Risk-Driven Timeline Architecture

Integrating Manufacturing Complexity and Readiness Assessment with Monte Carlo Schedule Analysis

Abstract

Traditional project risk management relies heavily on Impact × Probability (I×P) scoring to identify and prioritize threats. While intuitive, this approach fails to account for the architectural properties of project schedules—the network of dependencies, float, critical paths, and integration points that determine how delays propagate through a timeline. More fundamentally, it provides no principled basis for assigning probability and impact scores to manufacturing tasks whose uncertainty derives from technical complexity.

This document presents a hybrid methodology that transforms I×P scoring from a static prioritization tool into a dynamic engine for Monte Carlo schedule simulation. By grounding probability scores in Manufacturing Technical Level (MTL) complexity assessment and structuring schedule milestones around Manufacturing Readiness Level (MRL) category transitions, the methodology connects subjective risk judgment to both technical reality and schedule architecture. The result is a quantified understanding of schedule risk that reveals which elements actually drive program completion dates.

Introduction

Every project manager has encountered the familiar ritual of risk assessment: gather the team, brainstorm potential threats, assign probability and impact scores, multiply them together, and sort the resulting list. The Impact × Probability matrix has become so ubiquitous in project management that it often goes unquestioned. Yet beneath its apparent simplicity lies a fundamental limitation that undermines its utility for schedule risk management: the method knows nothing about the architecture of the timeline it purports to protect.

A project schedule is not merely a list of tasks with dates—it is a directed network with path lengths, slack, concurrency, resource constraints, merge points, and hard calendar gates. In such a network, risk is not evenly distributed. Some tasks, even those with high uncertainty, cannot move the finish date because they possess adequate float. Other tasks with seemingly modest uncertainty can devastate the timeline because they sit on critical integration sequences. The I×P method, by collapsing all this structural complexity into a single scalar score, creates a dangerous illusion: it suggests that risk can be understood and managed without reference to the network topology that determines whether a delay actually matters.

> *The same structural blindness afflicts how organizations assess manufacturing readiness. A program can report 80% readiness against milestones on both a routine component and a frontier manufacturing challenge—yet the risk profiles differ dramatically. Readiness metrics without complexity context are as misleading as I×P scores without schedule architecture.*

This document presents a methodology that addresses both forms of structural blindness simultaneously. By treating probability and impact as parameters governed by Manufacturing Technical Level (MTL) complexity scores, and by anchoring schedule milestones to Manufacturing Readiness Level (MRL) category transitions, we create a closed-loop system where complexity assessment informs distribution parameters, readiness assessment defines phase gates, and Monte Carlo simulation reveals which elements drive program-level risk.

The Three Integrated Frameworks

The methodology integrates three complementary frameworks, each addressing a distinct dimension of program risk:

Manufacturing Technical Level (MTL) answers: How inherently difficult is this manufacturing challenge? MTL is a complexity assessment—it characterizes the terrain, not progress along the path. A five-level scale captures the range from routine industrialization (MTL 1) through frontier manufacturing where process science is still being established (MTL 5).

Manufacturing Readiness Levels (MRL) answer: How far has manufacturing maturation progressed? A four-category structure groups the traditional ten MRL levels into functional phases: Feasibility Validated (MRL 1-3), Prototype Capability Demonstrated (MRL 4-6), Production System Validated (MRL 7-8), and Production Rates Demonstrated (MRL 9-10). Each category answers a distinct question about manufacturing maturation.

Monte Carlo Schedule Simulation answers: Given uncertainty in task durations, what is the probability distribution of project completion? By sampling from duration distributions thousands of times and propagating results through the dependency network, Monte Carlo reveals how uncertainty flows through schedule architecture to influence completion dates.

The integration works as follows: MTL complexity scores determine probability and impact parameters for each task. These parameters define duration distributions. MRL category transitions provide schedule milestones. Monte Carlo simulation propagates distributions through the milestone network. Criticality Index and Schedule Sensitivity Index identify which tasks actually drive program completion risk.

The Fundamental Problem with Traditional I×P Scoring

A project schedule operates like a metropolitan rail network where multiple lines must deliver workers to a central terminal by the start of business. The system doesn't depend on a single route—it requires the express from the northern suburbs, the crosstown connector, the airport link, and the southern local all functioning together. Each line has its own capacity, its own infrastructure, its own vulnerabilities. The city reaches working capacity only when the last constrained line has delivered its passengers.

I×P scoring is like a transit engineer inspecting each switch, signal, and railcar throughout the network, asking: What's the probability this component fails? How severe would the local disruption be? These are legitimate questions about equipment reliability. But I×P cannot answer the network question that actually matters: If this signal fails, does it delay the overall worker commute?

Architectural Blindness

The core issue is that Probability × Impact produces a number that ignores whether a task is structurally capable of delaying the project. This explains why a high-ranked risk might prove harmless while a low-ranked risk turns catastrophic. The structural properties that determine actual schedule impact—float, critical path membership, dependency density, resource contention—are invisible to the traditional approach.

I×P scoring treats risk as independent from the network structure of the project. A delay on a non-critical activity may have zero effect on the finish date, yet receives the same treatment as a delay on the critical path. The model fails to incorporate float and lag buffers—a "high impact" risk on a task with forty days of free float may have no real timeline consequence, but I×P would still rank it as dangerous.

The Garbage-In Problem

Probability and impact values present themselves as numbers, lending a false sense of rigor to what are fundamentally subjective judgments. Probability estimates for schedule disruptions are notoriously unreliable in complex environments—vendor readiness is non-linear, regulatory failures are binary, and first-article builds contain unknown unknowns. Teams anchor on non-analogous past projects, yield to optimistic internal narratives, and succumb to cultural pressures that make it difficult to acknowledge high slip probabilities.

What's missing is a principled basis for these judgments. When an assessor assigns P=3 and I=4 to a supplier qualification task, what grounds that assessment? Without a structured framework connecting probability to observable complexity characteristics, scoring devolves into opinion aggregation.

This is precisely where MTL complexity assessment adds value. Rather than asking "what's your gut feel about probability," the methodology asks "what is the Manufacturing Technical Level of this task"—a question with structured scoring criteria, dimension-specific anchors, and observable characteristics that multiple assessors can evaluate consistently.

MTL Complexity as the Foundation for Distribution Parameters

The key conceptual shift is to treat Probability and Impact not as final scores but as parameters that govern the shape and scale of task duration distributions. MTL complexity assessment provides the principled basis for assigning these parameters.

Conceptual Framework

Probability as Duration Volatility

The probability dimension captures how uncertain a task's duration is—not merely whether a discrete risk event might occur, but how volatile the task's completion timeline is expected to be. Higher MTL complexity directly correlates with duration volatility because higher complexity means less characterized process behavior, narrower operating windows, and more potential for unexpected variation.

A task with Probability 1 has highly predictable duration; historical data supports confident estimates. A task with Probability 5 operates in genuinely uncertain territory where duration estimates are essentially educated guesses subject to substantial revision as learning occurs.

Impact as Consequence Magnitude

The impact dimension captures the magnitude of schedule exposure if duration uncertainty materializes adversely. MTL complexity informs impact because higher complexity typically means fewer recovery options, more constrained supplier alternatives, greater downstream dependency, and higher likelihood that delays cascade rather than absorb within available float.

A task with Impact 1 affects only itself; disruption is contained and easily recovered. A task with Impact 5 creates program-level or enterprise-level consequences where delays threaten fundamental program objectives or market positioning.

MTL to Probability Score Mapping

MTL complexity directly informs probability assessment because higher complexity means less characterized process behavior, narrower windows, and more potential for unexpected variation.

MTL	P Score	Rationale
MTL 1	P = 1	Routine processes with well-characterized behavior; uncertainty is bounded and predictable
MTL 2	P = 2	Known processes with targeted customization; modest parameter uncertainty from window refinement
MTL 3	P = 3	Multiple complexity factors overlap; process windows not fully characterized; DOE required
MTL 4	P = 4	Process development required; fundamental uncertainty about capability; specialist knowledge needed
MTL 5	P = 5	Frontier manufacturing; process science still being established; high likelihood of discovery work

MTL to Impact Score Mapping

MTL complexity informs impact because higher complexity typically means fewer recovery options, more constrained supplier alternatives, greater downstream dependency, and higher likelihood that delays cascade rather than absorb.

MTL	I Score	Rationale
MTL 1	I = 1-2	Delays bounded; workarounds exist; many suppliers can execute; recovery straightforward
MTL 2	I = 2-3	Delays recoverable with effort; limited supplier options add modest exposure
MTL 3	I = 3	Delays affect downstream integration; limited recovery options; rework cycles likely
MTL 4	I = 4	Delays cascade through dependent activities; few alternatives; capital equipment at risk
MTL 5	I = 5	Delays potentially phase-resetting; design feedback loops likely; program-level consequences

Probability Rubric: Detailed Anchor Behaviors

This section provides detailed scoring anchors for the probability dimension. The score reflects inherent task characteristics grounded in MTL complexity, not current progress or team optimism. Discrete scores (1–5) map to continuous parameter values (0.0–1.0) for distribution calculation.

Probability 1 — Rare Disruption (Parameter: 0.0)

Definition: The task uses thoroughly proven methods with extensive organizational history. Process windows are wide relative to normal variation. Similar tasks have been executed dozens or hundreds of times with consistent outcomes. Disruptions occur only under exceptional circumstances—equipment failure, gross operator error, or external shocks.

Anchor Behaviors:

- Historically task completes on-time or ahead of time
- Schedule estimates based on this task type have proven reliable within ±5%
- No process-related surprises in the last twenty similar executions

Representative Examples:

- Ordering off-the-shelf fasteners from established suppliers
- Performing common machining operations on mature equipment with experienced operators
- Assembling well-documented subassemblies using proven work instructions on a mature production line

Probability 2 — Occasional Disruption (Parameter: 0.25)

Definition: The task is well-understood but includes parameters or conditions that periodically cause issues. The organization has strong experience, but some runs encounter problems requiring modest correction. Disruptions are annoying rather than surprising when they occur.

Anchor Behaviors:

- Historically there have been few noted schedule slips or very minor timing slips
- Schedule estimates typically reliable within ±10–15%
- Task seldom requires meaningful troubleshooting or rework

Representative Examples:

- New weld fixturing with moderately tight tolerances needed
- Trialing a new live tool or new carbide insert on an existing CNC lathe
- Supplier qualification for capable but new vendor required

Probability 3 — Periodic Disruption Expected (Parameter: 0.50)

Definition: The task involves complexity factors where problems are a normal part of execution rather than exceptions. The team expects to encounter issues and plans accordingly. Success requires active problem-solving, not just execution discipline.

Anchor Behaviors:

- Limited historical data exists on the task execution
- Schedule estimates carry ±20–30% uncertainty
- There are "known unknowns"

Representative Examples:

- First article production on MTL 3 components

- Process window characterization through Design of Experiments
- Multi-supplier coordination for integrated assemblies
- New equipment installation and commissioning for moderately complex systems

Probability 4 — Frequent Disruption Likely (Parameter: 0.75)

Definition: The task operates near the boundary of current capability. Problems are more likely than not on any given attempt. Success requires iteration, learning, and adaptation. The team should assume multiple cycles will be needed.

Anchor Behaviors:

- The task is "monolithic" in nature—highly irreducible
- Task schedule estimates are long (12 months plus)
- The task execution will require rework loops, redesign, or potentially restart

Representative Examples:

- Novel process development and validation
- Qualification of suppliers operating at the edge of their capability envelope
- Integration of new product technologies into existing products
- First production runs on MTL 4 components

Probability 5 — Disruption is the Default State (Parameter: 1.0)

Definition: The task ventures into genuinely uncertain territory where the path to success is not fully defined. Initial attempts are expected to fail or fall short. The task is fundamentally a learning exercise disguised as a project activity.

Anchor Behaviors:

- No reliable historical data exists for this task type
- Schedule estimates are essentially targets with ±100% or greater uncertainty
- Success on first attempt would be genuinely surprising

Representative Examples:

- Frontier process scale-up from laboratory to production environment
- First builds on MTL 5 components
- Development of custom automation for novel applications

Impact Rubric: Detailed Anchor Behaviors

This section provides detailed scoring anchors for the impact dimension. Impact reflects both direct effects (cost, time lost on the specific task) and systemic effects (downstream delays, integration failures, program-level risk exposure). Discrete scores (1–5) map to continuous parameter values (0.0–1.0) for distribution calculation.

Impact 1 — Contained Disruption (Parameter: 0.0)

Definition: Task disruption affects only the task itself. Recovery is straightforward and inexpensive. No downstream activities are meaningfully affected. Schedule buffer easily absorbs the delay.

Anchor Behaviors:

- Recovery cost is insignificant to the task budget
- Schedule slip is minor—less than one week
- No other tasks are delayed as a result

Representative Examples:

- Rework on non-critical cosmetic components
- Minor supplier delivery delays with adequate inventory buffer
- Documentation corrections or administrative rework

Impact 2 — Local Disruption with Modest Ripple (Parameter: 0.25)

Definition: Task disruption causes direct cost and schedule impact, plus minor effects on immediately adjacent activities. Recovery requires management attention but not escalation. Program schedule absorbs the impact without milestone movement.

Anchor Behaviors:

- Recovery cost has minimal impact to the task budget
- Schedule slip is minimal and recoverable—one to three weeks
- One or two downstream tasks experience minor delays
- Internal visibility only; no customer impact

Representative Examples:

- Rework or deviation on non-critical components
- Tooling modifications requiring re-validation
- Numerous FRACAS items discovered during pilot builds

Impact 3 — Significant Disruption Affecting Multiple Workstreams (Parameter: 0.50)

Definition: Task disruption propagates to multiple downstream activities or parallel workstreams. Recovery requires cross-functional coordination and resource reallocation. Program schedule pressure increases meaningfully, though launch date may still be achievable with intervention.

Anchor Behaviors:

- Recovery cost is 5–15% of task budget requiring change control approval
- Multiple downstream tasks are affected across workstreams
- Program milestone dates under pressure; management escalation required

Representative Examples:

- Major tooling rework or redesign
- Supplier qualification failure requiring alternative sourcing

- Process capability shortfall requiring control plan redesign

Impact 4 — Major Disruption Threatening Program Objectives (Parameter: 0.75)

Definition: Task disruption creates substantial risk to program cost, schedule, or technical objectives. Recovery requires significant investment and potentially trades against other program goals. Launch date movement becomes likely without aggressive intervention.

Anchor Behaviors:

- Recovery cost is 15–30% of task budget or triggers capital re-approval
- Executive visibility required; customer or market timing implications emerge
- Significant schedule slip; critical path is directly affected

Representative Examples:

- Fundamental process failure requiring technology pivot
- Key supplier fails to meet requirements
- Integration failures revealing design-manufacturing incompatibility
- First Pass Yield collapse during production ramp

Impact 5 — Program-Level or Enterprise-Level Disruption (Parameter: 1.0)

Definition: Task disruption threatens program viability or creates enterprise-level consequences. Recovery may not be possible within original program constraints. Fundamental re-evaluation of scope, schedule, or investment is required.

Anchor Behaviors:

- Recovery cost exceeds task budget requiring board-level approval
- Schedule slip misses market window entirely or is inestimable
- Regulatory, safety, or intellectual property issues emerge
- Program cancellation becomes a serious option under consideration

Representative Examples:

- Safety-critical process failure requiring product redesign
- Single-source supplier failure with no qualified alternative
- Intellectual property infringement discovered late in development

Distribution Parameter Calculation

The distribution shape determines how samples are distributed between the bounds established in Step 2. Both late-fail and early-fail modes use the BetaPERT distribution with identical lambda and shape parameter calculations. The behavioral difference between modes—late-fail's heavy right tail versus early-fail's more balanced shape—emerges entirely from the different bounds calculated in Step 2, not from different distribution families or shape formulas.

BetaPERT Distribution

The BetaPERT distribution allows lambda to control peakedness independent of the bounds. Higher lambda concentrates samples near the mode; lower lambda spreads samples toward the tails. This distribution is used for both late-fail and early-fail modes—the difference between modes lies in the bounds calculated in Step 2, not the distribution family.

Lambda Calculation

Lambda is determined solely by the probability score and applies identically to both fail modes:

$$\lambda = 6 - (4 \times P)$$

P Score	Lambda (λ)	Effect
P1 (P = 0.00)	6.0	Very peaked—samples cluster tightly around mode
P2 (P = 0.25)	5.0	Peaked—most samples near mode with modest spread
P3 (P = 0.50)	4.0	Standard PERT—classic weighting assumption
P4 (P = 0.75)	3.0	Flatter—increased sampling from tails
P5 (P = 1.00)	2.0	Flattest—substantial sampling across full range

Shape Parameter Calculation

Given a = Optimistic, b = Mode, c = Pessimistic (from Step 2):

- **Mean:** $\mu = (a + \lambda \times b + c) / (\lambda + 2)$
- **Alpha:** $\alpha = ((\mu - a) \times (2b - a - c)) / ((b - \mu) \times (c - a))$
- **Beta:** $\beta = \alpha \times (c - \mu) / (\mu - a)$

Probability Density Function

$$f(x) = [(x - a)^{(\alpha-1)} \times (c - x)^{(\beta-1)}] / [B(\alpha, \beta) \times (c - a)^{(\alpha+\beta-1)}]$$

Where $B(\alpha, \beta)$ is the Beta function: $B(\alpha, \beta) = \Gamma(\alpha)\Gamma(\beta) / \Gamma(\alpha+\beta)$

Late-Fail Mode Example

Late-fail bounds use the asymmetric coefficients (0.2 / 1.2) that produce a heavy right tail, reflecting problems that accumulate and compound toward the end of task execution.

P3/I3 Late-Fail Task: $a = 91d$, $b = 100d$, $c = 154d$, $\lambda = 4$

$\mu = (91 + 4 \times 100 + 154) / 6 = 107.5$ days

$\alpha = ((107.5 - 91) \times (200 - 91 - 154)) / ((100 - 107.5) \times (154 - 91)) = 1.57$

$\beta = 1.57 \times (154 - 107.5) / (107.5 - 91) = 4.43$

The resulting $\alpha < \beta$ produces a right-skewed distribution. The pessimistic tail extends 54 days beyond the mode while the optimistic side compresses only 9 days—a 6:1 asymmetry reflecting the compounding delay behavior of late-fail tasks.

Early-Fail Mode Example

Early-fail bounds use the more symmetric coefficients (0.4 / 0.6) that produce balanced tails, reflecting problems that surface quickly and allow early course correction.

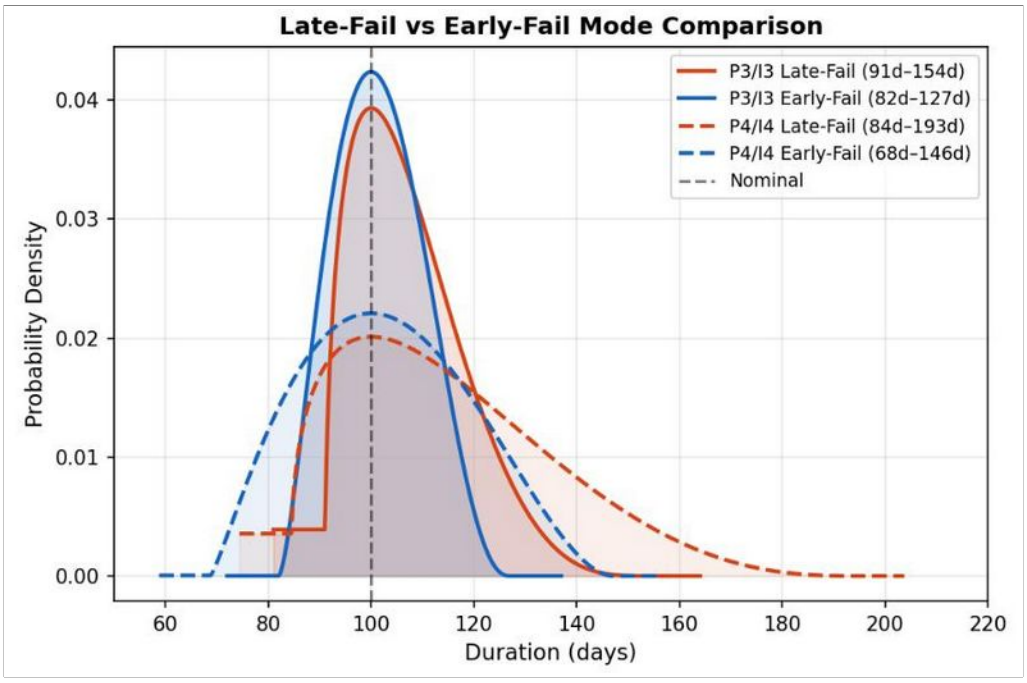
P3/I3 Early-Fail Task: $a = 82d$, $b = 100d$, $c = 127d$, $\lambda = 4$

$\mu = (82 + 4 \times 100 + 127) / 6 = 101.5$ days

$\alpha = ((101.5 - 82) \times (200 - 82 - 127)) / ((100 - 101.5) \times (127 - 82)) = 2.58$

$\beta = 2.58 \times (127 - 101.5) / (101.5 - 82) = 3.37$

The resulting α and β are closer in value, producing a more symmetric distribution. The optimistic side extends 18 days below the mode while the pessimistic side extends 27 days —a 1.5:1 asymmetry. Tasks can finish meaningfully early when problems are discovered and resolved quickly.



Conclusion: Architecture-Aware Risk Management

The Impact × Probability framework need not be discarded—but it must be evolved. When probability and impact scores are grounded in MTL complexity assessment, when those scores govern duration distributions with appropriate early-fail or late-fail characteristics, when MRL category transitions provide schedule milestones, and when Monte Carlo simulation propagates uncertainty through the network, the method transcends its structural limitations. It becomes capable of distinguishing between risks that matter architecturally and risks that, despite high subjective scores, cannot influence project completion.

The methodology presented here integrates three complementary frameworks into a coherent system:

MTL complexity assessment provides the principled basis for probability and impact scoring, connecting subjective judgment to observable manufacturing characteristics and eliminating the "garbage-in" problem of traditional I×P methods.

MRL category structure provides natural schedule milestones that anchor the Monte Carlo model, enabling domain-specific analysis across Design, Supplier, and Manufacturing critical paths.

Monte Carlo simulation propagates MTL-informed distributions through MRL-structured milestones, generating Criticality Index and Schedule Sensitivity Index metrics that identify true schedule risk drivers with mathematical precision.

Together, these elements constitute a risk-driven timeline architecture approach that acknowledges complexity, embraces uncertainty, and generates actionable insight grounded in both the physics and mathematics of schedules. For practitioners navigating the demands of modern product development—concurrent engineering, distributed supply chains, integrated validation campaigns—this methodology provides a substantially more realistic representation of timeline vulnerability than traditional qualitative methods can achieve.

The goal is not perfect prediction—uncertainty, by definition, resists precise forecasting. The goal is better decisions: knowing where to focus attention, how to allocate contingency, when to escalate concerns, and which mitigations will actually move the needle on project completion. Architecture-aware risk management delivers that knowledge through a rigorous, transparent, and defensible analytical framework.

Organizations that adopt this methodology often report a fundamental shift in how schedule risk conversations occur. Rather than debating whether a particular risk "feels" concerning or whether its I×P score warrants attention, teams can point to CI and SSI metrics grounded in MTL complexity and MRL milestone structure. Executive reviews become data-driven discussions about probability distributions and critical path dynamics rather than exercises in consensus opinion. Project managers gain credibility by presenting quantified risk profiles that acknowledge uncertainty while providing actionable guidance.

The transition requires investment—in MTL assessment capability, MRL milestone definition, simulation tools, and team training. But for organizations managing complex programs where schedule performance is critical to business success, the return on that investment manifests in earlier risk detection, more effective mitigation, and ultimately more reliable project delivery. In an era where product development cycles continue to compress while technical complexity expands, architecture-aware schedule risk management is not merely advantageous—it is increasingly essential.

ADDENDUM

Fail Mode Assignment by MTL

Early-Fail Behavior: MTL 1-2 Tasks

Early-fail tasks reveal bad news quickly. Most of the uncertainty is front-loaded—if the task is going to slip, you discover it early in the execution window. These tasks typically have narrow, symmetric or modestly skewed distributions with light tails. Risk is often detectable through early prototypes, early supplier feedback, or early engineering signals.

MTL 1-2 tasks exhibit early-fail behavior because their processes are well-characterized. Material procurement with stable suppliers reveals lead time problems within days of order placement. Internal design tasks with rapid iteration cycles expose missing inputs or ambiguous requirements immediately. Fixture builds in mature machine shops surface interference issues early in fabrication, not at final acceptance.

Late-Fail Behavior: MTL 4-5 Tasks

Late-fail tasks are the true schedule killers in complex programs. Work appears on track until near the end—often at integration, test, or validation milestones—when failure suddenly emerges. Duration uncertainty is heavily right-skewed, with tight clustering around the most-likely estimate but explosive tails at high percentiles.

MTL 4-5 tasks exhibit late-fail behavior precisely because their complexity characteristics—narrow process windows, novel materials, limited supplier capability, frontier manufacturing science—create risks that remain hidden until the system is assembled, tested, or validated. The problems aren't visible until you try to do the hard thing at scale.

Supplier readiness on high-MTL components exemplifies late-fail behavior. The supplier reports being "on track" throughout the lead time, but untested processes or understaffed capacity only reveal cracks at pre-shipment inspection, first-article runs, or acceptance testing. System integration events follow the same pattern: everything looks green until the whole system is assembled, when interface mismatches or timing issues appear.

MTL 3: Transitional Behavior

MTL 3 tasks occupy the threshold where fail-mode assignment requires judgment rather than automatic mapping. These tasks combine known and partially novel processes—the individual steps are familiar, but their combination or sequencing is less common. This mixed character means failure timing can go either way.

Assign Early-Fail when:

- Task involves familiar processes in new combination, where interface issues will surface during initial integration attempts
- Early prototyping or pilot builds are planned that will expose problems quickly
- Learning cycles are short—if something fails, you'll know within days or weeks, not months

Assign Late-Fail when:

- Task involves process window characterization (DOE) where capability limits only emerge after extended testing
- Success depends on cumulative factors—yield, cycle time stability, measurement system adequacy—that reveal themselves gradually
- Validation occurs at the end, with limited intermediate checkpoints

Default recommendation: When uncertain, assign MTL 3 tasks to late-fail mode. The asymmetric cost of error favors conservatism—underestimating a late-fail task is more damaging than overestimating an early-fail task.

Risk Prioritization and Intervention Targeting

The product of Probability × Impact discrete scores provides a composite risk score that identifies candidates for proactive intervention. Tasks with high composite scores warrant restructuring through 'shift and diffuse' strategies—decomposing monolithic high-risk tasks into staged phases with earlier failure discovery opportunities.

P × I Risk Matrix

P \ I →	I1	I2	I3	I4	I5
P1	1	2	3	4	5
P2	2	4	6	8	10
P3	3	6	9	12	15
P4	4	8	12	16	20
P5	5	10	15	20	25

Risk Zone Interpretation:

- **Green (1–4):** Standard monitoring. Task proceeds with normal governance.
- **Yellow (5–6):** Enhanced attention. Task warrants explicit risk ownership and contingency identification.
- **Orange (8–12):** Active intervention. Task is a candidate for restructuring, early prototyping, or parallel path development.
- **Red (15–25):** Mandatory restructuring. Task must be decomposed using shift-and-diffuse strategies before proceeding.