AGENTIC OPERATIONS • DECISION SCIENCE

How Do Agents Make Decisions Under Uncertainty?

The decision happens outside the model, in a decision layer.

Pieter van Schalkwyk, XMPro • Technical white paper

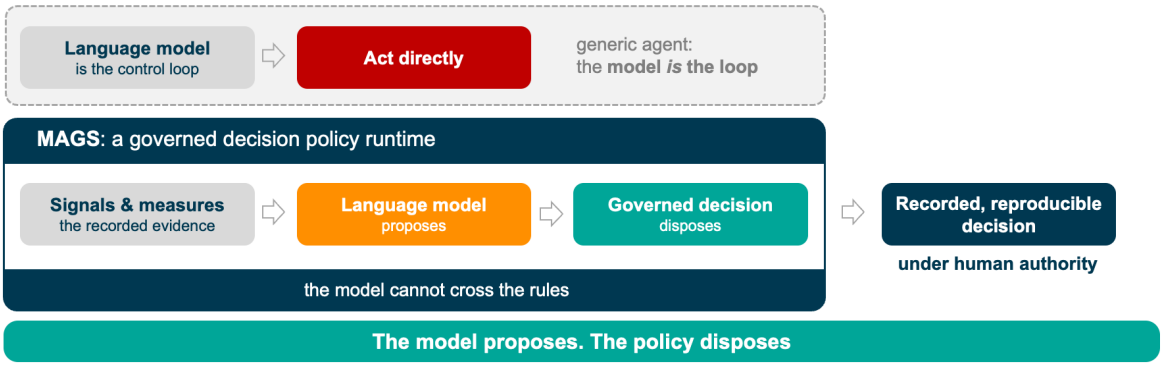
Ask how an AI agent should decide under uncertainty and you get two very different answers, and the gap between them is the whole subject of this paper.

The common answer is to make the model reason harder. Give it a longer chain of thought, a larger parameter count, more context, and trust that a better reasoner will weigh the situation and land on a good call. That answer is fine for a creator, an agent that drafts something for a person to read and check. It is the wrong answer for an operator, an agent that makes a decision on consequential state, a quality disposition, a setpoint, a maintenance call, where the decision has effect the moment it is issued and has to survive an audit afterward.

The distinction matters because uncertainty is exactly where a creator architecture fails an operator. A model asked to decide under uncertainty produces a confident-sounding answer that varies between runs and cannot be reproduced, and "the model weighed it up" is not a sentence that survives a regulator, an insurer, or the engineer who signed the work order. So the operator does not ask the model to resolve the uncertainty. It represents the uncertainty explicitly, outside the model, in a decision layer, and decides there. This paper walks through how that decision layer works, on the architecture we are building on Warren Powell's Sequential Decision Analytics.

The decision happens outside the model

Start with the structural point, because everything else follows from it. In a generic agent the language model is the control loop: it reads the situation, reasons, and acts, and whatever guardrails exist are instructions in its prompt that it may or may not follow. In an operator the language model is one bounded input to a governed decision policy. The model proposes, and the policy disposes.



The model proposes; the policy disposes. A generic agent is a language model acting directly in the loop. The operator makes the model one bounded input to a governed policy, producing a recorded, reproducible decision under human authority.

A decision cycle runs like this. The platform records the evidence, the signals and the measures, as structured state. The language model reads that state and produces an interpretation, a classification, or a draft plan, and that output is written into the record as one more input, tagged as what it is. A deterministic policy then maps the recorded state to a decision: it scores the candidate actions against an objective and a utility function, checks them against the constraints, requires corroboration from independent sources, and emits the decision with the evidence chain that produced it. The model did creator-like work inside the cycle. The decision was made by the layer around it.

Powell's framing makes this precise. What the system executes is a policy, written $X^{\pi}(S_t \mid \theta)$, a function from a defined state S_t to a decision, parameterized by tunable values θ. A function is deterministic by construction, so the same state and the same parameters produce the same decision. The question of whether an agent can decide reproducibly under uncertainty becomes a narrower and answerable one: which parts of the system sit inside that function, and which only feed it.

The five-element model the operator runs on

Powell's universal model writes any sequential decision problem with five elements, and the operator is built directly on them.

- The **state** S_t is the information necessary and sufficient to compute the objective, evaluate the constraints, and step the system forward.
- The **decision** x_t is what the operator controls, the disposition or the setpoint.
- The **exogenous information** W_{t+1} is everything that arrives after the decision and before the next one, which is where uncertainty enters. It carries the measured outcome of the last action and, as the next section describes, the language model's own output.
- The **transition function** $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ describes how the recorded state evolves once the decision is made and the world responds.
- The **objective** is the element that does the real work, because the optimization is over policies rather than over single decisions:

$$\max_{\pi} \quad \mathbb{E}\left\{\sum_{t} C\big(S_t, X^{\pi}(S_t)\big)\right\}$$

The operator does not search for the best decision in the moment. It runs a policy chosen to maximize the contribution C accumulated over time, which is the cumulative-reward form that fits an operation running continuously rather than answering once. That is the difference between a system tuned to look good on the next call and one tuned to operate well across a shift, a feed change, and a price swing.

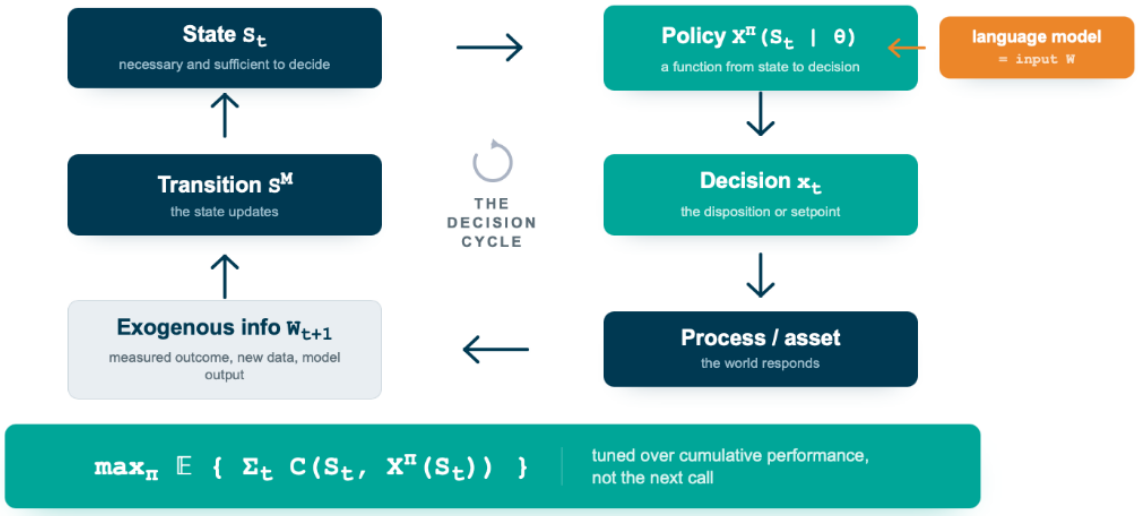


Figure 1. A decision cycle in the universal model: a policy maps the recorded state to a decision, the world responds with new information, and the state updates. The system is tuned over the cumulative objective, not the next call.

Represent the uncertainty, do not hide it

The first move is to stop treating the model's output as the truth. Powell's taxonomy of uncertainty includes a category that fits a sampled language model exactly, algorithmic instability, where identical runs produce different outputs. The operator does not deny that. It models it. The language model's recommendation enters the state as exogenous information, the same slot a noisy sensor reading occupies, and from there everything downstream is a deterministic function of the record. A noisy input is allowed. What is not allowed is letting that noisy input be the controller.

This one reframing is what lets the rest of the machinery work, because once the model is a sensor rather than a judge, the problem becomes the ordinary engineering problem of deciding well from imperfect signals, which decision science has studied for decades.

A map of the uncertainty

"Uncertainty" is not one thing. Powell catalogs about a dozen distinct sources of it, and an industrial operator meets most of them in a single decision cycle. The value of naming them is that each maps to a specific mechanism rather than to a vaguer instruction for the model to be careful. The table below lists the sources that bite hardest in industrial operations, a concrete example of each, and where the decision layer handles it.

Source	Industrial example	Mechanism in the decision layer
Algorithmic instability	the model reads the same event differently on two runs	enters as exogenous information W, recorded, never the controller
Observational error	a drifting analyzer, a miscalibrated probe	corroboration gate and confidence scoring down-weight a lone noisy read
Model or parameter	does a purge cycle really lift yield by the quoted figure?	belief state B_t with Bayesian updating; the prior is replaced by the measured effect
Experimental (sensing has a cost)	when to run an expensive assay	UCB or interval-estimation sensing, a cost function approximation
Inferential or diagnostic	is the impurity driven by feed composition or by reflux?	epistemic separation across specialist agents, plus objective scoring
Prognostic (forecast error)	demand, price spread, time to failure	deterministic direct lookahead over the transition model
Exogenous or environmental	feed composition or ambient conditions shift	the transition function S^M and a cumulative-reward objective tuned across conditions
Control or implementation	a valve moves less than commanded	constraints and the actuation boundary; intent routes to a human where no channel exists

Figure 2. Eight sources of uncertainty converge into the deterministic decision layer, and the language model is one input among them, never the place the others are resolved.

One line carries the weight of the table. Every one of these is modeled in the decision layer, the deterministic back end of the agent harness, and not in the language model. The model is one row on the list, algorithmic instability, and it is never the thing that resolves the others. This is the operator thesis applied to uncertainty: a creator architecture asks the model to absorb all of this inside its reasoning, where it cannot be inspected or reproduced, while an operator represents each source explicitly in that back end, where it can be. How to model each source in depth is the subject of Powell's books, and the point here is only where the work is done, not in the model.

Two of these mechanisms, the belief state and the sensing policy, are the parts of this architecture currently being implemented; the rest are in place today.

The belief state: deciding when you do not know what your actions do

The deepest source of uncertainty in an operator is not the sensor noise of the moment. It is that the agent does not actually know how much a given action will move a given outcome. A manual says a purge cycle improves yield by some figure, but the real effect on this unit, with this feed, drifts, and a static estimate is a guess wearing a number.

Powell decomposes the state S_t into three parts, a physical or resource component R_t, an information component I_t, and a belief component B_t, the last being probability distributions over quantities that cannot be observed directly. The uncertainty about what an action does lives in B_t, and the belief state is what makes the operator capable of learning, because decisions update beliefs and beliefs drive decisions.

So the agent does not store a number. It stores a belief. For each action a and outcome k the belief holds a mean effect $\bar{\mu}_{a,k}$ and a precision $\beta_{a,k} = 1/\bar{\sigma}^2$, seeded from the initial estimate. On executing a and measuring the outcome W with observation precision β^W, the conjugate Gaussian update is:

$$\bar{\mu} \leftarrow \frac{\beta\,\bar{\mu} + \beta^{W} W}{\beta + \beta^{W}}, \qquad \beta \leftarrow \beta + \beta^{W}$$

The mean is pulled toward the measured effect in proportion to how precise the new observation is against the current belief, and the precision β only grows, so the agent's confidence in what it has learned is itself a tracked quantity. After enough cycles it stops believing the manual and starts believing the plant.

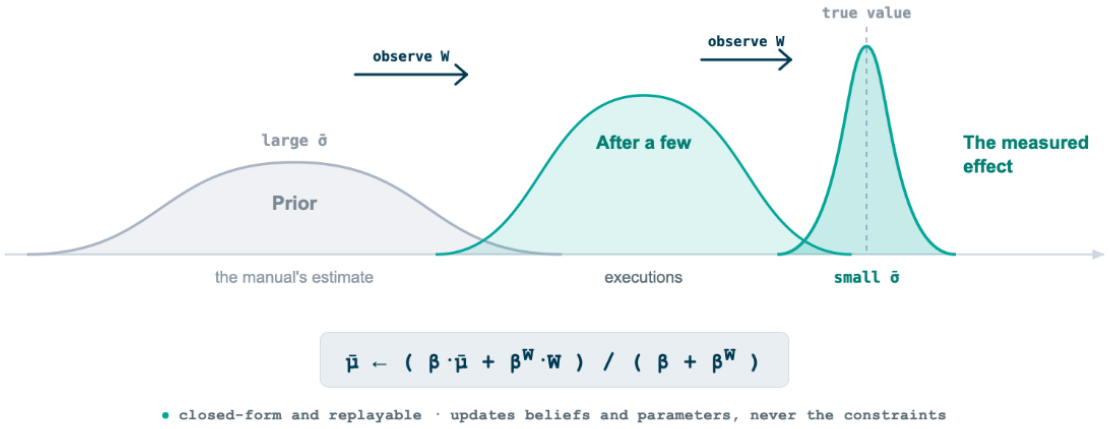


Figure 3. Each measured outcome pulls the belief distribution toward the real effect and narrows it, by closed-form arithmetic that is fully replayable and that updates beliefs and parameters, never the constraints.

Worked example: the belief about a purge cycle

The manual says the purge cycle lifts yield by 5.0 points, and the agent is unsure, so its prior is $\bar{\mu} = 5.0$ with $\bar{\sigma} = 2.0$, a precision $\beta = 1/2.0^2 = 0.25$. It runs the cycle and measures a 3.0-point improvement with process noise $\sigma_W = 1.0$, so $\beta^W = 1.0$. The update gives posterior precision $\beta = 0.25 + 1.0 = 1.25$ and posterior mean $\bar{\mu} = (0.25 \times 5.0 + 1.0 \times 3.0)/1.25 = 3.4$, with $\bar{\sigma} = 1/\sqrt{1.25} = 0.89$.
The belief moves from 5.0 ± 2.0 to 3.4 ± 0.89. Because the observation was more precise than the prior, the estimate jumps most of the way to the measured value, and the spread tightens. Feeding in two further measured outcomes of 3.2 and 2.9:

Step	Obs. W	β	$\bar{\mu}$	$\bar{\sigma}$
Prior	n/a	0.25	5.00	2.00
1	3.0	1.25	3.40	0.89
2	3.2	2.25	3.31	0.67
3	2.9	3.25	3.19	0.55

The agent starts on the manual's 5.0 and converges on the plant's measured effect of about 3.2, with confidence rising as the spread falls from 2.0 to 0.55. The same prior and

> the same recorded outcomes reproduce these numbers exactly.

Deciding when to look

Uncertainty also governs a decision most agents never make explicitly, namely when to spend a sensing action. Running a diagnostic, querying an expensive analyzer, or pulling a high-resolution reading each has a cost, so the agent has to decide which information is worth acquiring now. This is the exploration-versus-exploitation problem, and the operator uses the textbook answer directly, an interval-estimation policy that scores each sensing option as its estimated value plus a multiple of its uncertainty:

$$X^{\mathrm{UCB}}(S_t) \quad = \quad \arg\max_a \left(\bar{\mu}_a \; + \; \theta^{\mathrm{UCB}}\, \bar{\sigma}_a\right)$$

This is the upper-confidence-bound rule from the multi-armed bandit literature, and in Powell's taxonomy it is a cost function approximation, a deterministic optimization whose inputs are tuned to perform under uncertainty, with θ^{UCB} the single tunable that sets how much the operator values resolving what it does not yet know. A source the agent is unsure about has a large $\bar{\sigma}$, so it gets investigated rather than ignored, an information channel that went quiet last week is not permanently neglected, and the agent's curiosity becomes a governed, tunable quantity rather than a model's whim.

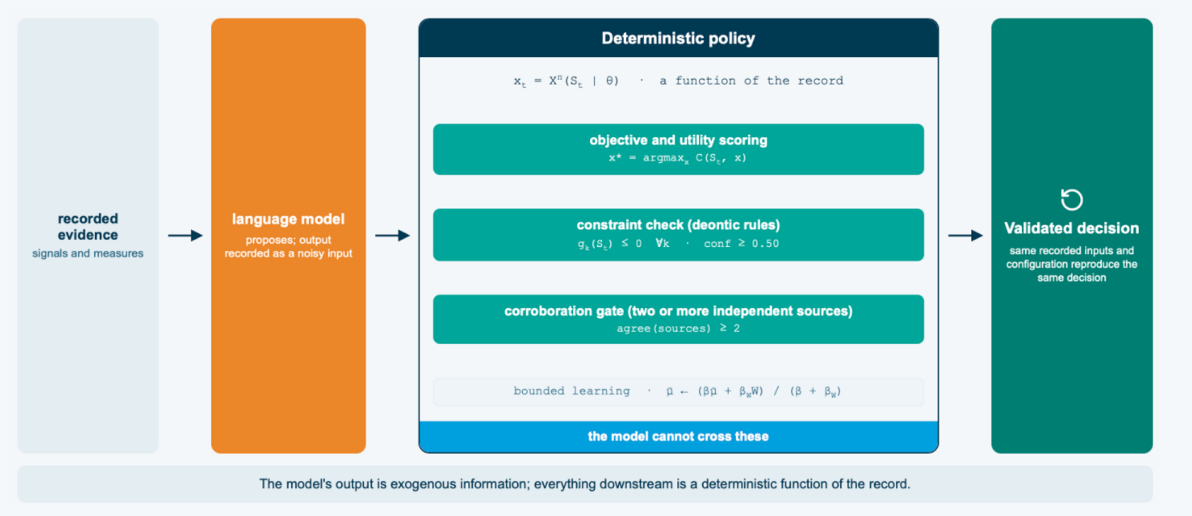
Corroboration as uncertainty reduction

A single line of evidence under uncertainty is a guess, so the operator requires more than one. Detection, context, and consequence are assigned to separate agents with their own context and, where it helps, their own models, and an orchestrator requires at least two independent sources to agree before a consequential decision is issued. Because the detecting agent and the reasoning agent do not share one chain of thought, their agreement is genuine corroboration rather than a single model nodding at itself. Independence is the whole point, since two correlated opinions do not reduce uncertainty and two independent ones do.

The part uncertainty never touches

Everything above adapts as the agent learns. The constraints do not. The deontic rules, the safety envelopes, and the physical limits define the feasible region the policy decides within, and they are evaluated deterministically and cannot be crossed. This is the line that makes uncertainty tolerable in a safety-critical setting: the agent is uncertain about how well an action will work, never about whether it is permitted, because the first is a belief it updates and the second is a constraint it cannot. A control engineer recognizes the shape of this immediately, because it is how a well-designed controller already behaves.

Why the decision is still reproducible



Inside the operator. Recorded evidence feeds the language model, whose output is a noisy input to a deterministic policy that scores objectives and utility, checks constraints, and requires corroboration before issuing a validated decision. Everything downstream of the model is a deterministic function of the record.

Put the pieces together and the apparent paradox resolves. The agent decides under genuine uncertainty, and the decision still reproduces exactly. What makes both true at once is where the uncertainty lives. It lives in the state, as belief distributions, recorded confidence, and the model's noisy proposal, all of it captured in the record, while the control flow that turns that state into a decision is a deterministic function. Given the same recorded inputs and the same configuration, the same decision comes out, and it arrives with the evidence chain that produced it. Uncertainty is represented in the data and determinism is preserved in the policy, and those two facts stop being in tension the moment the model sits on the inside.

The picture has a familiar human analog. Put two experienced operators in a control room, give them the same readings, and they will reason about the situation differently, in their own words and their own mental shortcuts. A well-run operation does not depend on their reasoning matching. It depends on the procedure, the SOPs, the operating limits, and the objective the unit is run to, which take the same recorded situation to the same action regardless of whose hands are on the panel. Their private reasoning is the stochastic part, the governed decision is the deterministic part, and the operation is engineered so the second does not inherit the variance of the first. An operator agent holds itself to that standard. Two reasoning passes over the same evidence may differ in wording, and the disposition still converges, because it is fixed by the constraints and the objective over the recorded state rather than by the reasoning that proposed it.

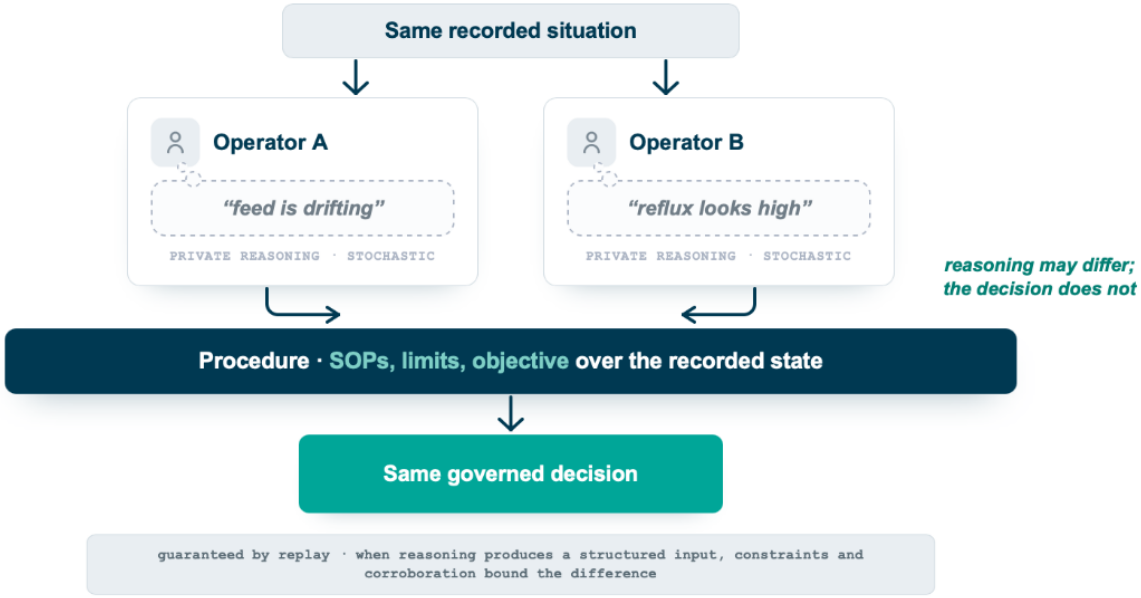


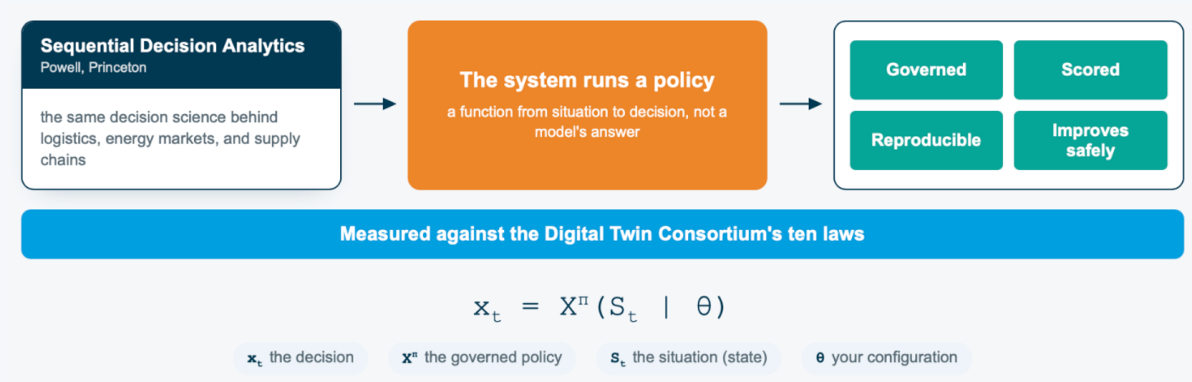
Figure 4. Two operators reason differently about the same readings, yet the procedure and the objective take them to the same governed decision. An operator agent holds to that standard.

One precision keeps the analogy honest. The convergence is guaranteed when the decision rests on the structured evidence and the rules, which is the replay claim: the same recorded inputs and the same configuration reproduce the same decision, exactly. The case to handle with care is when the reasoning itself produces a structured input, a classification or an estimate that then feeds the policy, because two passes can hand the policy different inputs. There the constraints, the cross-stream corroboration, and the belief state bound how far that difference can travel, so convergence is a designed property rather than an accident, and the unconditional guarantee remains replay.

Which policies the operator uses

Powell's second universal result is that every policy for a sequential decision problem falls into one of four classes, or a hybrid of them, and naming them shows exactly where the operator sits.

- **Policy function approximations (PFAs)** map state to decision analytically, with no embedded optimization. The operator's constraint and threshold rules are PFAs.
- **Cost function approximations (CFAs)** solve a parameterized deterministic optimization tuned to perform under uncertainty. The objective-and-utility scoring that selects the disposition, and the upper-confidence-bound sensing rule above, are CFAs.
- **Value function approximations (VFAs)** estimate the downstream value of a state and decide from it, which is the home of most reinforcement learning.
- **Direct lookahead approximations (DLAs)** optimize over a model of the future at each decision, with the deterministic case corresponding to model predictive control and classical planning.



The decision-science anchor. The system runs a policy, a function from situation to decision rather than a model's answer, grounded in Sequential Decision Analytics and measured against the Digital Twin Consortium's ten laws.

The operator is a hybrid of PFAs, CFAs, and deterministic DLAs, which is the corner of the design space Powell found is by far the most common in working systems. It deliberately avoids VFAs and deep reinforcement learning, which dominate the literature but are unstable and data-hungry and demand exploratory episodes an industrial process cannot give them. The uncertainty is carried by belief states and tuned deterministic policies rather than by a learned value function, and that is a choice about what can be validated and operated, not a shortfall.

A reference implementation: XMPro MAGS

Everything above is deliberately vendor-neutral, because Sequential Decision Analytics is mathematics any conformant platform can build on. It is also concrete, because we build one. XMPro's Multi-Agent Generative System occupies exactly the corner of the design space described here, a hybrid of policy function approximations, cost function approximations, and deterministic direct lookahead, with a language model bounded as exogenous information inside a governed policy.

What runs today maps onto the mechanisms in this paper almost one to one. Each agent observes, reflects, plans, and acts on the ORPA cognitive cycle, with significance and confidence scored from defined factors rather than asserted by the model. The separation between synthesizing and deciding is visible in the prompts themselves. The Observe stage instructs the model to observe and analyze only, not to make recommendations or plan actions, and the Reflect stage states that reflection is not an action but internal reasoning about how well the agent is fulfilling its role. Whether reflection or planning then fires is a significance-threshold computation rather than a model judgment, so the model's cognitive work stays synthesis while the governed decisions are made by the layer around it. Policy constraints are expressed as explicit deontic rules, obligatory, forbidden, and permitted, and evaluated deterministically as the feasible region the agents cannot cross. Objective and utility functions select among candidate actions, with lexicographic ordering so a safety objective is consulted before any throughput objective. A team of specialist agents with a consensus gate supplies the independent corroboration. Every cycle emits a structured decision record, with observation and reflection decisions represented in the W3C PROV-O provenance vocabulary, so the audit trail is a property of the design. These are in production now, and the reproducibility argument of this paper rests on them.

Two capabilities from the earlier sections are being added as we extend the implementation onto the full SDA foundation, and they are in active development rather than shipped. The first is the

belief state: for every action-to-measure pair, a Gaussian distribution updated by the conjugate rule after each execution, so the agent learns the real effect of its actions from recorded outcomes instead of trusting a static estimate. The second is the sensing policy: the interval-estimation, upper-confidence-bound rule for choosing which sensing tool to run and when. Both live inside the policy parameters, with no write path to the constraint set, so the learning is bounded by construction and the safety envelope does not move as the agent improves. The fuller account of the determinism and bounded-learning arguments is in the Governed Decision paper, and the cognitive architecture is developed in the companion ORPA paper.

Less creativity, more operation

This is why Sequential Decision Analytics is the right foundation rather than a technique we borrowed. Powell showed that the many fields studying decisions under uncertainty, dynamic programming, stochastic optimization, optimal control, reinforcement learning, and bandits, are solving one problem with one structure, and the policies that result, as the four classes above lay out, are analytic rules and tuned deterministic optimizations rather than the creative reasoning of a large model. The intelligence that handles the uncertainty lives in the governance, the belief state, the corroboration, the scoring, and the constraints, which is why the model underneath can be right-sized rather than maximal.

An operator deciding under uncertainty is therefore not a more creative model. It is a capable model placed inside a structure that represents what it does not know, learns the parts it is allowed to learn, refuses to cross the parts it is not, and records enough that the decision can be replayed. That structure is what turns a decision under uncertainty into something you can validate, and validation, not eloquence, is the operator's job.

> **Status note**
>
> The belief-state Bayesian updating and the upper-confidence-bound sensing described above are in active development as part of the platform's Sequential Decision Analytics foundation. The decision layer, constraints, objective and utility scoring, multi-agent corroboration, and decision provenance, on which the reproducibility argument rests, are in production today.

Acknowledgement. This paper builds directly on the Sequential Decision Analytics framework developed by Professor Warren B. Powell at Princeton. The universal model, the four policy classes, and the treatment of belief states and learning are his, and the architecture described here applies that work to industrial multi-agent systems. His Sequential Decision Analytics resource is at wbpowell328.github.io/sda-website.

Further reading. For the formal treatment of the determinism and bounded-learning arguments, see the companion paper *Governed Decision-Making for Industrial Multi-Agent Systems: A Sequential Decision Analytics Foundation for Trustworthy and Validatable Autonomy* (van Schalkwyk, Green, Kontsos, Filippou, and Dannhauser, 2026). For the cognitive architecture, see *Neuroscience-Inspired Cognitive Architecture for Multi-Agent Systems* (van Schalkwyk and Green, 2025).