dimensions matter: the decision logic must be explainable before execution and the agent's behavior must be auditable after it.

Conclusions

The April 2026 challenge demonstrates that five different teams, working independently, converged on the same core foundational principle: deterministic decision logic should not live inside the LLM. Every submission keeps the rules in an external artifact, which is a meaningful consensus and a clear marker of where the community's thinking has settled.

What the challenge also reveals is that consensus on rules externalization does not resolve the governance questions that arise once those external services are wired to an LLM.

- How is sequencing enforced?
- How is the agent's behavior recorded?
- Where does PHI handling live?
- How does a non-technical reviewer inspect the logic?
  The entries all answer these questions differently, and the variation is instructive.

From a combined decision management and agentic governance perspective, the most complete individual contribution is Trisotech's architectural argument: *externalize decisions into DMN, enforce sequencing with BPMN, constrain the LLM to the conversational layer.* The most complete individual implementation of governed execution is Novus-Forge, which builds structural enforcement and audit infrastructure at the cost of standards alignment. Aletyx's submission points to a third path that neither of these fully explores: *use AI to produce the governed artifact, then remove the AI from the runtime path entirely*.

A production architecture that combines Trisotech's standards discipline, Novus-Forge's governance infrastructure, and Aletyx's AI-assisted authoring capability would be a compelling answer to the challenge the April 2026 round posed. That is a reasonable target for a future challenge or a follow-on submission.

Jeremiah Connelly | Novus-Forge | contact@novus-forge.com