"ADVANCED JOURNAL OF UNIVERSITY SCIENCE AND CREATIVE INNOVATIONS"

FORMALIZING THE DIGITAL LIFECYCLE OF STATE SCIENTIFIC PROGRAMS USING IDEF0 AND IDEF3 PROCESS MODELS

A'zamov Temur Narzullayevich
Advisor to the Director of the Agency for Innovative Development on Digitalization and Artificial Intelligence Implementation, Uzbekistan
https://doi.org/10.5281/zenodo.21897754

Abstract. *Public research funding runs as a chain of stages from call formation to the handover of results. Regulations frame that chain as obligations and deadlines rather than as an information process, an omission that becomes costly during digitalization. This article formalizes the lifecycle of state scientific programs from two perspectives. A context-level IDEF0 model fixes what the process transforms and under which controls and mechanisms; an IDEF3 scenario model of 13 units of behavior and two exclusive-OR junctions fixes stage order and the conditions for acceptance, revision, or rejection. Four descriptors - a layered system tuple, a routing triple, an integration descriptor, and an exchange graph - link the diagrams to an implementable architecture, and a status function with an explicit transition relation makes the scenario executable by a workflow engine. Modelling exposed three transitions the governing documents leave implicit: divergent expert scores, monitoring deviations, and the target of the revision loop. Each requires an owner, a status value, and an audit entry. Indicators from a deployed platform are illustrative, not evaluative; conditions for proper evaluation are stated.*

Key words: *IDEF0, IDEF3, business process modeling, research administration, workflow automation, digital government, state scientific programs.*

Introduction

A state scientific program passes through a long series of decisions before any research money is spent. A ministry or funding council forms the program and announces a call. Research teams prepare and submit applications. Those applications are screened for completeness and for compliance with the financial rules, specialists review the scientific content, a scientific-technical council weighs the reviews and recommends an outcome, and a ministry-level body confirms it. Contracts are then signed, financing is released, progress is monitored, and a final report is assessed before the results are handed to whoever will use them. Every one of these steps is regulated. Taken together, however, they are seldom written down as a single process.

The omission has practical consequences. When the stages exist only as separate administrative instructions, no document states which office holds an application between two

stages, what its status is at any given moment, or what evidence survives once a decision has been taken. Applicants ask where their file is and get an answer only if somebody happens to know. Reviewers are assigned by hand, and deadlines are tracked in parallel spreadsheets that disagree with one another. Governments at very different income levels have responded to this pattern by moving administration onto shared digital platforms and by trying to do more with fewer administrative resources (Janssen & Estevez, 2013). A platform, though, can only automate a process that has first been described precisely enough to be automated.

Business process modeling supplies that description, and several notations could supply it. The IDEF family is attractive in this particular setting for one specific reason: it keeps apart two questions that regulatory texts habitually run together. IDEF0 asks what a function transforms, under which constraints, and by which agents (National Institute of Standards and Technology [NIST], 1993). IDEF3 asks in what order things happen and what decides the branches (Mayer et al., 1995). Research administration needs both answers at once. A funding decision is a transformation of documents into an entitlement, and it is also a position in a queue with a deadline attached.

The literature on IDEF applications is dominated by manufacturing, logistics, and engineering design. Public research administration has attracted far less attention, even though it is unusually well suited to formal process description: the actors are named in law, the decision criteria are published, and the artifacts are documents with legal standing. This article addresses that gap by asking two questions. First, what does the lifecycle of a state scientific program look like when it is expressed simultaneously as an IDEF0 function and an IDEF3 scenario? Second, what has to be added to those two diagrams before software can actually execute them? The remainder of the article reviews the modeling background, sets out the procedure and notation used, presents the two models together with the descriptors that link them to system architecture, and then discusses what the exercise revealed and what it cannot yet support.

Theoretical Background

Functional and Process Perspectives in the IDEF Family

IDEF0 descends from the Structured Analysis and Design Technique developed for the U.S. Air Force ICAM program and was later published as a federal standard (NIST, 1993). Its central construct is an activity box carrying four classes of labeled arrow, conventionally abbreviated as ICOM. Inputs arrive from the left and are consumed or converted by the activity. Controls arrive from the top and constrain how the activity is performed without being consumed by it. Outputs leave to the right. Mechanisms arrive from below and denote the people, bodies, or systems that carry the work out.

Most of the analytical value of IDEF0 sits in the distinction between an input and a control, because that is the distinction administrative prose most often blurs. An application is consumed by a review; an evaluation criterion is not. The criterion governs the review and survives it unchanged. Forcing a modeler to place each item on one side of that line makes explicit which legal instrument constrains each step and which body performs it - precisely the information a regulation assumes its reader already possesses.

IDEF3 works differently. It captures a process as a network of units of behavior (UOBs) connected by precedence links, with junctions specifying how flow splits and merges: conjunctive junctions where all paths proceed, inclusive disjunctive junctions where one or more proceed, and exclusive junctions where exactly one does (Mayer et al., 1995). IDEF3 was designed as an elicitation method, which matters for how its output should be read. A scenario diagram records

how participants describe a process as happening, not necessarily how a normative document says it should happen, and the difference between those two readings is often where the interesting problems live.

Combining the Two Perspectives

The complementarity of the IDEF methods has been examined repeatedly. Kim et al. (2003) showed that IDEF models can serve as a business-oriented front end to UML, with semantic content re-expressed in the target notation rather than rebuilt from scratch. Whitman et al. (1997) took a more demanding route and converted static IDEF0 and IDEF3 models into executable discrete-event simulations, adding the annotations that the static notations leave out. Both lines of work converge on the same conclusion: neither a functional nor a scenario view is self-sufficient, and the effort involved in reconciling them is where much of the analytical benefit accumulates.

Why IDEF Rather Than BPMN

BPMN is the mainstream choice for this kind of work and would have been defensible here (Object Management Group [OMG], 2013). It has richer event semantics, wider tool support, and - decisively for many projects - a direct path to execution. Petri-net foundations and process mining allow a model to be checked against real event logs rather than accepted on the modeler's authority (van der Aalst, 2016), and the broader business process management literature supplies the redesign and governance frame around such models (Dumas et al., 2018; Scheer, 2000; Weske, 2019). Choosing IDEF over BPMN was a trade. What IDEF0 offers that BPMN does not enforce is the mandatory separation of control from mechanism at every activity. In a regulated domain, where the question "under which rule?" is as consequential as the question "who does it?", making that separation compulsory rather than optional is worth the loss of executability - provided, as this article argues below, that the executability is recovered by other means.

Method

Scope and Modeling Inputs

The model covers one program cycle, bounded at the front by the formation of a call and at the back by the transfer of completed results to a receiving organization. Two adjacent processes are deliberately outside the boundary: the budget planning that determines how large a program will be, and whatever happens to a result after it has been transferred. Both are governed by different actors and different instruments, and pulling them in would have produced a diagram too coarse to be useful at either end.

Three sources fed the model. The first was the body of regulatory documents governing the announcement, evaluation, contracting, and monitoring of state scientific programs. The second was the module and status specification of the digital platform under development, which supplied the vocabulary of information objects and the set of statuses each object can hold. The third was the sequence of stages as practiced, reconstructed from the order in which documents are produced and signed. Where the second and third sources disagreed with the first, the disagreement was recorded rather than smoothed over; two such cases are reported in the Results.

Modeling Procedure

Modeling proceeded in five steps. The context diagram was fixed first, with every candidate arrow tested against a single question: does the activity consume this item, or is the item still intact afterwards? Items that survive were classified as controls, items that do not as inputs, and the bodies and systems performing the work as mechanisms. Second, the scenario was decomposed into units of behavior, each named with a verb phrase and each corresponding to a point at which an information object changes hands or changes state. Third, junction types were

assigned. Both branch points in the lifecycle turned out to be exclusive: an application is accepted, returned, or rejected, and these outcomes cannot hold simultaneously.

Fourth, the two models were cross-checked. The test applied was bidirectional - every UOB had to be traceable to at least one function in the IDEF0 decomposition, and every output named in the context model had to be produced by at least one UOB. This is the step that generated the most revision, and it is described in the Results because its findings are substantive rather than procedural. Fifth, the formal descriptors set out below were attached, so that the diagrams could be read as a specification rather than as documentation.

Notation

Four descriptors connect the process models to an implementable architecture. Note that the notation used here departs from the informal version circulated earlier in the project, in which the symbol for the integration layer collided with the symbol for the integration descriptor and the symbol for status collided with the symbol for the system as a whole. The revised notation is as follows.

The platform is described as a five-layer tuple:

$$\mathbf{\Sigma = \langle\ U, B, D, X, T\rangle}$$

where U is the user layer, B the business-logic modules, D the data layer, X the integration layer, and T the technical infrastructure. This layering follows the conventional practice of describing a system through the concerns of its stakeholders rather than through its source structure (ISO/IEC/IEEE, 2011).

Movement of information objects between modules is described as a triple:

$$\mathbf{R = \langle\ N, P, C\rangle}$$

where N is the set of nodes, $P \subseteq N \times N$ the set of admissible transmission paths, and C the set of routing rules that determine which path in P is taken for a given object.

Each integration point $x \in X$ carries a descriptor:

$$\mathbf{\iota(x) = \langle\ a, \tau, \pi, \sigma\rangle}$$

with a the endpoint, τ the data type exchanged, π the protocol, and σ the current integration status. The exchange topology of the platform is then the directed graph $G = (V, E)$, where V is the set of modules and $E \subseteq V \times V$ the set of exchange links; the routing rules C induce a labeling on E.

These four descriptors are standard, and on their own they do not make the scenario executable. What does is the addition of an explicit status machine. Let O be the set of information objects and Q the finite set of statuses an object may hold. A status function

$$\mathbf{st: O \to Q}$$

assigns a current status to every object, and a transition relation

$$\mathbf{\delta \subseteq Q \times \Phi \times Q}$$

specifies which status changes are permitted, with Φ the set of conditions evaluated at branch points. Every transition $(q, \varphi, q') \in \delta$ writes an audit record of the form $\langle$ actor, timestamp, object, q, q′, module$\rangle$. The practical significance of this addition is discussed below: it is the point at which an IDEF3 junction stops being a picture of a decision and becomes a decision the system can make and later account for.

Results

The Context-Level Functional Model

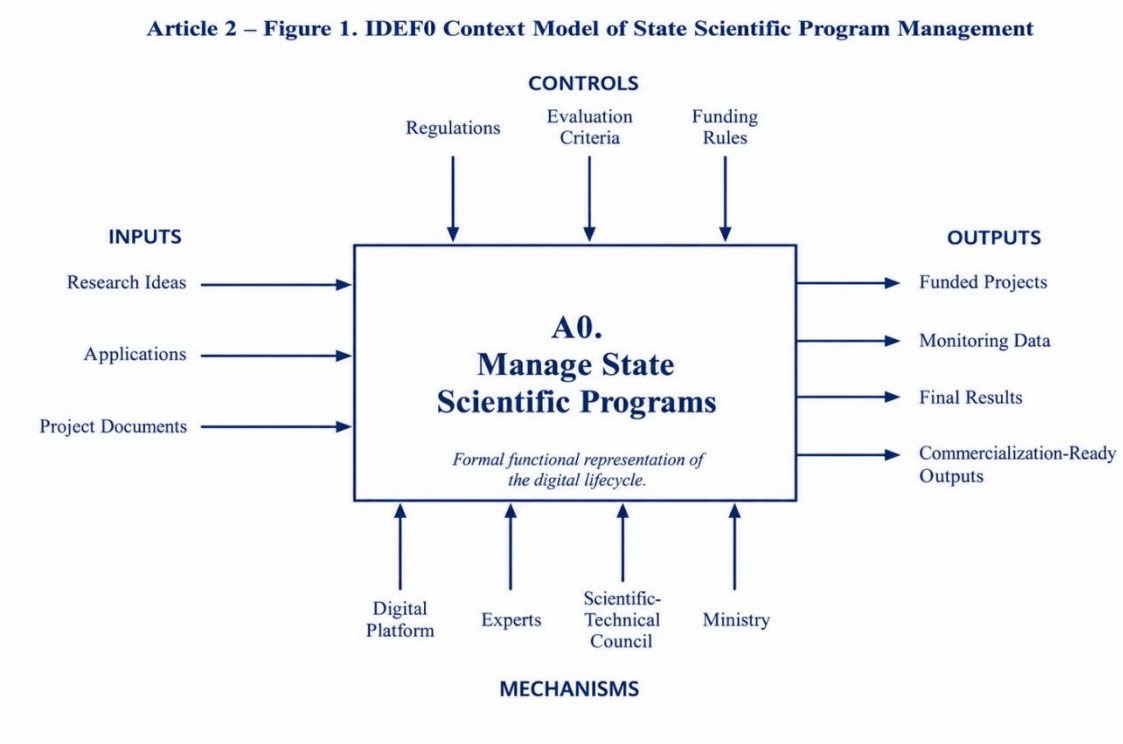
Figure 1 presents the process at its highest level of abstraction as a single activity, A0, Manage State Scientific Programs. Three inputs cross the left boundary: research ideas, applications, and project documents. Three controls constrain the activity from above: regulations,

evaluation criteria, and funding rules. Four mechanisms carry it out: the digital platform, the expert corps, the scientific-technical council, and the ministry. Four outputs leave to the right: funded projects, monitoring data, final results, and commercialization-ready outputs.

One classification in Figure 1 deserves comment because it is not obviously correct. Research ideas are shown as an input, but an idea is not a document and cannot be consumed by an administrative activity in the way an application can. It is better read as a placeholder for pre-application intent - the point at which a research team decides to respond to a call. Retaining it makes the model honest about where the lifecycle actually begins for the applicant, at the cost of some formal tidiness. The alternative, starting the diagram at application submission, produces a cleaner model of a narrower process.

Figure 1

Context-Level IDEF0 Model of State Scientific Program Management



Note. Arrows are classified according to the ICOM convention: inputs enter from the left, controls from the top, mechanisms from below, and outputs leave to the right.

The Scenario Model and Its Decision Points

Figure 2 shows the same process as an IDEF3 scenario of 13 units of behavior arranged in two bands. The upper band runs from the scientific idea through the call announcement, application submission, technical expertise, scientific expertise, financial expertise, scientific-technical council review, and ministry-level review. It terminates at an exclusive-OR junction with three outcomes: accepted, revision required, or rejected. The lower band picks up the accepted path and runs through contracting, financing, monitoring, the final report, and transfer to commercialization, ending at a second exclusive-OR junction offering completed, suspended, or terminated.

Three features of this scenario are worth drawing out, and only the first of them is flattering to the model.

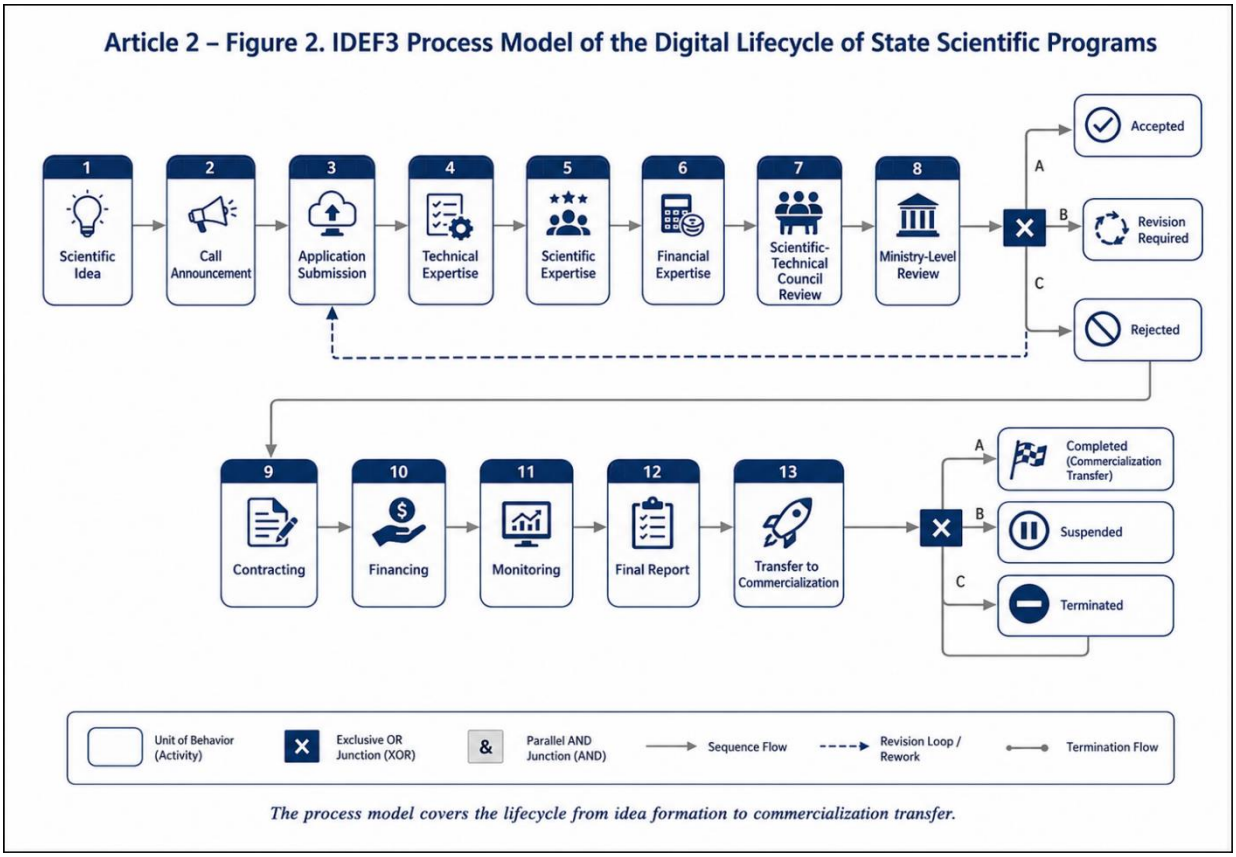
The revision loop returns to unit 3, application submission, rather than to whichever check produced the negative result. That is a design decision with a real cost. A submission returned because of a missing financial annex re-enters the process at the same point as one returned after an unfavorable scientific review, which means the full expertise chain repeats in both cases. A more discriminating loop would return the object to the failed check and revalidate only downstream of it. The current arrangement is simpler to implement and simpler to explain to applicants, and for low return rates the difference in throughput is small; for high return rates it is not.

The second junction sits later in the process than practice would suggest. Suspension and termination are shown as outcomes evaluated after transfer to commercialization, whereas in reality a project is far more likely to be suspended or terminated during monitoring, when a deviation is detected and cannot be corrected. The diagram as drawn describes the formal close-out decision accurately and the practical one poorly.

Third, the three expertise activities are drawn in series. Technical, scientific, and financial review each wait for its predecessor. There is no procedural reason for this ordering - the three reviews consume the same submission and produce independent conclusions - and modeling them as a conjunctive branch would shorten the critical path considerably. The legend of Figure 2 in fact defines a conjunctive junction that the diagram never uses, which is the clearest available evidence that the parallel option was considered and not taken. Treating this as a limitation rather than an oversight, the sequential arrangement is defensible only if the reviews are resource-constrained in a way the model does not currently represent.

Figure 2

IDEF3 Scenario Model of the Digital Lifecycle of State Scientific Programs



Note. Numbered boxes are units of behavior. Junctions marked X are exclusive-OR. The dashed link is the revision loop returning a rejected submission to unit 3.

Control Points and What the System Must Record

The cross-check between the two models produced the mapping in Table 1. Each row identifies a point at which the scenario branches or a status changes, states the condition evaluated there, names the platform mechanism responsible, and specifies what has to be written to the record. The last column is the one that matters for implementation: a branch that changes an object's status without writing an audit entry is a branch that cannot afterwards be defended.

Table 1

Control Points in the Lifecycle and Their Implementation Requirements

Control point	Condition evaluated	Responsible mechanism	Record written
Completeness screening	Required fields and annexes present	Platform validation module	Status change; list of missing items
Compliance screening	Applicant eligible under funding rules	Platform rules engine	Status change; rule invoked
Expert score aggregation	Aggregate score above threshold	Expertise module	Individual and aggregate scores
Score divergence check	Spread between reviewers exceeds tolerance	Expertise module; council	Flag; identity of third reviewer
Council recommendation	Reviews support funding	Scientific-technical council	Protocol; dissenting positions
Ministry confirmation	Recommendation and budget consistent	Ministry-level council	Signed decision; effective date
Monitoring deviation	Reported progress below plan	Monitoring module	Alert; recipient; response deadline
Final report acceptance	Deliverables match contract	Monitoring module; council	Acceptance act; transfer target

Note. Each row corresponds to a transition $(q, \varphi, q') \in \delta$. The tolerance in row 4 and the threshold in row 3 are policy parameters and are not fixed by the model.

From Diagram to Architecture

The descriptors introduced above give each part of the diagram a home in the architecture. The nodes N of the routing triple correspond to the business-logic modules in B: submission, the three expertise modules, council review, contracting, financing, and monitoring. The admissible paths P are the precedence links of Figure 2, which is to say that the diagram is not merely illustrative of the routing but constitutive of it. The rules C encode the conditions Φ of Table 1.

What prevents an application from arriving at contracting without a council decision is therefore not a convention observed by staff but a constraint on the path set.

The layer decomposition Σ locates the remaining responsibilities. The user layer U carries the portals and dashboards through which applicants, reviewers, and council members interact; the data layer D holds the objects O and their statuses; the integration layer X handles identification, electronic signature, and exchange with adjacent government systems, each of its endpoints described by ι(x); and the technical layer T carries the load. The audit trail, notably, is not confined to one layer. It is written by transitions in B, stored in D, and exposed through U, which is why treating auditability as a feature of a single module tends to fail.

Discussion

Why Neither Perspective Suffices Alone

The methodological case for using both notations rests on what each one cannot express. IDEF0 says nothing about order. Reading Figure 1 alone, one cannot tell whether financial screening precedes scientific review or follows it, and one certainly cannot tell what happens when a review is unfavorable - conditional returns, additional expertise, and status changes are simply outside its vocabulary. IDEF3 has the opposite blind spot. Figure 2 shows the sequence clearly and shows nothing at all about which rule authorizes each step or which body is accountable for it. A scenario diagram cannot represent the resource and control structure of the process. Used together, each notation covers the other's silence, and the reconciliation step - checking that every UOB maps to a function and every output to a UOB - is what forced the inconsistencies reported above into the open.

What Formalization Made Visible

Three transitions turned out to be genuinely undetermined by the governing documents. The first is the treatment of divergent expert scores. Regulations specify who reviews and what scale is used, but not what constitutes an unacceptable spread between reviewers or who is entitled to commission an additional review. The second is the response to a monitoring deviation: a deviation is reportable, but the threshold at which it triggers an alert, and the recipient of that alert, are left to practice. The third is the target of the revision loop discussed earlier.

None of these gaps causes visible trouble in a paper-based process, because a human coordinator resolves them case by case and the resolution leaves no trace. The moment the process is expressed as a transition relation δ, each gap becomes a transition with no defined condition, and the system cannot be built until someone with the authority to do so supplies one. This is the most useful thing the modeling exercise produced, and it is a general point rather than a local one: formalizing an administrative process does not primarily automate existing decisions, it identifies decisions that were never actually made.

Where Automation Should Stop

The model draws a deliberate line between computation and confirmation. The platform can validate completeness, apply eligibility rules, distribute submissions to reviewers, compute aggregate scores, detect divergence, and raise monitoring alerts. It does not decide. Council recommendation and ministerial confirmation remain with the bodies that hold legal responsibility for them, and the model records their decisions rather than producing them.

This is not merely deference to legal form. The specific risk in a scored evaluation process is that an aggregate score, computed reliably and displayed prominently, comes to be treated as the decision rather than as one input to it. A council that ratifies a ranking it did not examine has been automated in substance while remaining accountable in form, which is the least defensible of

the available arrangements. Designing the interface so that scores are presented alongside the dissenting positions and the divergence flags, rather than as a single sorted list, is a small choice with a large effect on whether the human confirmation step remains real.

Practical Implications

Two figures are associated with the deployed platform: an acceleration of the introduction of scientific and innovation results into sectors by 15.7%, and an approximately twofold acceleration of commercialization and sectoral application. These should be read with care. They are implementation indicators recorded alongside deployment, not the outcome of a controlled evaluation. No baseline cohort was held back, no adjustment was made for the composition of projects before and after deployment, and the observation period was short relative to the time it takes a research result to reach application. The direction of effect is plausible and consistent with the mechanism proposed here - a shorter administrative path leaves more of a project's calendar available for the work itself - but the magnitude should not be quoted as an effect size.

The operational indicators in Table 2 are on firmer ground, because they measure the duration of administrative steps rather than downstream research outcomes, and those durations are recorded by the system itself. Even so, they compare a digitized process against remembered practice rather than against a measured baseline, and they are offered as a description of what changed rather than as evidence of why.

Table 2

Reported Duration of Administrative Steps Before and After Platform Deployment

Step	Before deployment	After deployment	Basis of the figure
Application submission	20–30 min	5–7 min	Recalled practice vs. system log
Expertise cycle	7–10 days	1–2 days	Recalled practice vs. system log
Monitoring report preparation	Manual, 1–2 days	Generated on request	Recalled practice vs. system log
Paper documents in circulation	Reference level	10–15% of reference level	Estimate
Status visibility to applicant	On request only	Continuous	Feature comparison

Note. Pre-deployment values are reconstructed from practice and were not measured under a common protocol. The final row is a qualitative change and is included for completeness rather than comparison.

Limitations

Four limitations bound what can be claimed. The IDEF0 decomposition stops at the context level, so the internal structure of the expertise and monitoring functions remains unelaborated; a second-level decomposition would probably relocate some of the control points now listed in Table 1. The scenario model has not been checked against event logs. Conformance checking of

the kind described by van der Aalst (2016) would establish whether the process as executed matches the process as drawn, and until that is done the model rests on the modeler's account of the domain. IDEF3 carries no time or resource semantics, so the sequential expertise chain cannot be evaluated against a parallel alternative without the annotation-and-simulation route taken by Whitman et al. (1997). Finally, the model was built around one institutional arrangement. How much of it transfers to a funding body with a different council structure or a different legal basis for its decisions is an open question, and one worth answering, since the value of a reference model lies precisely in its transferability.

Future Work

The obvious next step is to instrument the deployed platform and replay its event log against Figure 2, which would settle the conformance question and quantify how often the revision loop is taken. Decomposing A0 to a second level would let the parallel-expertise variant be specified properly and then simulated. A comparison of the IDEF-based specification with a BPMN rendering of the same lifecycle would also be informative, since it would show how much of the control-versus-mechanism distinction survives translation into an executable notation - and whether what survives is enough.

Conclusion

The lifecycle of a state scientific program can be expressed as a pair of complementary formal models: an IDEF0 function that states what is transformed and under what authority, and an IDEF3 scenario of 13 units of behavior that states in what order and under what conditions. Neither model is executable on its own. Adding a status function, an explicit transition relation, and an audit record at every transition closes that gap and turns the diagrams into a specification a workflow engine can carry out.

The more interesting result was not the model but what building it uncovered. Three points in the lifecycle - the handling of divergent expert scores, the response to monitoring deviations, and the destination of the revision loop - proved to be undetermined by the documents that govern the process. They had been resolved informally, case by case, without leaving a record. A digital platform cannot resolve them informally, which is why the discipline of formal modeling is worth its cost in a domain where the decisions carry legal weight.

REFERENCES

1. Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of business process management* (2nd ed.). Springer. https://doi.org/10.1007/978-3-662-56509-4
2. ISO/IEC/IEEE. (2011). *Systems and software engineering - Architecture description* (ISO/IEC/IEEE 42010:2011). International Organization for Standardization.
3. Janssen, M., & Estevez, E. (2013). Lean government and platform-based governance - Doing more with less. *Government Information Quarterly, 30*(Suppl. 1), S1–S8. https://doi.org/10.1016/j.giq.2012.11.003
4. Kim, C.-H., Weston, R. H., Hodgson, A., & Lee, K.-H. (2003). The complementary use of IDEF and UML modelling approaches. *Computers in Industry, 50*(1), 35–56. https://doi.org/10.1016/S0166-3615(02)00145-8
5. Mayer, R. J., Menzel, C. P., Painter, M. K., deWitte, P. S., Blinn, T., & Perakath, B. (1995). *Information integration for concurrent engineering (IICE): IDEF3 process description capture method report* (Report No. AL-TR-1995-XXXX). Knowledge Based Systems, Inc.; Armstrong Laboratory, Wright-Patterson AFB.

6. National Institute of Standards and Technology. (1993). *Integration definition for function modeling (IDEF0)* (FIPS PUB 183). U.S. Department of Commerce.
7. Object Management Group. (2013). *Business process model and notation (BPMN), version 2.0.2* (formal/2013-12-09).
8. Scheer, A.-W. (2000). *ARIS - Business process modeling* (3rd ed.). Springer. https://doi.org/10.1007/978-3-642-57108-4
9. van der Aalst, W. M. P. (2016). *Process mining: Data science in action* (2nd ed.). Springer. https://doi.org/10.1007/978-3-662-49851-4
10. Weske, M. (2019). *Business process management: Concepts, languages, architectures* (3rd ed.). Springer. https://doi.org/10.1007/978-3-662-59432-2
11. Whitman, L., Huff, B., & Presley, A. (1997). Structured models and dynamic systems analysis: The integration of the IDEF0/IDEF3 modeling methods and discrete event simulation. In S. Andradóttir, K. J. Healy, D. H. Withers, & B. L. Nelson (Eds.), *Proceedings of the 29th conference on winter simulation* (pp. 518–524). IEEE Computer Society. https://doi.org/10.1145/268437.268559