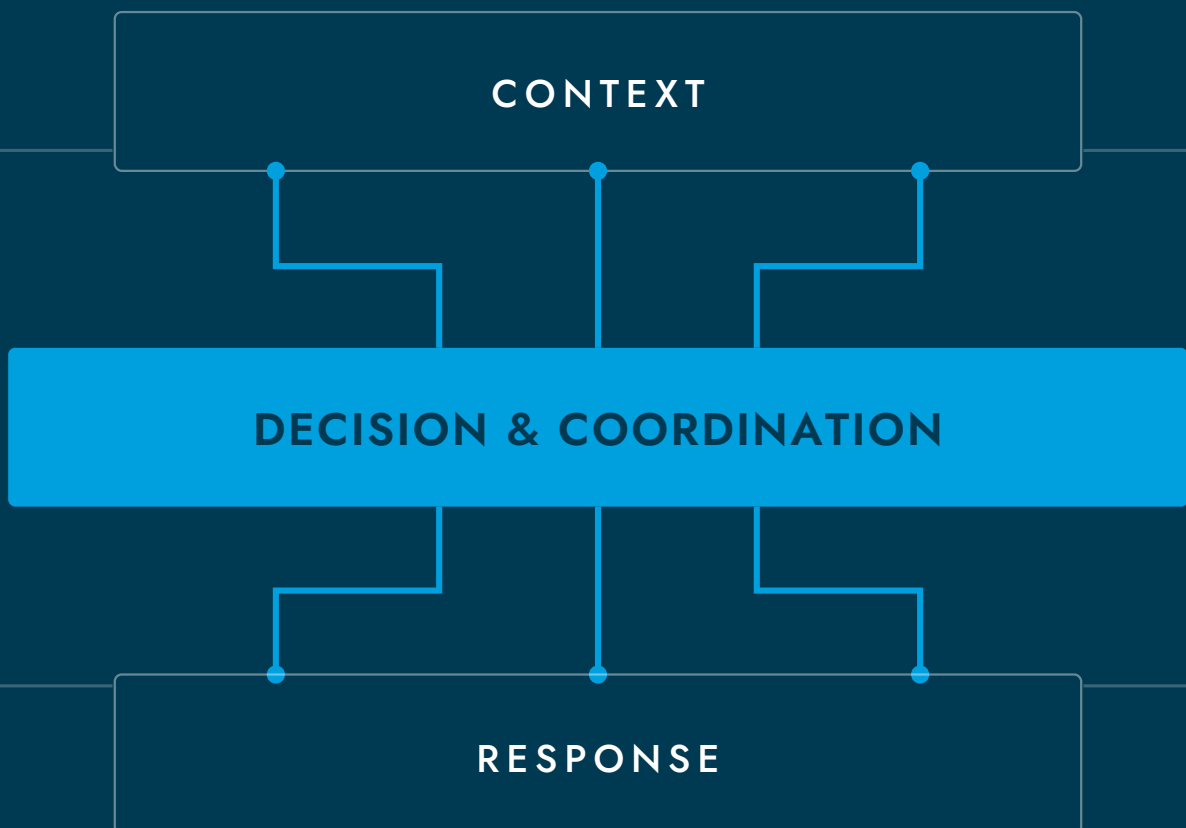


Software-Defined Operations

Change How Operations Work





Software-Defined Operations

Change How Operations Work

Pieter van Schalkwyk

XMPro · First edition

September 2026

Copyright © 2026 XMPro. All rights reserved.

Software-Defined Operations: Change How Operations Work

Pieter van Schalkwyk

First edition, September 2026. Production version 3.3.1, October 6, 2026.

Published by XMPro. xmpro.com

About this book

This book explains Software-Defined Operations as an architectural pattern for maintaining operational decision and coordination logic in software. It is written for operations leaders, engineering and systems teams, and business sponsors assessing what that pattern could mean for their organization.

The transfer station, candidate plans, and eligibility results are illustrative teaching assumptions. They establish no operating limit, hydraulic result, or measured performance improvement. The XMPro mapping describes proposed implementation responsibilities, with sources and scope recorded in Appendix A.

AI assistance and accountability

This book was prepared with the assistance of artificial intelligence tools. Pieter van Schalkwyk is the author and accountable document owner.

Contents

1	Why operational change becomes a systems problem	1
2	What becomes software-defined	5
3	Run a decision, then change its definition	11
4	What changes for the enterprise	16
5	Recognize the pattern and choose where to start	20
A	Map the architecture to XMPro	23
B	Working aids for a first application	31
	Glossary	37

How to read this book

The five chapters follow one argument: understand the operational problem, identify what becomes software-defined, follow a decision and a change to its definition, assess the enterprise implications, and choose a useful starting point.

Appendix A maps the architecture to XMPro. Appendix B contains working aids for a first application. The glossary explains the terms used in the argument and points back to the chapters where they are developed.

Why operational change becomes a systems problem

One of three pumps at a drinking-water transfer station shows signs of deteriorating condition. Maintenance wants an inspection window. Operations must maintain supply to the service reservoir while the pump is unavailable. The remaining pumps have different operating characteristics, and their availability matters to the plan. Energy prices and expected demand affect when the station should pump.

The condition alert identifies a problem. The response depends on a decision that involves several systems and people: **how should the station adjust its pumping plan when equipment condition, demand, energy prices, or operating requirements change?**

This illustrative station will recur throughout the book. It gives us a decision we can follow, change, and examine at a second site. The plans and their results are teaching assumptions; an actual station would require its own engineering assessments, operating limits, and acceptance evidence.

The decision extends beyond the application

The control system supplies reservoir measurements and equipment status. A condition-monitoring application reports the deteriorating pump. The maintenance system holds work plans and availability. Forecasts describe expected demand, while tariff information supports the energy calculation. Procedures and experienced staff establish the requirements under which the station may operate.

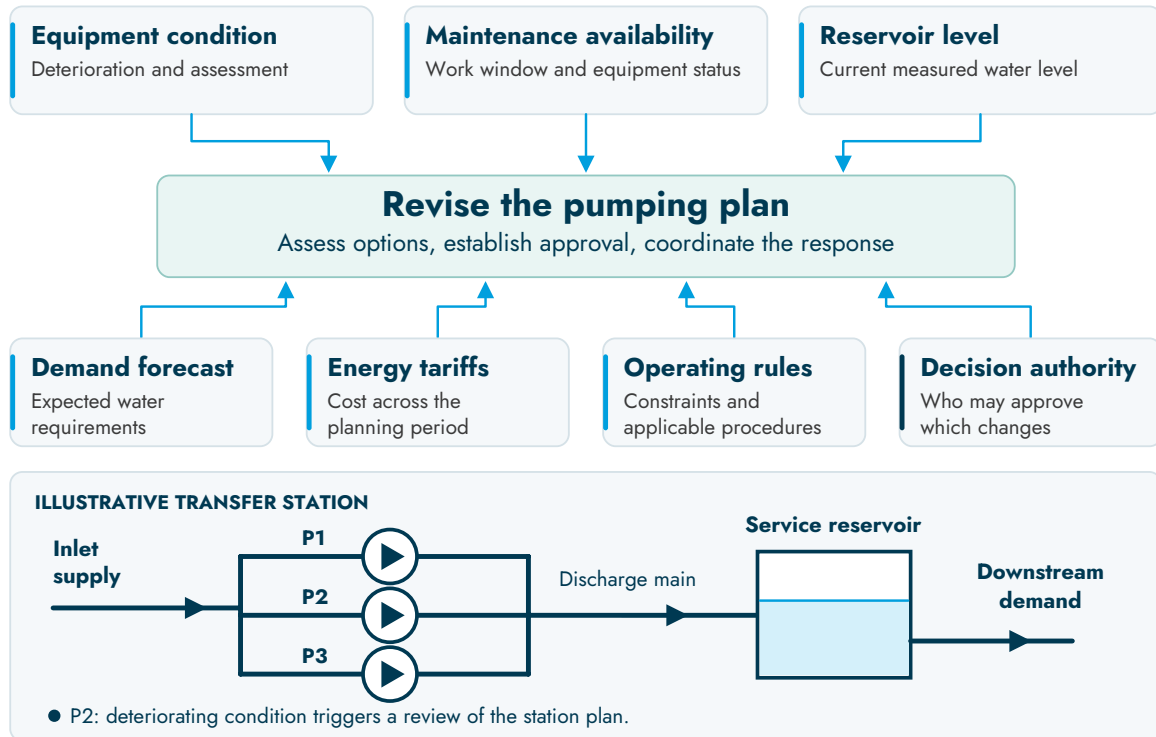
Each contribution has a purpose. A maintenance planner may find a convenient inspection window before knowing whether the resulting pumping plan meets the reserve requirement. An energy calculation may favor a period when a pump will be unavailable. An operator may have a current reservoir measurement but an outdated demand forecast.

The team has to evaluate these conditions together. It must identify acceptable plans, compare them, establish who may approve a response, and coordinate the necessary requests. A useful recommendation must also reach the person who can act on it in time.

The operational decision therefore has a scope of its own. That scope includes information, rules, responsibilities, and actions distributed across the systems involved. Figure 1.1 makes the dependencies visible before we consider how to organize them.

ONE DECISION, MANY DEPENDENCIES

What must the station bring together before changing its plan?



© 2026 XMPro. All rights reserved.

Figure 1.1. One decision, many dependencies. Cyan arrows represent contributions to the decision; navy piping represents the physical station. The diagram implies no hydraulic sizing or operating result.

Find the logic governing the response

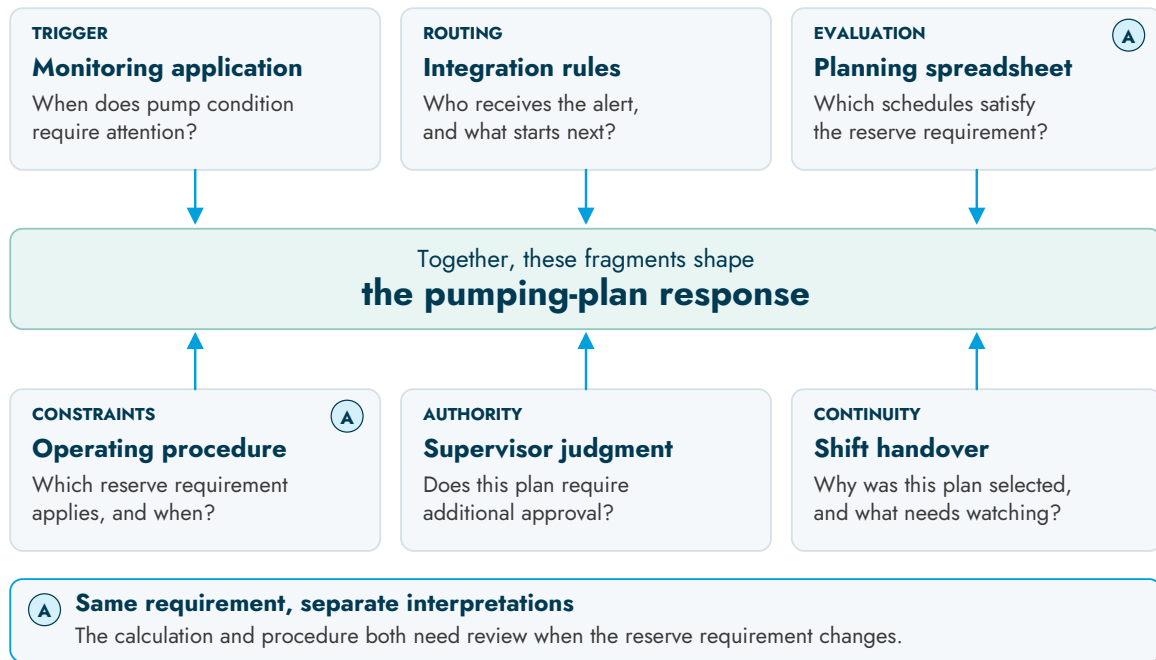
Follow the station's response from the first condition alert to an agreed inspection window. The monitoring application uses a threshold to raise the issue, and an integration rule sends a notification. The spreadsheet used to compare pumping schedules depends on a reserve requirement stated in a procedure. A supervisor knows when the plan needs additional review. At shift handover, the next operator receives an explanation of the choice.

These elements contain operational decision and coordination logic: what matters, which options qualify, how to choose among them, who must participate, and what can happen next. The station already has a decision process, even if nobody maintains it as a coherent whole.

Experienced people can coordinate it well. The difficulty arises when they repeatedly have to reconstruct its dependencies or when the knowledge needed to interpret one part is held by someone who is unavailable. Improving an individual application may help its own task while leaving that coordination work largely intact.

WHERE THE LOGIC LIVES TODAY

A conceptual map of one decision assembled across tools and people.



© 2026 XMPro. All rights reserved.

Figure 1.2. Where the logic lives today. The calculation and procedure express the same requirement in different places. A revision creates work to locate, reconcile, and test the affected uses.

A revised requirement exposes the architecture

Suppose the station adopts a revised minimum reservoir reserve through its engineering and operating approval process. The planning calculation must apply it. The procedure must explain where and when it applies. Pending recommendations based on the previous requirement need reassessment. The approval route may need attention if different plans now qualify.

The work extends beyond entering a new value. The team must establish which expression of the requirement governs each decision and whether all affected uses reflect the approved interpretation. A corrected document provides no assurance that a calculation has changed. A corrected calculation does not tell an approver which revision produced a recommendation.

The interfaces between systems matter, but so does the location of the governing logic. If that logic is difficult to find, each revision begins with investigation. At several sites, the same investigation may happen repeatedly, with different interpretations and different local fixes.

Give the response an explicit definition

Software-Defined Operations addresses this architectural problem. Its central proposition is that operational decision and coordination logic can be represented and maintained explicitly in software, with defined connections to the systems it uses. The organization can then identify a proposed change, examine its dependencies, and test the resulting response.

That proposition creates a practical obligation. The team must be able to point to the rules, configurations, calculations, models, and permissions that govern a decision. It must know who owns them and how they relate to the current operation. Calling a collection of applications an architecture does not establish those relationships.

Consider two changes at the station. A new demand forecast may require the existing decision to run again. A new reserve requirement changes the definition governing future evaluations. **Changing an input and changing the decision definition are different kinds of work.** The first asks what the operation should do now; the second changes how it determines its response.

Where existing measurement and scheduling interfaces still meet the revised need, they can remain in use. Some changes will require new information, different calculations, or altered equipment arrangements. The architecture should make that scope easier to establish, including the work that cannot be confined to software configuration.

The business purpose is to improve how the organization coordinates and adapts its operation. A maintained definition can support more consistent decisions, make specialist knowledge usable by others, and give another site an assessable starting point. Those possibilities depend on the implementation. Any reduction in effort or improvement in operating performance needs evidence.

The station does not have to delegate consequential decisions to an agent to benefit from this approach. People can retain approval and direct the response. Local control continues to operate the pumps and enforce protective functions. The architectural question is how the wider pumping-plan decision is defined and maintained around those responsibilities.

Chapter 2 identifies what becomes software-defined. Chapter 3 then follows one decision and changes its governing requirement. The test throughout is concrete: can the team show where the response is defined, what a revision affects, and why the revised behavior is fit to use?

What becomes software-defined

At the transfer station, revising the reserve requirement should not automatically require rebuilding the connection that supplies reservoir measurements. Changing the person who approves a pumping plan should not automatically require a new energy calculation. These responsibilities depend on one another, but a useful architecture makes their relationships explicit enough to establish which changes can be made separately.

In this book, we use the following definition:

Software-defined operations is an architectural pattern that decouples operational decision and coordination logic from the underlying applications, automation systems, and physical infrastructure, enabling that logic to be configured, changed, and governed through software.

This is the book's proposed architectural pattern. It describes a way to organize operational capabilities, with implementation choices and limits to examine in each application.

Three responsibilities in one architecture

Figure 2.1 places the decision logic between the systems that supply operational information and the people and systems that carry out a response. Read the diagram as three logical responsibilities.

Existing systems describe the operation: measurements, asset relationships, work history, procedures, and other context. The decision and coordination logic establishes how to use that information: what to achieve, which requirements apply, how to evaluate alternatives, and how to arrange a permitted response. People, workflows, and control systems perform their assigned actions and supply evidence of what happened.

The separation is logical. The components may run in several applications or environments. A service that calculates reserve, a workflow that requests approval, and an interface that validates a scheduling request can all contribute to one decision definition. Their placement matters for implementation; their identifiable responsibilities matter for understanding the pattern.

At the physical station, local control retains equipment operation, interlocks that block actions under defined conditions, and other protective functions. The wider decision concerns the pumping plan and its coordination with maintenance. A change to equipment control or protection remains a separate engineering responsibility.

Software-Defined Operations

Separate changeable decision logic from the systems it uses.

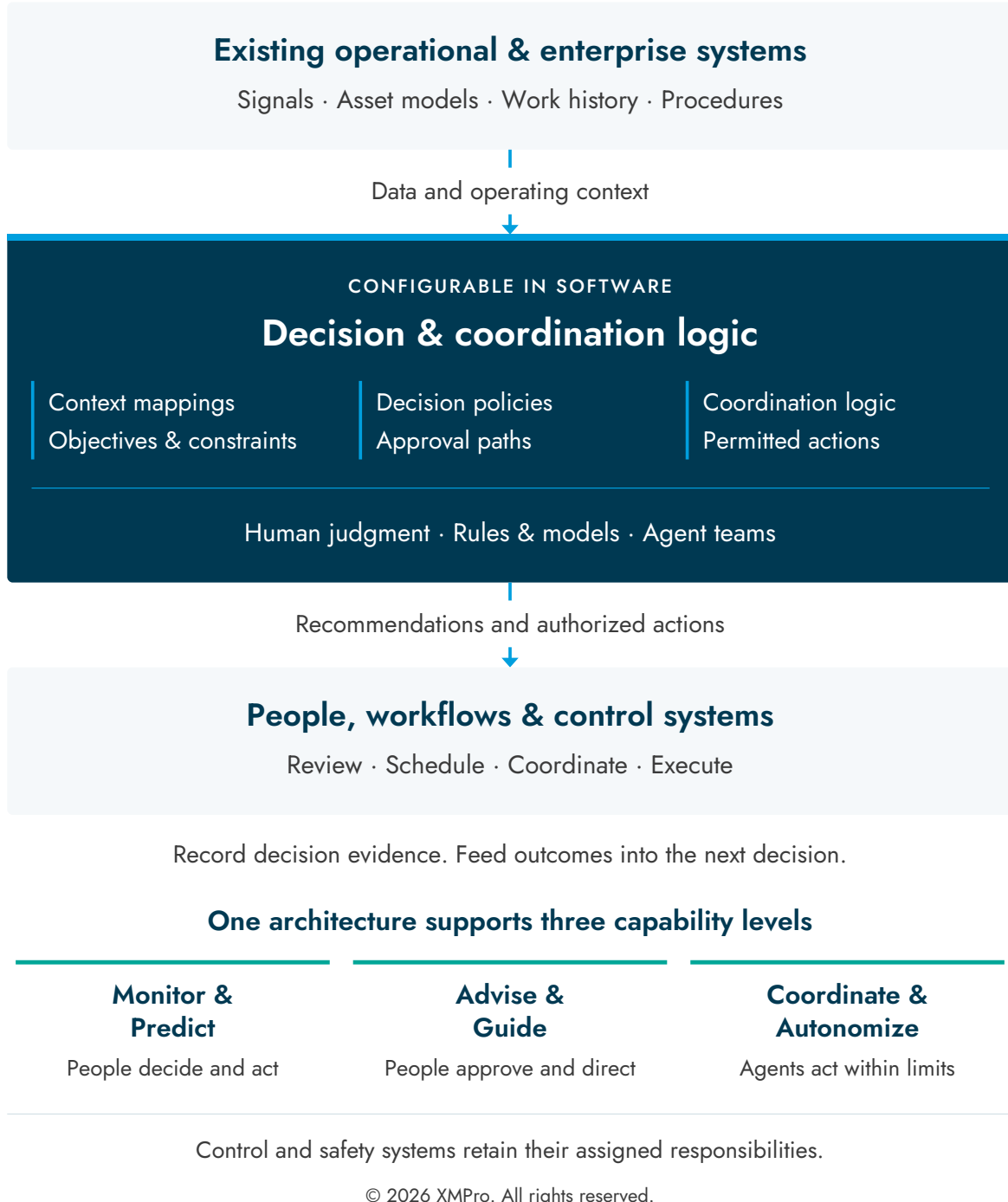


Figure 2.1. The Software-Defined Operations architecture. The central responsibility is the logic governing the operational response. The same architecture supports different levels of human and agent participation, with control and safety systems retaining their assigned responsibilities.

Identify the decision definition

The central region needs an inspectable form. A **decision definition** is the maintained description and software representation of what a decision seeks, requires, may request, and must record. At the station, its scope is revising the pumping plan when relevant conditions change.

Part of the definition	Station example	Where to identify its software representation
Objective and ranking	Maintain supply; compare energy cost among eligible plans.	The configured objectives and comparison policy.
Constraints	Apply the approved reserve and equipment requirements.	The applicable rule, parameters, and evaluation method.
Context requirements	Use the right reservoir measurement, forecast, and availability records.	Source mappings, units, time requirements, and quality checks.
Coordination	Align the inspection window, pumping plan, and maintenance request.	The handoff sequence and handling of incomplete or uncertain responses.
Approval policy	Route the proposal to the responsible operations reviewer.	The assigned role and conditions requiring review.
Permitted actions	Prepare a recommendation; initially leave submission to a person.	Capabilities exposed to the decision and permission checks at action interfaces.
Evidence requirements	Preserve inputs, alternatives, the active definition, approval, and outcome.	The record structure and identifiers connecting the contributing records.

A written procedure may supply the reserve instruction, but the team also needs to locate the calculation or rule that applies it. The procedure owner's approved interpretation and the active software representation must agree. Otherwise the definition remains incomplete at the point where it matters: the evaluation of a plan.

The definition may be distributed across configurations and services. Maintain the relationships among those parts, identify their revisions, and assign an accountable owner. A version label on one workflow is insufficient if the team cannot identify which reserve rule, forecast, and permission policy were used with it.

Establish what can change independently

An input interface must describe more than the format of a message. The decision needs to know which asset a value describes, its unit, when it was observed, and the period for which it is useful. An action interface must identify what a request means and how the caller will distinguish acceptance, rejection, and an uncertain response.

Suppose the station replaces its demand-forecast service. If the replacement supplies the agreed meaning, time coverage, quality, and failure behavior, the team can assess retaining the decision logic that consumes it. If it changes those assumptions, the affected evaluation needs revision too. The same reasoning applies when replacing a model or a receiving application.

Independence is therefore something to demonstrate. For a proposed change, identify the part of the definition being revised, the dependencies it uses, and the parts expected to remain stable. Run appropriate cases and inspect the result. A favorable diagram or an editable screen provides no evidence that these relationships are enforced in the implementation.

This also explains why the pattern can use familiar technology. Workflow engines can coordinate approvals; integration services can acquire information and dispatch requests; rules and validated calculations can evaluate known conditions. The architectural distinction concerns how the governing logic is organized across those components and what can be changed without unnecessary changes elsewhere.

An organization may already implement part of the pattern. It can examine an existing pumping-plan application against the same questions: where is the definition, which dependencies are explicit, who maintains it, and what evidence supports the claimed separation? A new product name is unnecessary for that assessment.

The software-defined idea has precedents in other domains. Networking separates network control from packet forwarding; software-defined automation separates control software from underlying hardware; software-defined manufacturing separates production applications from hardware and infrastructure. Their scopes differ, and they can overlap with operational coordination. They illustrate the value of naming the responsibility being separated and the interfaces it depends on. [1](#) [2](#) [3](#)

For this book, that responsibility is the definition of the operational response. The station's pumps, measurements, and maintenance application remain necessary. Software-defined operations concerns how the organization decides and coordinates their use as conditions and requirements change.

Choose how people and agents participate

The three capability levels in Figure 2.1 describe participation within the same architecture. Under **Monitor & Predict**, systems organize information and forecasts while people interpret them and decide how to respond. Under **Advise & Guide**, the

system evaluates options or develops a proposal for people to review and direct. Under **Coordinate & Autonomize**, agents may coordinate or execute particular actions within explicitly granted limits.

These choices can coexist. An application may prepare a recommendation, require human approval of the plan, and later receive permission to copy an approved window into a planning record. The evaluation and the action permissions remain identifiable parts of the definition. Granting a narrow administrative permission does not confer authority to operate equipment or isolate a pump for maintenance.

Artificial intelligence (AI) can contribute where the process needs to determine what information is missing, develop alternatives, or revise an inquiry after a new observation. At the station, an agent might request a refreshed forecast, ask whether a condition assessment permits a later inspection window, and explain the alternatives to the reviewer. Defined rules can still check freshness, and validated calculations can still evaluate reserve.

The architecture supplies a purpose, context, permitted capabilities, and a route to action for that reasoning. The choice to use an agent and the choice to authorize an action require separate decisions. A person can retain every consequential approval while the organization gains a clearer and more maintainable definition of the response.

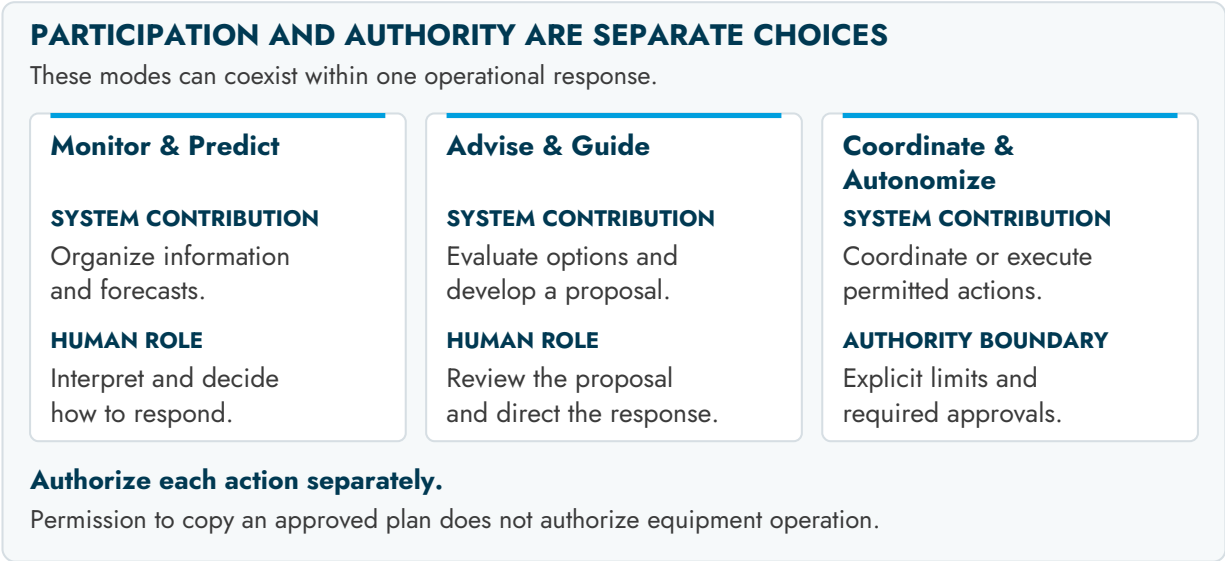


Figure 2.2. Participation modes can coexist. Using an agent and granting permission to act are separate decisions; each action remains subject to its defined limits and approvals.

Look for a maintained operational capability

The definition needs to remain useful after its first demonstration. Operations must be able to identify the version in use. A specialist must have a route to correct an interpretation. An interface owner must know which decisions depend on a changed source. These are the practical consequences of treating the decision as something the organization maintains.

Its boundaries also make limitations visible. If the proposed revision requires a measurement that does not exist, the team has new engineering work. If an application exposes no suitable action interface, that dependency must be resolved. The architecture does not make every response configurable; it provides a way to distinguish a change to the governing logic from the other work required to put that change into operation.

The next chapter tests this idea with the pumping plan. We will first use the definition to select an inspection window, then change the reserve requirement while keeping the input case fixed. Seeing both operations together makes the separation assessable.

Sources and scope

1. [Open Networking Foundation: SDN Definition](#). Separation of network control and forwarding.
2. [Schneider Electric: Software-defined automation](#). Separation of control software and underlying hardware.
3. [Bosch: Flexible production for the factory of the future](#). Separation of production applications and infrastructure.

These source descriptions were checked on September 18, 2026. The Software-Defined Operations definition and station design are the book's proposed synthesis; the cited organizations do not endorse them through these references.

Run a decision, then change its definition

Pump P2's condition has changed. Maintenance wants an inspection window, and operations needs a pumping plan that meets the station's requirements while accommodating that work. We will follow the decision under its current definition, then examine what happens when the approved reserve requirement changes.

The example is advisory. Software may assemble context, evaluate options, and prepare a proposal. People approve the pumping plan and submit the scheduling request. The eligibility results below are stipulated teaching assumptions, with no numerical reserve, hydraulic calculation, or plant performance implied.

Begin with the active definition

The decision owner identifies the definition in use: its objectives, applicable requirements, input sources, evaluation methods, review route, and permitted actions. The condition alert identifies P2 and the observation time. Repeated notifications about the same event should resolve to the same review, avoiding duplicate work requests.

The process then connects P2 to the station and service reservoir. It obtains the reservoir measurement, demand forecast, equipment availability, condition assessment, maintenance options, and tariff information. Those inputs need a common interpretation for the decision. A field marked "available" in a maintenance system may require clarification if a condition assessment imposes a restriction.

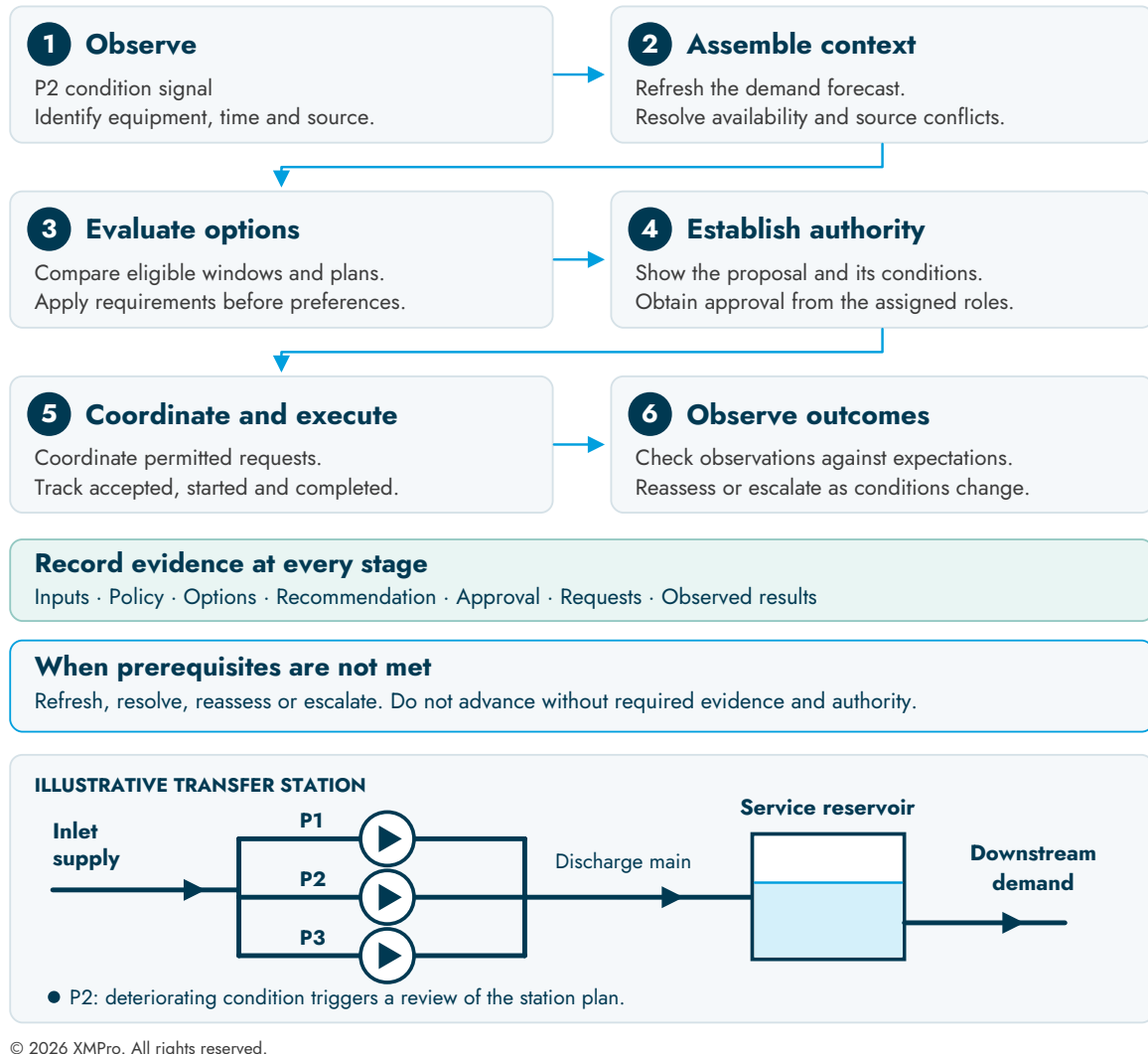
In this case, the forecast is outside the definition's freshness requirement. The process requests an update and holds the recommendation until the required evidence is available. If the update cannot be obtained, it follows the agreed escalation or fallback. A guessed forecast would change the basis of the decision without resolving the missing evidence.

An agent can help organize that inquiry. It can identify the missing information, request the refreshed forecast through an allowed capability, and ask the responsible specialist whether P2 can remain in service until a later inspection window. The agent's explanation does not establish the equipment conclusion; the specialist and applicable engineering assessment do.

This is the first use of the architecture. The definition identifies what evidence is required, how to obtain or resolve it, and when the decision must wait. Operations and maintenance can inspect the same basis for the subsequent comparison.

FOLLOW THE PUMPING-PLAN DECISION

One decision, six responsibilities, a continuing record of evidence.



© 2026 XMPro. All rights reserved.

Figure 3.1. Run the defined decision. The stages may involve several systems and people. A return to an earlier stage reflects new information or an unresolved prerequisite; it does not silently authorize a change to the governing requirements.

Make one requirement usable

The reserve rule illustrates the connection between operating knowledge and software. Consider the teaching instruction: "Evaluate each proposed pumping plan against the approved minimum reservoir reserve." Before applying it, engineering must establish the applicable value, units, planning period, calculation method, and treatment of uncertainty.

The implementation associates that reviewed meaning with the correct reservoir and station configuration. The evaluator compares the candidate plan's minimum predicted reserve with the applicable approved minimum. It returns pass or fail when

the evidence is sufficient, and unresolved when required evidence is inadequate. Retain the source revision, interpretation, asset scope, input references, and evaluation-rule revision with the result.

This is how the requirement becomes part of the active decision definition. A source document provides evidence for the rule, while the identified evaluator applies it to a candidate plan. The record connects those responsibilities so a reviewer can check both meaning and use.

Compare the windows and select a plan

For this teaching case, assume the refreshed forecast is suitable, the responsible specialists permit P2 to remain in service until either window, and both windows have the required staff and equipment. The evaluated plans meet the current reserve requirement. Their differences are qualitative:

Candidate	Result under the current requirement	Basis for the choice
Window A: the proposed inspection window	Eligible, with less reserve margin than B.	Lower expected energy cost under the assumed forecast and tariff.
Window B: a later permitted window	Eligible, with greater reserve margin.	A viable alternative, with higher expected energy cost in this case.

The agreed ranking policy favors lower expected energy cost among eligible plans. The process recommends **Window A**, retaining B and the reason for its lower ranking. If no candidate met the mandatory requirements, the result would be an escalation identifying the unmet condition. A preference for lower cost cannot make an ineligible plan acceptable.

The operations reviewer checks the comparison, condition assessment, maintenance availability, and applicable reserve requirement. In the illustrative sequence, the reviewer approves A with those conditions recorded. B remains an alternative requiring its own review if circumstances change.

Preserve what actually happened

The responsible person submits the maintenance scheduling request through the established interface. The receiving system acknowledges it and records the window. At this point, the planning handoff has succeeded. The inspection has not started, and the physical operating outcome remains pending.

The decision record distinguishes those states. It identifies the inputs and definition used, both candidates, the basis for choosing A, the approval, and the request acknowledgment. If acknowledgment were uncertain, the coordinator would inspect the request identifier and receiving state before attempting another submission.

A new availability restriction or demand forecast can reopen the affected evaluation. Operations must establish what remains valid and which subsequent actions should wait. Restoring a software record or withdrawing approval cannot undo an inspection that has already started; the physical response depends on the current equipment and work state.

We have now used the definition to reach an explicit choice. The example shows who did what and what remains unknown. It has not yet shown why the logic is software-defined. For that, we need to change the definition and examine the dependencies.

Revise the definition and retain the input case

The station adopts a revised minimum reserve through its engineering and operating approval process. To test the change, retain the input case used in the A/B comparison. For this teaching case, stipulate that A fails the revised minimum while B still qualifies. These are expected test results, not new engineering findings.

The accountable team identifies the applicable reserve rule in the decision definition and updates its approved configuration. The release record connects the new requirement, its interpretation, the evaluator that consumes it, and the cases used to test it. All other requirements and the ranking policy remain the same.

SAME INPUT CASE, CHANGED ELIGIBILITY

Stipulated test expectations. Other requirements and ranking policy remain unchanged.

CANDIDATE	PREVIOUS RESERVE	REVISED RESERVE
Window A	Eligible; lower expected energy cost. Recommended for human review.	Fails the revised reserve test. Excluded from selection.
Window B	Eligible; higher expected energy cost. Retained as an alternative.	Still eligible. Recommended for its own review.

Separate pending A proposal: reassess before dispatch
Hold the pending instance under the revised requirement. Preserve its earlier evaluation.
Prepare B for review if its evidence is still current; approval of A does not transfer.

Demonstrate what changed and what stayed stable
Changed: the applicable reserve requirement and resulting plan eligibility.
Stable where still fit: measurement and action interfaces, other constraints, and authority.
Evidence: the same retained inputs, both definition revisions, and the compared results.

These results test the decision definition; they do not establish hydraulic feasibility.
© 2026 XMPro. All rights reserved.

Figure 3.2. Change the definition and inspect the consequences. The outcomes are stipulated test expectations. Measurement and action interfaces remain stable only where they still meet the revised need. Recommendation of B requires its own review.

Show the architectural difference

Under the earlier requirement, both plans qualify and A ranks first. Under the revision, A is excluded and B becomes the recommendation. The change in eligibility is observable, but a different answer alone provides limited evidence about the architecture. A person could also edit an isolated spreadsheet and obtain a different answer.

The stronger demonstration identifies the governing rule and every dependent use within the scoped decision. The team shows how the reviewed requirement reaches the active evaluator and how the recommendation records the revision it used. It checks that the approval path and action permissions remain as intended. The implementation has to preserve these relationships, including when the definition spans several components.

In the scattered arrangement from Chapter 1, the team must discover and reconcile the rule wherever it is expressed. In the proposed arrangement, the rule, its ownership, and its dependent uses are maintained as part of the decision definition. That gives the change a defined scope to inspect and test. Whether it reduces total effort is a separate question to measure.

The reservoir measurement connection can remain unchanged if its identity, units, quality, and timing are still suitable. The scheduling interface can remain unchanged if it still supports the required request and status information. If the revised requirement needs a different measurement or calculation, that dependency becomes part of the change. Separation has practical limits that the demonstration should expose.

Introduce the revision without losing decision history

Consider a separate A proposal still awaiting approval when the revised requirement takes effect. Hold that instance for reassessment before dispatch. Applying the new definition excludes A and prepares B for review if its evidence remains current. Preserve the earlier proposal and the reason it was replaced. The approved A example remains a historical record of its own inputs and requirement.

Release the revision only within its accepted scope, with compatible configuration references and a way to observe or suspend its use. A faulty change may require people to resume coordination while the team investigates. An earlier reserve requirement may no longer be authorized, so recovery cannot simply mean restoring the last software version.

The distinction is now visible. A refreshed forecast allowed the original decision to proceed under its existing definition. A revised reserve requirement changed the definition and therefore changed which plan qualified. The organization can identify both events, explain their different consequences, and establish which interfaces still meet the need.

That is the architectural capability this example is intended to demonstrate. The next question concerns its value beyond one station decision: what becomes more manageable when the organization can maintain operational responses in this way?

What changes for the enterprise

The station example establishes an architectural possibility: the organization can identify the definition of a response, use it to make a decision, and revise it while retaining interfaces that still meet the need. The enterprise question is whether that capability makes important work easier to coordinate, change, or reuse.

The answer depends on the decision and its current arrangement. A simple local rule may already be easy to maintain. A pumping-plan response spread across several applications, specialist interpretations, and approval practices presents a different opportunity. The value comes from addressing a specific source of effort or limitation in that work.

Coordinate a response across responsibilities

The pumping plan brings maintenance timing, equipment condition, service requirements, and energy cost into one evaluation. Each function can still own its information and judgments. The maintained definition specifies how those contributions affect the response, including which requirements exclude a candidate and how acceptable candidates are compared.

That gives the participants something concrete to inspect when their priorities differ. Operations can explain the reserve requirement; maintenance can clarify the permitted inspection windows; the energy calculation can show a cost difference. A change in one contribution has an identifiable route into reassessment of the plan.

The potential benefit is less repeated reconstruction of the same decision. Software can assemble required information, identify stale inputs, and track handoffs. People can spend more attention resolving the gaps and tradeoffs that require judgment. The implementation should measure preparation effort and delays, including the work of correcting unsuitable recommendations.

Consistency follows from applying a maintained definition to current conditions. Different shifts may reach different results because equipment or demand has changed. The useful test is whether they applied the same approved requirements and can explain the reason for a different choice. Uniform answers to different situations would provide little assurance.

Specialist knowledge becomes more accessible when its interpretation, scope, and source are retained with the definition. Another qualified operator can inspect why the reserve rule applies and when specialist help is required. This can support continuity when an experienced person is unavailable, provided the record is kept current and its use has been tested with the people who need it.

Change the response while retaining suitable systems

The reserve revision illustrates a change to one constraint. Other changes may affect the ranking policy, choice of input source, approval route, coordination sequence, or permitted action. Naming those parts of the definition helps the team assess which dependencies require attention before estimating the scope and cost of the work.

The organization can also obtain more value from systems it already uses. A maintenance application can continue to own work records, while a historian, the system that stores plant measurements over time, continues to supply those measurements. The decision process uses the information and requests the actions required for a shared response. Maintaining a useful interface may be sufficient where rebuilding the underlying application would add little value to the decision.

This arrangement can make technology replacement more reviewable. A new forecast source or analytical model can be assessed against the decision's expected inputs, outputs, and limitations. Actual replacement still depends on access, configuration portability, compatible behavior, licensing, and migration work. The pattern identifies those dependencies; it does not promise independence from every supplier.

Authority can change at a narrower scope than the complete decision. In the station example, a person submits the approved scheduling request. A later application might receive permission to copy an already approved window into a designated planning record. That permission would specify the fields, target, current approval, and conditions under which submission is allowed. The pumping-plan evaluation and approval could remain unchanged.

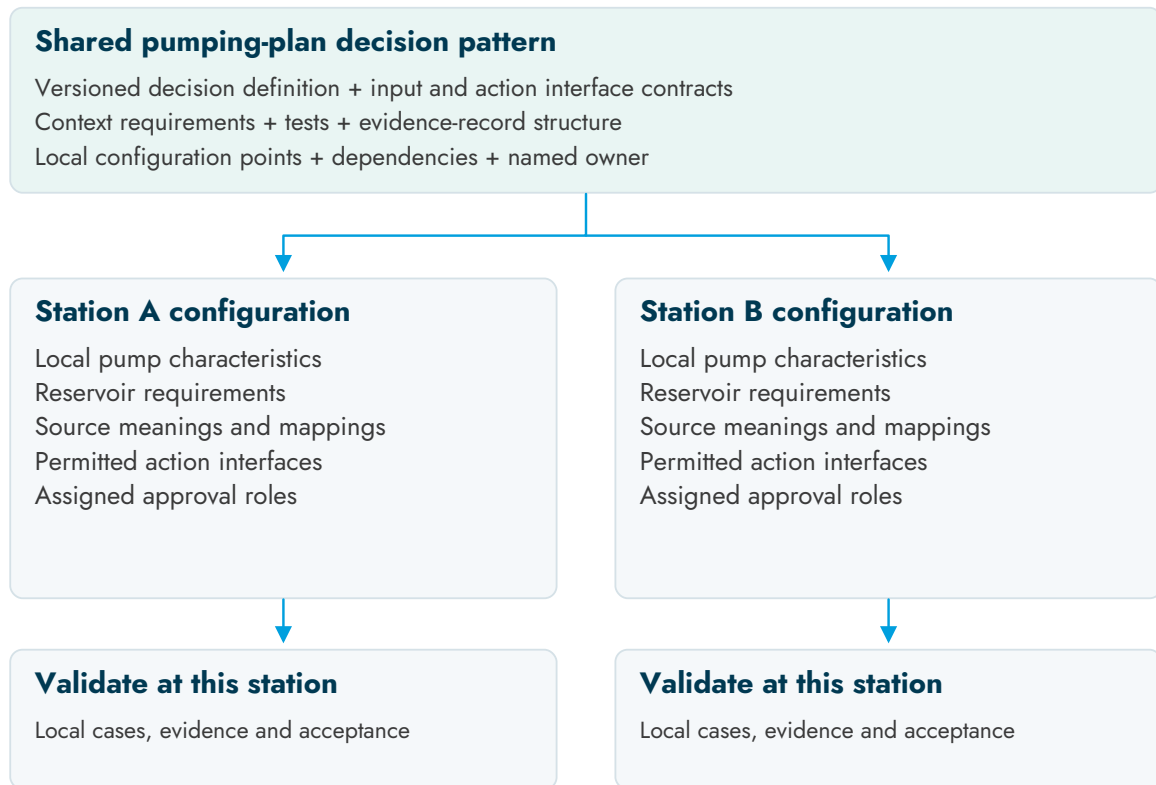
The team could then assess whether the delegated administrative action reduces routine handling while preserving the required review. Equipment control and maintenance isolation would remain outside that permission. The architecture permits this distinction because the action boundary is an identified part of the definition.

These changes also carry costs. Someone must maintain the definition, its interfaces, tests, and operating assumptions. A configurable response that nobody owns can accumulate uncertainty as readily as a collection of scripts. Include maintenance, training, and review effort when assessing whether the architectural separation has made change more economical.

Reuse the structure and establish the local fit

A second station may face a similar pumping-plan decision with different pumps, reservoir limits, source meanings, and approval roles. It can assess the first station's decision structure and test approach, then identify the local work required to use them.

SHARE THE PATTERN; ESTABLISH THE LOCAL FIT



Shared structure does not make equipment, limits or permissions interchangeable.

© 2026 XMPro. All rights reserved.

Figure 4.1. Reuse with explicit local differences. A shared definition supplies an assessable starting point. Each station must establish the meaning of its inputs, the suitability of its evaluation methods, and its own operating and action requirements.

The receiving team needs more than a copied workflow. It needs the decision's assumptions, the interfaces it expects, the evidence record, and tests with their intended results. Local configuration identifies the actual assets, requirements, sources, permitted actions, and responsible people. A reviewed local variation must remain distinguishable from the common definition.

Station B must establish its own cases. It can reuse the method of comparing a retained input case under two requirements, but cannot import the conclusion that Window A fails and Window B passes. Those results belong to the first station's teaching case. Technical acceptance includes the local calculations, source meanings, and conditions of use.

Maintain what is common and what is local

A common definition needs an owner who maintains its intended use, shared logic, interface expectations, and tests. Each site needs ownership of its local configuration and operational acceptance. These responsibilities can be assigned to existing roles, provided their boundaries are clear.

A correction to the shared logic may be relevant to several sites. Its introduction still requires an assessment of the affected local configurations. A local procedure change may affect only one station. Preserve the relationship between the common definition and each site's variation so that the team can establish which changes belong where.

Reuse can reduce repeated design and interpretation work, but only where enough of the decision is common. Compare equivalent scopes, including adaptation, validation, and continuing maintenance. Ask whether the receiving team can understand and maintain the definition without relying on the original developers for every explanation. That is a practical test of whether expertise has become usable beyond its first implementation.

Connect the capability to an operating result

The architecture creates ways to coordinate, adapt, and reuse. Those capabilities may support business benefits such as lower preparation effort, more consistent application of requirements, or shorter change lead time. Operating outcomes need an additional connection to the application: lower energy cost, improved equipment availability, or more consistent service must be measured in the operation.

At the station, a faster recommendation is useful only if it is timely, sufficiently complete, and fit for the approved scope. A lower-cost plan still has to meet service and equipment requirements. When demand, tariffs, or equipment condition change, the comparison must account for those influences before attributing an improvement to the application.

The decision record can also help explain disappointing results. A missing forecast, unsuitable calculation, incorrect requirement, and failed scheduling handoff call for different corrections. Linking the evidence gives the team a basis for investigating what happened and testing a proposed improvement. It does not establish a cause merely because two events appear together.

An operations leader takes ownership of a maintained response and its accepted use. Engineering and system owners need visibility of the dependencies that support it, while the business sponsor needs evidence that the capability justifies its cost and any wider adoption.

The organization should be able to explain that relationship before expanding the scope. A maintained decision that produces no useful operational benefit still needs revision, a different application, or a decision to stop. The next chapter turns the architectural idea into a small set of questions for making that choice.

Recognize the pattern and choose where to start

A useful first application is a recurring decision whose governing logic is important, distributed, and difficult to change consistently. The station's pumping plan is one example. Your candidate may be another response that requires information and action across several systems, with requirements that change over time.

Start by tracing a recent instance. Identify what initiated it, which information people used, how they evaluated the options, who approved the response, and what happened next. Then identify one plausible approved change to the governing logic. That exercise exposes whether the organization can locate the decision definition and understand its dependencies today.

Ask for evidence of the pattern

A proposal should make its architectural claims inspectable. Five questions provide a useful starting point:

Question	What to examine
Where is the decision defined?	The maintained rules, configurations, calculations, models, coordination, and permissions that govern the response.
What does it depend on?	Input meanings and quality, applicable requirements, evaluation methods, action interfaces, and their owners.
What can change separately?	An approved revision whose effects can be shown while other suitable parts remain stable.
What evidence connects definition and use?	The input case, active revisions, evaluated alternatives, approval, requested action, and subsequent status.
Who maintains and accepts it?	Accountable ownership of the intended response, technical dependencies, and operational use.

These questions apply to an existing application as well as a new proposal. A workflow or integration platform may implement the pattern. The assessment concerns the relationships it actually maintains and the changes it can demonstrate within the chosen scope.

An impressive recommendation on one example establishes little about maintainability. Ask to see the governing requirement in the implementation and a change to it. The demonstration should expose the dependencies and limitations that remain, including the work needed when an input or action interface cannot meet a new requirement.

Choose a decision the team can own

The first scope should be small enough to understand and important enough to improve. Identify the operation, the decision, its owner, the people affected, and the systems that contribute. At the transfer station, that means revising a pumping plan for one station and its reservoir. Coordination across a wider network would introduce additional constraints and responsibilities.

A candidate is more promising when teams repeatedly assemble the same context, reconcile several expressions of a rule, or repeat similar engineering work at different sites. Frequent changes in requirements can make the cost of scattered logic visible. The team still needs suitable sources, evaluation methods, interfaces, and an accountable owner willing to maintain the result.

Some decisions will not justify the added structure. A stable local rule already maintained and tested in an appropriate control system may need no separate operational decision layer. A poorly understood physical process may first require engineering investigation. An absent owner or unavailable source is unresolved implementation work, not something the architecture supplies by itself.

Choose the initial participation level to suit the decision. An advisory application can make context, alternatives, and the governing definition visible while people retain consequential approval. Further permissions can be examined later at the action level. Establish how people intervene and how the application responds when required evidence is unavailable.

Demonstrate a decision and one controlled change

The first demonstration should connect an identifiable definition to an actual evaluated case. Show the required information, the alternatives, the selected result, and the assigned review. The team should be able to explain which requirements governed the choice and how an unresolved input would affect it.

Then introduce one approved change using the same retained case. Compare the results, identify the revised configuration, and show which interfaces remained suitable. Account for pending proposals and the revision that should govern each one. This is the practical test developed in Chapter 3; an actual engagement must replace its teaching assumptions with locally validated cases.

Agree expected behavior before running the test. Include a stale input, a request outside the permitted scope, and an uncertain response from a receiving system. The evidence should show that the implementation handles these situations as intended, including when a person must take responsibility for the next step.

Establish whether the capability is useful

Select the benefit that justifies the initial application and establish a baseline. For the station, that might be the staff effort needed to prepare a reviewable plan, delays caused by missing information, or work required to introduce a revised requirement. Define the observation period and who will collect and interpret the evidence.

If the objective includes lower energy cost or better service, measure those outcomes as well. Keep the applicable operating requirements alongside efficiency measures, and account for demand, tariffs, equipment condition, and other changes. Include preparation, validation, training, and maintenance costs in the assessment.

Architectural acceptance and operating value answer different questions. The team may demonstrate a well-defined, changeable decision whose recommendations arrive too late to help. It may also observe an operating improvement that came largely from new equipment. The decision to continue or expand should use evidence appropriate to the claim being made.

The team that accepts the application needs a maintained definition, identified interfaces and dependencies, representative tests, connected evidence, and a way to introduce or suspend revisions. Appendix B supplies compact working aids for those responsibilities. Appendix A maps the architecture to XMPro for readers assessing that implementation.

Make the next change understandable

At the start of this book, an approved reserve change required the station to find and reconcile logic distributed across systems and people. The proposed architecture gave that response an identifiable definition. We used it to select a window, changed a requirement, and examined which result and dependencies changed with it.

Apply the same reasoning to a decision in your own operation. Locate its governing logic and identify the systems it depends on. Choose a change that matters, establish the intended behavior, and ask the team to demonstrate both the change and the limits of its scope.

Software-Defined Operations provides an architectural way to maintain how the organization responds. Its value depends on making that response understandable, changeable, and useful in the operation. When the next requirement changes, the enterprise should know where it belongs, who owns its use, and what evidence is needed before relying on the revised response.

Map the architecture to XMPro

At the transfer station, an XMPro implementation needs to carry the pumping-plan decision from observation to a reviewed response and then back to evidence of the outcome. Connecting the condition signal is one part of that work. The team also needs to locate the calculations, context, approval rules, execution interfaces, and records that make the decision inspectable and changeable.

This appendix maps those responsibilities to XMPro components. Product roles draw on the cited descriptions and documentation; the station configuration and its acceptance checks are proposed designs. The implementation brief must identify the selected releases, configured handoffs, and evidence needed to accept the application.

DataStreams connect the decision to the operation

DataStreams provide configured paths for acquiring signals, enriching and transforming data, performing calculations, invoking analytical models, routing events, and calling downstream interfaces. Teams configure those flows in **Data Stream Designer**. Listeners acquire observations, functions perform calculations, and Action Agents invoke configured actions. A DataStream can therefore contribute to both the observation and execution sides of the architecture. [3](#)

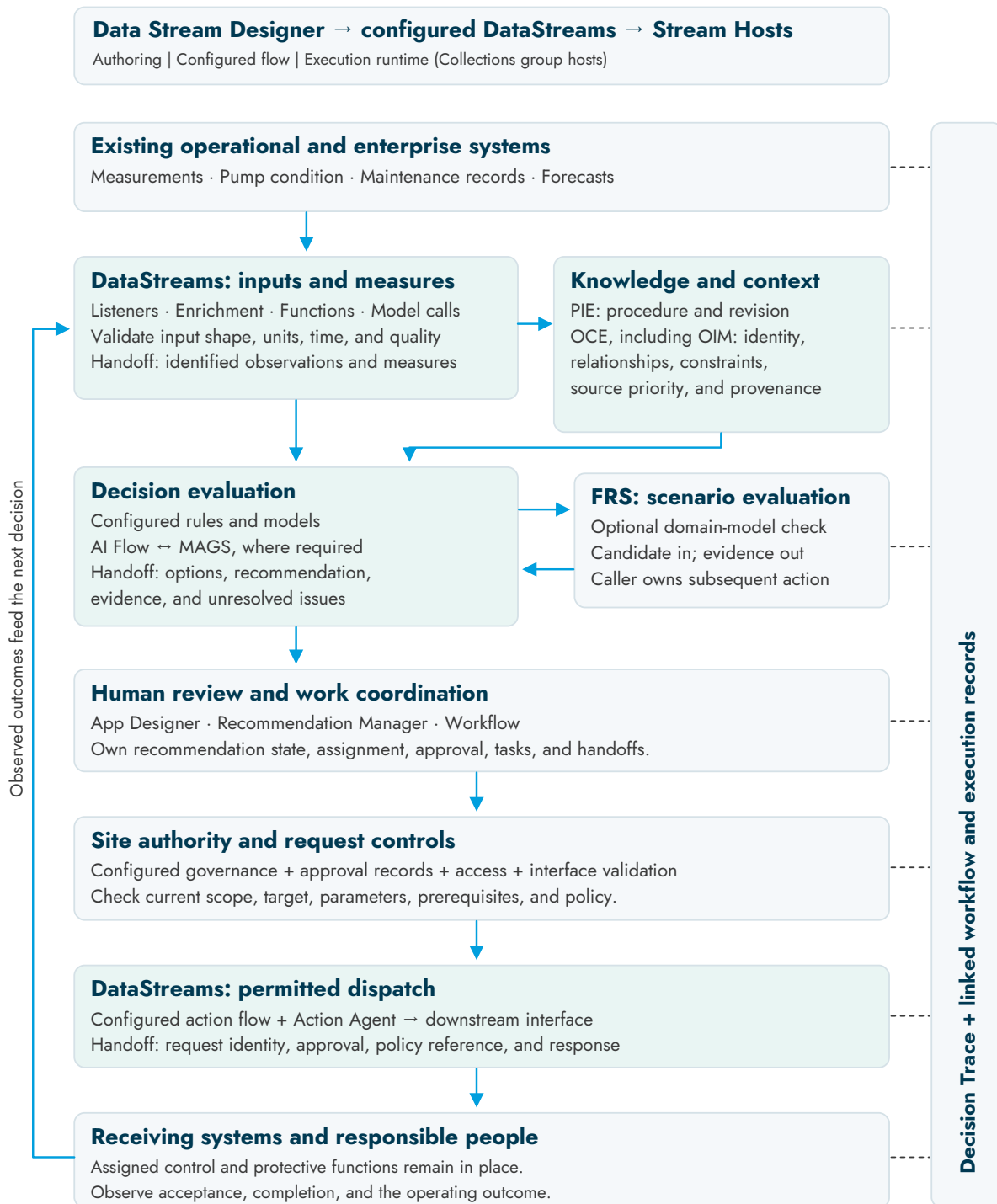
At the station, an input flow could acquire reservoir measurements and pump-condition events. A calculation could evaluate a candidate plan against the reserve requirement. A separately authorized action flow could submit an approved request to the maintenance system. These are distinct responsibilities even when they share a runtime or configuration environment. **The configured flow must expose where a decision is evaluated and where an action is permitted.** The team should be able to change the reserve calculation while identifying which input and action interfaces remain unchanged.

Separate authoring, configuration, and execution

Data Stream Designer is the authoring environment. A DataStream is the configured flow. A **Stream Host** is the application that executes the processing components. **Collections** group Stream Hosts and provide the execution grouping selected for a stream or its individual components. These distinctions let the team describe both the logical design and where its parts run. [14](#)

For the station, record the Collection assignment, required host environment, component versions, connections, and access rights. Runtime placement follows the site's connectivity and operating requirements. It does not require the entire decision process to run on one central server.

CONNECT OBSERVATION, DECISION, AND RESPONSE



Proposed station configuration. Arrows are handoffs to configure and test.

© 2026 XMPro. All rights reserved.

Figure A.1. A proposed XMPro implementation of the decision cycle. DataStreams appear on the observation and action paths; optional FRS evaluation returns evidence before dispatch. The evidence record spans the process. Arrows specify handoffs to configure and test, not prebuilt integrations or permission to act. Product roles draw on sources [1-14].

Establish the meaning of the inputs

The **Procedural Intelligence Engine (PIE)** prepares procedural knowledge for review. XMPro describes extraction into a structured model, comparison with source passages, retained review states, and a signed, versioned handover. Preserve the requirements, conditions, exceptions, and source references that determine whether a passage applies. Signing records integrity information; the responsible specialist still resolves open interpretation and review questions. [1](#)

The **Operational Context Engine (OCE)** supplies governed identities, relationships, constraints, source priority, and provenance. The **Operational Intelligence Model (OIM)** belongs within OCE's identity and ontology model, which defines entities and their relationships. DataStream enrichment can supply useful attributes or candidate mappings. OCE governs the operational meaning that consuming flows, models, and applications use. [2](#)

For P2, connect the condition signal, maintenance record, and procedure reference to the same asset. Identify the approved source for availability and the reservoir reserve requirement. Pass units, timestamps, quality, and applicability with the values the decision consumes. The handoff must make conflicting or stale information visible; a successful lookup alone cannot establish fitness for this decision.

Use calculations and reasoning where each is needed

Known decision logic can run in configured DataStream rules, functions, and model calls. For the pumping plan, an engineer defines and validates the reserve calculation and eligibility checks. The resulting measures become evidence for comparing candidate plans. Repeatability of a calculation does not establish that its formula or assumptions are correct for the station.

AI Flow adds model reasoning inside visible DataStream workflows, with context assembly, output validation, and routing. **Multi-Agent Generative Systems (MAGS)** supports recurring cognitive decision loops with objectives, planning, memory, and team coordination. AI Flow can prepare or process information for a MAGS loop; a MAGS decision can request a bounded calculation or workflow service. The design must specify each request, response, failure condition, and permitted scope. These components can cooperate within one application. [4](#) [5](#)

Front-Running Simulation (FRS) evaluates proposed scenarios against a suitable domain model. In Figure A.1 it supplies evidence to option evaluation, before any request is dispatched. The caller receives the evaluation and decides how to use it under the applicable authority. A favorable result does not authorize execution. For this station, the team must select and validate a model that supports the intended pumping-plan question; a general water-domain description cannot establish hydraulic feasibility. [6](#)

An **Agent** in Data Stream Designer is a packaged processing component. The name alone does not imply cognitive reasoning. Publishing a `DataStream` starts its execution on the relevant Stream Hosts, so a test publication needs controlled connections and permissions before it runs. [11](#) [12](#)

Assign review, state, and action ownership

App Designer provides the operational experience: the context, alternatives, evidence, approval state, and observed outcome people need to see. **Recommendation Manager** supports recommendation handling, including assignment and disposition. **Workflow** supports human tasks, approvals, escalation, handoffs, and process state. In the station design, specify which component owns each state and how the others receive updates. An operator's approval, a sent request, and a completed job must remain distinguishable. [7](#) [8](#) [9](#)

The action boundary combines configured governance checks, approval records, access controls, and request validation. A `DataStream` with an Action Agent can call the downstream interface, but the configuration must establish which request is allowed, for which target, under which current conditions. Carry the approval reference and applicable policy revision into that check. An expired approval or a changed prerequisite must produce the agreed rejection or escalation. [3](#) [10](#)

Local control and safety systems retain their assigned functions. For the first advisory implementation, the permitted output may be a recommendation for a person to act on. The same responsibility map can later support a bounded integration action after the team has granted and tested that permission.

Keep three design choices separate

Dimension	Labels used	What it determines
Implementation tiers	<code>DataStream</code> s; AI Flow; MAGS	How configured processing, model reasoning, and cognitive decision loops contribute. They can coexist.
Participation levels in this book	Monitor & Predict; Advise & Guide; Coordinate & Autonomize	How people and software participate in the decision and response.
Control modes	Human-Controlled; Human-Approved; Policy-Controlled	How authority is retained, approval is required, or action is permitted within policy.

The implementation tiers describe mechanisms; using MAGS does not itself grant authority. XMPPro's control-mode descriptions address permission arrangements. Assign those arrangements to particular actions, as Chapter 3 does. [4](#) [10](#)

The current platform website labels its participation phases **Monitor & Predict**, **Advise & Coordinate**, and **Operate Autonomously**. This book retains its broader capability labels consistently across the chapters and figures. The phrases cover related territory, but they are not exact equivalents: coordination can remain human-approved, and autonomous operation depends on explicitly granted limits. [18](#)

Change the reserve policy through explicit configurations

Return to Chapter 3's revised minimum reservoir reserve. Engineering and operations have approved the requirement; the implementation now needs to apply it consistently. The following is a proposed demonstration for the station, using identified configurations and recorded handoffs. It does not assume an automatic end-to-end release mechanism.

- 1. Represent the approved requirement.** Use PIE to preserve the reviewed procedure passage and its revision. Represent the applicable constraint in OCE with its source, station scope, units, effective time, and approval reference. The procedure owner and context owner confirm that the operating interpretation matches the approved requirement. The handoff must select the applicable revision explicitly.
- 2. Update the evaluation.** Configure the reserve calculation or model call in the relevant `DataStream` to consume that constraint. Record the calculation and model versions. Compare a candidate plan that meets both requirements, one that meets only the previous requirement, and a case with missing or stale evidence. If FRS is used, record its scenario inputs, model revision, and returned evaluation. A reasoning component can explain the result; the validated calculation establishes whether the candidate meets the specified reserve test.
- 3. Apply the approval rule.** Configure the Workflow rule that routes the recommendation to the responsible reviewer. Identify whether the revised requirement changes the approval path, the evidence presented, or both. In this example, a plan that fails a mandatory reserve requirement is excluded. Escalation asks for a revised plan or an authorized engineering resolution; clicking an approval button cannot waive the requirement.
- 4. Check the request at dispatch.** Before a configured `DataStream` and Action Agent submit an approved request, validate its target, parameters, permission, approval reference, and policy currency against the site's requirements. Specify how the interface handles a changed condition, a duplicate request, or an uncertain response. An approval made before the policy change must follow the agreed rule for re-evaluation. The receiving system retains its own access and execution controls.
- 5. Preserve the versions and results.** Record the procedure revision, context or constraint identifier and revision, `DataStream` version, calculation or model version, approval-rule version, and action-interface configuration reference. Link these to the recommendation, approval, request identifier, acknowledgment, and observed outcome. Where a component does not expose a native version field, define a controlled identifier or manifest entry. These are acceptance requirements for the evidence record, not a claim that every field is captured automatically.

The measurement connection can remain unchanged.

Show that the existing reservoir input still supplies the same identity, units, timestamps, and quality required by the revised evaluation. Then demonstrate that the changed constraint alters the appropriate candidate-plan result without rebuilding that connection. If the new requirement needs information the interface does not supply, record the additional integration work. The architectural benefit is a change whose scope the team can explain and test.

Support access, release, and reuse

The runtime diagram concentrates on one decision cycle. The surrounding platform and implementation work determine who can configure it, how changes reach a site, and how another team can reuse it.

Subscription Manager provides identity, role-based access, product subscriptions, and sign-on arrangements. Those controls govern access to platform objects and services. Authority to perform a particular operating action also depends on the site's policy, the request's scope, and the approval or conditions required at dispatch. Configure both levels and test them together. [15](#)

Versioning and deployment tooling support the change discipline in Chapters 3 and 4. XMPro's versioned documentation describes revisions for DataStreams, Agents and Connectors, applications, recommendations, and forms. It also distinguishes the version opened for editing from the version currently published. The release record should name the versions actually running, their dependencies, and their environment bindings, including Collection assignments and downstream endpoints. [19](#)

Verify the deployment scope in the chosen releases. Object versioning does not establish coordinated promotion or rollback across every module. Context revisions, model packages, workflow state, credentials, and requests already sent may need separate handling. The recovery plan must address each affected responsibility and the physical operation, following Chapter 3. For reuse at another station, retain the common decision structure while checking local mappings, requirements, permissions, and acceptance evidence.

APEX and XMForge support the implementation work around the runtime. XMPro's AgentOps documentation describes APEX (Agent Platform EXperience) as support for developing and managing agent systems. XMForge's public description emphasizes readiness, use-case selection, validation, implementation, and replication, supported by templates, stage gates, and evidence. Relate that work to Chapter 5's decision brief: establish the decision owner, scope the first permitted response, agree the evidence, and determine what allows the team to proceed. Keep these implementation and management responsibilities outside the operational flow in Figure A.1. [16](#) [20](#)

Real-Time Causal AI is an optional analytical capability when the decision needs diagnosis or evaluation of cause and effect. XMPro describes engineering-informed models that produce explained causal assessments from operating data. Include such a model only when its intended use, inputs, and validation support the decision. The pumping-plan example can begin with an accepted condition assessment and a validated reserve calculation; it does not require every analytical capability in the portfolio. [17](#)

The component selection follows the work the decision requires. A reusable configuration becomes useful at another site when its assumptions, dependencies, and authority can be examined there.

Connect the decision to evidence of its outcome

XMPro describes **Decision Trace** as preserving observations, planning, action intent, constraints, and related evidence for a MAGS decision. In the station application, link that trace to the recommendation, workflow approval, dispatch record, receiving-system response, and observed outcome. Assign the owner of each record and the identifiers that connect them. The diagram's evidence column represents this integration responsibility; it does not imply one product automatically records every event. [13](#)

ASK FOR EVIDENCE FROM THE CONFIGURATION

Proposed acceptance checks for the station application.

CHECK	SHOW THE RESULT	REVIEW RESPONSIBILITY
Procedure source and revision	A requirement linked to its source; review status and selected revision visible.	Procedure owner and specialist
Context identity and freshness	The P2 signal resolves to the right asset; stale required input triggers the agreed response.	Source owners and operations
Action authority	An allowed proposal reaches its reviewer; an out-of-scope request is blocked.	Decision owner and access owner
Request status	Recommendation, approval, sent request, acceptance and completion remain distinct.	Operations and receiving-system owner
Controlled policy change	A revised rule changes the expected cases; active revision and affected work are identifiable.	Release owner and decision owner

Name actual reviewers and agree expected results before the demonstration.

© 2026 XMPro. All rights reserved.

Figure A.2. Evidence to request from the configured implementation. These tests apply the book’s architecture to the station; they are proposed acceptance checks, not a product certification or completed test report.

When MAGS is absent, the implementation still needs the decision evidence described in Chapter 3. Name the workflow, calculation, and execution records that supply it. A sent request and a successful tool response do not prove a physical outcome. Observe what happened, associate it with the decision, and make unresolved results visible to the next review.

Sources and implementation checks

Sources checked September 18, 2026. Public descriptions establish component roles; versioned documentation supports the narrower runtime and configuration statements. The source register also records the targeted local architecture-document checks used to review the handoffs. Selected-release behavior, licensing, integration, model suitability, and site acceptance require implementation evidence. No configured deployment was tested for this appendix.

1. [Procedural Intelligence Engine](#). Procedure extraction, review, source references, and signed versions.
2. [Operational Context Engine](#). Governed meaning and provenance; OIM within OCE.
3. [Data Stream Designer](#). Listeners, transformations, functions, analytics, and Action Agents.
4. [AI Flow](#). Workflow reasoning, validation, routing, and cooperation with MAGS.
5. [Multi-Agent Generative Systems](#). Agent teams and recurring cognitive decisions.
6. [Front-Running Simulation](#). Evaluation of proposed scenarios against domain models.
7. [App Designer](#). The user-facing operational application.
8. [Recommendation Manager](#). Recommendation handling and disposition.
9. [Workflow](#). Human tasks, approvals, handoffs, escalation, and state.
10. [Governance, Security & Auditability](#). Control modes and authority boundaries.
11. [Agent, documentation 4.5](#). Packaged DataStream processing components.
12. [Running Data Streams, documentation 4.5](#). Publishing and Stream Host execution.
13. [How MAGS Works](#). Decision Trace and bounded execution.
14. [Collection and Stream Host, documentation 4.5](#). Execution applications, grouping, and assignment.
15. [Subscription Manager](#). Identity, access, sign-on, and subscriptions.
16. [XMForge & Methodology Execution](#). Readiness, stage gates, validation, and replication.
17. [Real-Time Causal AI](#). Engineering-informed causal assessment.
18. [XMPro Agentic Operations Platform](#). Current website participation labels.
19. [Version, documentation 4.5](#). Object revisions and published versions.
20. [APEX AgentOps, XMPro public repository](#). Agent development and lifecycle-management roles.

Scope and sources

Product roles are attributed to the cited sources; the station mapping, policy-change demonstration, and acceptance checks are proposed designs.

© 2026 XMPro. All rights reserved.

Working aids for a first application

Use these aids after identifying a decision worth examining. They organize the information and evidence needed for a scoped application. Replace the station's illustrative entries with local requirements and named owners; an unfilled field is work to resolve before accepting the relevant use.

B.1 Define the decision and its intended value

THE FIRST-DECISION CANVAS

Illustrative brief for one transfer station. Resolve site details before use.

01 Objective Prepare a reviewable pumping plan when operating constraints change.	02 Owner Name the accountable operations owner and the required specialist reviewers.
03 Trigger P2 condition or another relevant operating constraint changes.	04 Context Availability, demand, reservoir state, procedures and source quality.
05 Options Compare eligible windows A and B and their associated pumping plans.	06 Constraints Applicable reserve, equipment and procedure requirements; cite sources.
07 Authority Advise & Guide. People direct the response and approve as assigned.	08 Evidence Inputs, policy revision, options, rationale and approval record.
09 Outcome Agreed measures for reviewability, corrections and decision preparation.	10 Scope One station and its service reservoir; advisory use through an agreed path.

Unresolved sources, role assignments and acceptance measures become assigned tasks.

© 2026 XMPro. All rights reserved.

Figure B.1. A brief for the first decision. The canvas is a planning aid, with illustrative station entries. It implies no approved operating limit, implementation commitment, or validated result.

Give the brief an owner and a revision. Record why the decision matters, the current difficulty, and the benefit to assess. Identify the operation and systems in scope. Agree the initial participation level and the people who will accept the result. A broader objective, another station, or a new action permission requires its own assessment.

B.2 Prepare the requirement and its context

For each required source, record its owner, revision or observation time, equipment scope, meaning, and conditions of use. Include units and freshness expectations for live inputs. Record how conflicts are resolved and what the decision does when required information is missing.

Requirement record	What the station example needs
Source and approval	The controlled instruction and source establishing the applicable reserve minimum.
Reviewed meaning	Applicable value, units, planning period, method, and treatment of uncertainty.
Asset and operating scope	The reservoir, station configuration, and conditions to which the rule applies.
Software use	The identified evaluator, its configuration, and the sources it consumes.
Test and retained evidence	Expected pass, fail, and unresolved cases; references linking source, interpretation, inputs, revision, and result.

Test the meaning, as well as the connection. A successful interface call does not establish that a maintenance status or reservoir measurement means what the evaluator assumes. Have the relevant specialist review the result against representative local cases.

B.3 Assign authority to each action

Action in the example	Initial arrangement	Boundary to verify
Prepare a recommendation	Software assembles and evaluates; people direct the response.	Required evidence and applicable constraints.
Approve the pumping plan	Assigned operations reviewer.	The identified plan, conditions, and current evaluation.
Submit the scheduling request	Responsible person.	Current approval and the receiving system's request semantics.
Copy an approved window into a planning record	Possible later, separately approved delegation.	Allowed fields and target, valid approval, unchanged plan, rejection and intervention behavior.
Isolate equipment or change protection	Outside this example's delegated scope.	Assigned site and engineering procedures.

For each software action, name the capability, accountable owner, prerequisites, approval requirement, and enforced rejection path. Record how people suspend permission and handle work already accepted by another system. An instruction in an agent prompt needs corresponding enforcement at the action interface.

B.4 Test and introduce a change

Describe the approved requirement being revised, the reason, applicability, and affected decision definitions. Identify expected changes in behavior and the boundaries that should remain stable. Use a retained case where the effect is observable, with independently established expected results.

Check	Evidence to retain
Intended behavior	Results under both definition revisions, with the same identified inputs.
Unmet prerequisites	Stale or missing inputs, conflicting sources, and the resulting hold or escalation.
Authority	An allowed action, a blocked action, and changed conditions after approval.
Pending work	Which proposals need reassessment and which revision governs further requests.
Receiving state	Request identity, acknowledgment, uncertain-response handling, and prevention of duplicate submission.
Release and intervention	Accepted scope, compatible revisions, observation owner, and how to suspend use.

Record the configuration actually used, including dependencies. If a component exposes no suitable version identifier, establish a controlled reference and prove how it identifies the active configuration. Do not assume that one release label covers every module or external dependency.

Recovery needs both a software and an operational assessment. An earlier requirement may no longer be approved. An action already accepted or physically started may need cancellation, correction, or a site response. Record the remaining work explicitly when suspending or restoring software.

B.5 Prepare reuse at another site

The receiving package should identify the decision definition, input and action interfaces, context requirements, tests, evidence-record structure, and local configuration points. Include its revision, dependencies, intended use, and accountable owner.

For each element, state what can be shared, what must be supplied locally, and what the receiving team must validate. Equipment identities, operating requirements, source meanings, calculations, access rights, and approval assignments require local assessment. Preserve the relationship between any local variation and the common definition.

The receiving site supplies its own cases and expected results. It may reject the pattern or reuse selected parts where the operation differs materially. Measure adaptation and validation effort alongside ongoing maintenance. Confirm that local people can interpret and maintain the accepted decision.

B.6 Connect the benefit to a baseline

Choose the measures that justify the application. Agree who collects them, the observation period, and what result would support continued use. Include preparation, integration, validation, training, and maintenance effort in the assessment.

Intended benefit	Evidence to collect	Comparison to make
Less preparation effort	Staff time, missing-input requests, handoff delays, and corrections.	Comparable decisions before and after introduction.
Consistent application of requirements	Applied revisions, unexplained differences, and reviewed exceptions.	Matched cases across shifts under the same definition.
More accessible expertise	Whether another qualified person can explain and use a rule, including when to escalate.	Task completion and specialist help required with and without the maintained record.
Controlled change and reuse	Work to locate logic, revise it, test it, adapt local settings, and obtain acceptance.	Equivalent changes or sites, including continuing maintenance.
Retained systems and replaceable components	Integration, interface testing, adapter, and migration work.	Total effort for an equivalent connection or replacement.
Useful AI assistance or delegated action	Proposal corrections, rejected requests, intervention, handling time, and completed authorized actions.	The advisory arrangement and each separately approved scope.
Improved operating performance	Relevant energy, equipment availability, and service measures.	An agreed baseline accounting for demand, tariffs, equipment, and operating requirements.
Improvement from recorded experience	Evidence supporting an investigation and the tested correction.	Investigation effort and recurrence after the reviewed change.

Pair an efficiency measure with the relevant service or operating requirement. A shorter preparation process is useful only if the response remains fit for its accepted purpose. Explain other changes that may have influenced an observed result.

Before acceptance, the accountable owner should be able to locate, change, test, release, and explain the scoped decision logic. Separately, assess whether its demonstrated value justifies the cost of maintaining it. These aids organize that assessment; they do not certify the implementation or establish a plant's operating limits.

Glossary

These definitions describe how terms are used in this book. Chapter references lead to the fuller explanation. XMPro product names and their implementation roles are covered in Appendix A.

Action authority

Permission to perform a particular action under defined conditions. Approval of a plan, permission to submit a request, and authority to operate equipment are separate responsibilities. See Chapters 2 and 3.

Action interface

The connection through which a person or application requests an action from another system. Its definition includes the request's meaning, required permissions, and how acceptance, rejection, or an uncertain response is reported. See Chapters 2 and 3.

Agent

In the main narrative, software that can reason about a task and use permitted capabilities, such as requesting missing information. Its reasoning does not itself grant authority to act. In XMPro Data Stream Designer, an Agent is a packaged processing component; that product term does not necessarily imply reasoning. See Chapter 2 and Appendix A.

Approval policy

The rules that determine who reviews a proposal, what evidence they need, and which conditions require approval. A previous approval may need reassessment when its governing requirement or relevant evidence changes. See Chapters 2 and 3.

Architectural acceptance

Evidence that the implementation organizes and changes the decision logic as intended within an agreed scope. This assessment is distinct from measuring whether the application improves the operation. See Chapter 5.

Constraint

A requirement that a candidate plan must satisfy to remain eligible. In the station example, the approved minimum reservoir reserve is a constraint; a preference for lower energy cost cannot override it. See Chapter 3.

Context

The information needed to interpret an observation or evaluate a decision, including asset identity, units, time, operating conditions, applicable requirements, and source quality. A connected data value still needs the right meaning for its intended use. See Chapters 2 and 3.

Coordination logic

The rules and sequence for arranging the people, systems, approvals, and requests needed to carry out a response. It includes what should happen when a handoff is incomplete or uncertain. See Chapters 2 and 3.

Decision definition

The maintained description and software representation of what a decision seeks, requires, may request, and must record. Its parts can span several configurations and services, provided their relationships, revisions, and ownership remain identifiable. See Chapter 2.

Decision evidence

Records connecting the inputs, governing definition, evaluated alternatives, recommendation, approval, requested action, and observed outcome. Together they allow the team to examine why a response occurred and what remains unresolved. See Chapter 3.

Dependency

Something a decision relies on, such as an input source, calculation, rule, approval role, or action interface. Identifying dependencies helps establish which parts need attention when a requirement changes. See Chapters 1 and 2.

Freshness requirement

The condition determining whether an observation or forecast is recent enough for the decision. Required information outside that condition needs the defined response, such as refresh, hold, or escalation. See Chapter 3.

Input interface

The connection supplying information to the decision. Its agreed meaning includes what a value describes, its units, observation time, period of applicability, quality, and behavior when the source fails. See Chapter 2.

Local configuration

The site-specific assets, sources, requirements, permissions, and responsible people associated with a shared decision pattern. Each receiving site must establish the suitability of its own configuration and test cases. See Chapter 4.

Operational decision and coordination logic

The rules and relationships governing what matters, which options qualify, how to compare them, who participates, and what may happen next. This is the responsibility the proposed architecture makes explicit and maintainable in software. See Chapters 1 and 2.

Participation levels

The book's descriptions of how people and software contribute within the architecture. **Monitor & Predict** organizes information for people to act on. Under **Advise & Guide**, software develops proposals for human review. **Coordinate & Autonomize** permits agent actions within explicitly granted limits. These levels can coexist within an application. See Chapter 2.

Ranking policy

The method for comparing candidates that already meet the mandatory requirements. The illustrative station prefers the eligible plan with lower expected energy cost. Ranking cannot make an ineligible plan acceptable. See Chapter 3.

Retained input case

The same identified set of inputs used to compare behavior under two decision definitions. Holding the inputs fixed helps the team inspect the effect of a changed rule. Expected results still need an independent basis. See Chapter 3 and Appendix B.

Reusable decision pattern

A shared decision structure accompanied by its assumptions, interface requirements, configuration points, tests, evidence structure, dependencies, and ownership. Another site assesses which parts fit and supplies its own validated local configuration. See Chapter 4.

Revision

An identified change to a requirement, configuration, or other part of the decision definition. A useful record links the active revisions to the decision that used them; one workflow's version label may not identify all of its dependencies. See Chapters 2 and 3.

Software-Defined Operations

An architectural pattern that decouples operational decision and coordination logic from the underlying applications, automation systems, and physical infrastructure, enabling that logic to be configured, changed, and governed through software. This is the book's proposed definition. See Chapter 2.

Unresolved result

A result indicating that the required evidence is inadequate to establish whether a candidate satisfies a requirement. The decision follows its agreed hold, refresh, escalation, or fallback behavior. See Chapter 3.

Make the next operational change understandable.

When an operating requirement changes, a team may have to revise a calculation, reconcile a procedure, update an approval route, and find every affected use. The governing logic is spread across systems and people.

Software-Defined Operations gives that logic an identifiable definition in software, with explicit connections to the systems it uses. This book follows one pumping-plan decision to show what the architecture means, how a controlled change works, and what the organization must maintain.

Learn how to assess the potential to coordinate work across responsibilities, adapt a response while retaining suitable systems, and reuse a decision pattern at another site. Then use the recognition questions and working aids to choose a first application and establish what would count as useful evidence.

For operations leaders, engineering and systems teams, and business sponsors.

Pieter van Schalkwyk