EPOCH SRA
SCHEDULE RISK ANALYSIS

The Quantitative Schedule Risk Analysis Handbook

Edition 1 · August 2026

epochsra.com/resources/handbook – free online, corrections welcome at contact@epochsra.com

Contents

1. Why single dates lie
2. Merge bias: the arithmetic of being late
3. Reading a distribution: what P80 promises and what it doesn't
4. Is your schedule analyzable?
5. From risk register to simulation
6. Sensitivity: measured, not assumed
7. Calibration: whose numbers, and why trust them
8. Cost through the schedule
9. Running an SRA your review board believes
10. Choosing tools: the questions that matter

Chapter 1 – Why single dates lie

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

Every program we have worked on had two schedules.

The first is the official one: a network of tasks with single dates, reviewed, baselined, and signed. It says the payload integrates in May and the program finishes on 02 December. Full stop.

The second schedule lives in the project manager's head. It says: "if the optics slip again, May is gone." It says: "the test chamber is oversubscribed – we might wait six weeks for a slot." It says, when asked privately: "realistically? February."

The second schedule is usually closer to the truth, and it is stored in exactly one place: one person's intuition. It cannot be reviewed, cannot be handed over, cannot be argued with, and leaves the building when that person does.

Quantitative Schedule Risk Analysis is, at its core, the act of moving the second schedule out of someone's head and into arithmetic. Everything else – Monte Carlo simulation, percentiles, tornado charts – is mechanism. The point is honesty about what you do not know, made precise enough to plan with.

Where the single date comes from

A deterministic schedule assigns each task one duration. Where does that number come from? Usually from an engineer who was asked "how long will this take?" – a question that quietly means "what number can you commit to?" Human beings answer that question in two well-documented ways: with optimism (the plan assumes the design review passes the first time), or with hidden padding (the estimate protects its owner against being blamed). Both distortions are invisible in the final plan, because a single number carries no record of the uncertainty it summarizes.

The result is a schedule that is *precise* – dates to the day – and systematically *inaccurate*. Precision without accuracy is worse than useless: it invites confidence.

Tasks finish late more than they finish early

Duration uncertainty is not symmetric. A task estimated at 30 days can plausibly take 45 or 60 – a failed test, a late delivery, a requirement change – but it can rarely take 10. The distribution of real outcomes has a short left side and a long right tail.

A single-point estimate silently averages that distribution, or worse, reports its most optimistic defensible value. When many such estimates are chained into a network, the errors do not cancel – they compound, and always in the same direction. This is why programs are late far more often than they are early, even when every individual estimate seemed reasonable.

What this chapter asks you to accept

Only two things. First: your official schedule is a set of point estimates drawn from distributions nobody wrote down. Second: the information needed to write those distributions down already exists in your organization – in the risk register, in engineers' hedged answers, in the last program's actuals. QSRA is the discipline of connecting them.

The next chapter shows why chaining even *good* estimates into a network produces systematic lateness – an arithmetic effect called [merge bias](/guide.html#merge-bias) that no deterministic schedule can see.

> **How we do it in Epoch SRA** – The tool is built around the premise of this chapter: your file's dates are read as stored, and uncertainty enters explicitly, from your risk register and duration inputs, never from silent assumptions. The gap between the two schedules becomes a computed distribution instead of a private worry.

--- In practice: a useful first exercise on any program is to ask, for the three most uncertain tasks, "what would this duration be if it went badly – and how likely is that?" If the answers surprise the official schedule, the second schedule exists, and it is worth formalizing.

Chapter 2 – Merge bias: the arithmetic of being late

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

Here is the whole chapter in three sentences. Two parallel paths, each 90% likely to finish by March, give the point where they merge an 81% chance of holding March. Add a third parallel path and it drops to 73%. Nobody decided to be late – the network structure did it.

The arithmetic

For a milestone to start on time, *every* path feeding it must finish on time. If the paths are independent, the probability that all of them hold is the product of their individual probabilities:

- Two paths at 90%: 0.9 × 0.9 = **81%**
- Three paths at 90%: 0.9 × 0.9 × 0.9 = **73%**
- Five paths at 90%: **59%**

Each path, examined alone, is green. The merge point is quietly amber. This effect – [merge bias](/guide.html#merge-bias), sometimes called merge-node bias – appears at every point where parallel work converges: integration events, design reviews, test readiness gates. Which is to say: at exactly the moments programs care about most.

Why deterministic schedules cannot see it

A deterministic schedule computes each merge point's date as the maximum of its predecessors' single dates. If all three paths "finish 15 March," the merge holds 15 March, with no hint that the *probability* of that date just fell to 73%. The critical path method is blind to merge bias by construction: it operates on one realization of the schedule, and merge bias only exists *across* realizations.

This is the honest answer to a question every program office has asked: how can every subsystem report green while the program is late? Both statements are true simultaneously. Subsystem status is a statement about individual paths; program lateness is a statement about their product.

Why simulation catches it for free

A Monte Carlo schedule simulation does not average paths – it lives them. In each iteration, every task draws a duration from its distribution, the network is computed, and the merge point lands wherever that particular future puts it. Across thousands of iterations, the merge point's *distribution* emerges, and merge bias is simply present in it – not modeled specially,

just no longer ignored. The gap between the deterministic merge date and the simulated P50 at that node is merge bias, measured.

Two practical consequences follow. First, merge bias grows with parallelism – so accelerating a program by parallelizing work (the standard recovery move) silently increases the very effect that makes programs late. There may be good reasons to parallelize anyway; the point is to price it. Second, the tasks feeding a merge deserve scrutiny proportional to the merge's fan-in, not to their individual risk scores – a modest task on the fifth parallel path can matter more than a risky task on a chain of one.

> **How we do it in Epoch SRA** – Merge bias needs no special feature: each of 20,000 iterations computes the full network, so every merge point's probability erosion is simply present in its distribution. The measured gap between a milestone's deterministic date and its simulated P50 is merge bias, made visible per milestone.

--- In practice: count the predecessors of your next major integration milestone. Raise 0.9 to that power. If the result is uncomfortable, that discomfort is information no status meeting has given you.

Chapter 3 – Reading a distribution: what P80 promises and what it doesn't

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA](https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

A schedule risk analysis does not produce a date. It produces a *distribution* of dates – typically from tens of thousands of simulated program futures – and then summarizes it with percentiles. Reading those percentiles correctly is the difference between using the analysis and being decorated by it.

The definitions, precisely

P50: the median. In half of the simulated futures the program finishes by this date; in half it does not. A P50 commitment is a coin flip, knowingly taken.

P80: the 80th percentile. In 80% of simulated futures, the program is done by this date. Committing at P80 means accepting a one-in-five chance of missing – explicitly, on the record, instead of implicitly and by surprise.

P-anything is a statement about the model's futures, not a promise about the world. If the inputs are optimistic, the P80 is optimistic. Percentiles inherit the honesty of what feeds them – a theme Chapter 7 (calibration) treats fully.

The gap is the message

The distance between P50 and P80 measures how much the program does not yet know. A program with P50 in December and P80 in mid-February carries roughly ten weeks of uncertainty in that band alone. Two programs can share a P50 and differ enormously in P80 – the second number is what distinguishes a tight plan from a hopeful one, and it is the number single-date scheduling never produces.

When someone presents one date with no percentile attached, it is fair – and clarifying – to ask which percentile it is. In our experience the honest answer, once computed, is usually nearer P20 than P50: the official date is a date the program will *probably miss*, presented as a plan.

Distributions have shape, and the shape talks

Two features of a finish-date histogram deserve attention before any percentile is quoted.

The left edge. If a visible fraction of simulations lands exactly on the deterministic finish, that spike is the probability that *nothing goes wrong* – on a program with discrete risks, it

approximately equals the product of the risks' non-occurrence probabilities. A tall spike is not an error; it is the model saying your uncertainty is dominated by discrete events rather than duration noise. It can be checked with one multiplication, and auditable arithmetic is the cheapest trust a model can buy.

The right tail. Its length reflects compounding – risks landing on risks, [merge bias](/guide.html#merge-bias) amplifying both. Deep-tail percentiles (P90, P95) are the least certain numbers the analysis produces, because tails are where model assumptions matter most and calibration evidence is thinnest. A tool that labels which of its percentiles rest on evidence and which are model extrapolation is telling you something most tools do not.

Choosing a commitment percentile

There is no universally correct choice; there is a correct *conversation*. Planning internally at P50 and committing externally at P80 is a common and defensible convention. Safety-critical or contractually severe milestones may justify P90. What is not defensible is committing at an unknown percentile – which is precisely what single-date scheduling does, every time.

> **How we do it in Epoch SRA** – Every percentile in the output carries a provenance label: field-calibrated (P50/P80, backtested against historical program actuals) or model estimate (the tails). On our demo program the left-edge spike is 38.6% of 20,000 iterations – and the product of the open risks' non-occurrence probabilities is 0.378. One multiplication audits the model.

--- In practice: at your next schedule review, ask for two numbers instead of one — the P50 and the P80 — and put the gap between them on the slide. The size of that gap, and whether it shrinks review over review, tells you more about program health than either date alone.

Chapter 4 – Is your schedule analyzable?

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

Before running any simulation, ask a harder question: is your schedule a network, or a drawing of one?

The distinction is simple to test. In a network, dates are *consequences*: task B starts when task A finishes, so changing A moves B. In a drawing, dates are *decorations*: bars placed by hand where someone wanted them, connected by few or no links. Both look identical on a Gantt chart. Only one can carry a risk analysis.

The test

Pick a task in the middle of your plan. Delay its finish by a month. If nothing downstream moves, you have found painted bars. Repeat with an early task feeding your main integration milestone. If the milestone holds its date regardless, the milestone is not scheduled – it is asserted.

We have run this test on real programs. One 50-task plan we imported had exactly one task on the critical path – a level-of-effort management bar spanning the whole program – while every technical task floated with hundreds of days of slack. Reviews sat at the project start date, disconnected from the phases they were meant to gate. The plan had been maintained for years. It was a drawing.

Why simulation amplifies the problem

A Monte Carlo run on a painted schedule executes without complaint and produces precise percentiles. They mean nothing. Risk propagates through links; where there are no links, a stretched task stretches alone and the program finish never feels it. The result is a suspiciously tight distribution and false comfort, delivered with three decimal places.

This is why schedule quality checks belong *before* probability, not after. The discipline has names: the [DCMA 14-Point Assessment](/guide.html#dcma-14-point), the [GAO Schedule Assessment Guide](/guide.html#gao-schedule-guide), NASA's schedule management practices. Their shared core is a handful of questions any tool or analyst can ask: Do tasks have predecessors and successors? Is float plausible, or are hundreds of days of slack hiding missing logic? Are hard date constraints doing a link's job? Do relationship types and lags reflect real dependencies, or workarounds?

What the checks buy you

Not compliance – information. A missing-predecessor count tells you which fraction of your plan is connected to reality. A float distribution with a long tail of 500-day slack tells you where logic was never entered. A constraint doing a link's work tells you a dependency exists in someone's head but not in the file. Each finding is repairable in minutes in your scheduling tool; the checks are a repair map, not a verdict.

An honest import report belongs in the same category. When a schedule moves between tools, some constructs may not translate. A tool that lists what it could not carry – by name, with the reason – is giving you the same gift as the quality checks: the difference between what everyone assumes the plan contains and what it actually contains. In our experience that difference, seen once, changes how a team maintains its schedule permanently.

> **How we do it in Epoch SRA** – Every Compute runs DCMA-style quality checks automatically (8 of the 14 points as of this edition; the implemented and not-yet-implemented lists are published), and the MS Project import reports every construct it cannot translate by name – re-anchored links, unhonored constraints, manually scheduled tasks – rather than silently approximating. The repair map arrives before the first percentile does.

--- In practice: run the delay test on your current plan today – one mid-plan task, plus one month. Then count tasks with more than 200 days of float. The first number tells you if the network exists; the second tells you how much of it is missing.

Chapter 5 – From risk register to simulation

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

Every program keeps a risk register. Almost none of them are connected to the schedule. Likelihood and severity get scored, the heat map gets presented – and the finish date stays exactly where it was. That disconnect is the single cheapest thing to fix in most risk processes, because the register already contains the information a simulation needs.

What a register row really says

"Thermal-vacuum chamber availability. Likelihood 3, severity 2. Affects the [TVAC] (/guide.html#tvac) test task." Read as simulation input, this row makes three claims: an event exists with some probability of occurring; if it occurs, a specific task gets longer; the impact has a rough magnitude. That is precisely the shape of a *discrete risk overlay*: a Bernoulli draw per iteration – does the risk fire in this future? – and, when it fires, days added to the mapped task, propagating through the network from there.

Note what the register does *not* say: which dates change. It cannot know – that depends on the network, the float, and every other risk firing or not in the same future. The simulation exists to compute exactly the consequence the register cannot state.

The mapping is the work

Connecting register to schedule means answering, for each risk: *which task does this hit?* In our experience this question exposes more process weakness than any audit. Risks mapped to nothing ("program-wide") are often risks nobody has thought through to a mechanism. A register where most rows map to specific tasks is a register that has been argued with – and arguing with the register is the point.

Two honest categories remain after mapping. Genuinely program-wide risks (a funding delay touches everything and nothing specifically) deserve the label. And risks that turn out, on inspection, to be *issues* – already happened – belong in an event log with recorded actuals, not in the probability space.

Scores are not probabilities – yet

A likelihood of "3" on a 5×5 grid is an ordinal judgment, not a percentage. Turning scores into simulation inputs requires a mapping – 3 means roughly such-and-such probability; severity 2 means roughly so-many days – and the quality of that mapping decides the honesty of everything downstream. Chapter 7 treats this fully; the short version is that mappings

disciplined by historical outcomes beat mappings voted on in a meeting, and both beat the most common alternative, which is no mapping at all.

The trap to avoid at this stage is the seductive one: ranking risks by likelihood × severity and calling it analysis. That product ignores the network entirely. A modest risk on a zero-float chain into your integration event can move the finish more than a frightening risk sitting on 200 days of slack. Which risks actually matter is a measured question, and it is the next chapter's subject.

> **How we do it in Epoch SRA** – The register imports from your own sheet in your own layout: select the range, and the tool maps your columns, preserving your taxonomy verbatim. Your likelihood/severity scores map to priors through a separate field, so your categories stay yours while the statistics stay calibrated. Risks that cannot be mapped are reported and excluded – never silently spread across the plan.

--- In practice: take your register's top three risks and write, next to each, the specific task ID it would delay and a days-range for the impact. If any of the three resists this exercise, you have learned something about that row — and about how ready the register is to inform dates.

Chapter 6 – Sensitivity: measured, not assumed

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

Two rankings of the same risks. The first: sorted by likelihood × severity, the register's own arithmetic. The second: sorted by how much each risk actually moves the program finish, measured across thousands of simulated futures. If the two rankings matched, the second would be redundant. In our experience, they do not match – and the differences are where the money is.

Why the rankings diverge

Likelihood × severity is a statement about a risk in isolation. The finish date is a property of the whole network. Between the two sit three amplifiers the register cannot see. *Position*: a risk feeding a zero-float chain into the integration event transmits every day of impact; the same risk on 200 days of float transmits nothing. *Merge structure*: a risk on the fifth parallel path into a milestone attacks a merge point already eroded to 59% – its marginal damage is outsized. *Correlation with criticality*: some tasks are only critical in the bad futures, exactly when risks fire; a static critical path never shows them.

How measurement works

The clean method uses the simulation's own bookkeeping. For each risk, partition the iterations: those where it fired, those where it did not. Compare the finish distributions of the two groups – the difference at the percentile you care about (say P80) is that risk's measured contribution, in days, network effects included. No second model, no assumed elasticities: the same futures, sorted two ways.

Presented as a tornado – bars ordered by contribution – this becomes the most decision-dense chart an SRA produces. It answers the only question a mitigation budget cares about: *if we killed this risk, what would we actually buy?*

One honesty rule for the method: some risks fire too rarely in a finite run to measure. The honest rendering is "unmeasurable", never zero – zero is a claim, and an unearned one.

The row that wins the meeting

On our demo program, the largest single mover of the P80 finish is a comms-subsystem risk that sits on the critical path in only 19% of futures. Sorted by criticality it is row twenty; a critical-path review never discusses it. Sorted by measured contribution it is row one, at over

130 days of P80 impact. That inversion is not a curiosity – it is the routine product of position and merge structure, and it is invisible to every method short of measurement.

The practical consequence: mitigation money allocated by register score is systematically misallocated. Not because the scores are wrong, but because the scores answer a different question than the one the program finish asks.

> **How we do it in Epoch SRA** – Sensitivity is computed from the fired-versus-not partition of the actual run, per risk, and rendered as a tornado sortable by days or by cost contribution. Unmeasurable rows show "–", never zero. The comms-subsystem example above is our demo dataset; the tornado's top row and the register's top row disagree there by design of reality, not of the demo.

--- In practice: before your next mitigation review, write down which risk your team would fund first. Then measure. The comparison of those two answers — gut ranking versus computed contribution — is the fastest credibility test an SRA can pass in front of a skeptical room.

Chapter 7 – Calibration: whose numbers, and why trust them

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

A Monte Carlo simulation is arithmetic on assumptions. The arithmetic is never wrong; the assumptions usually are. Calibration is the discipline of forcing those assumptions to answer to reality – and its absence is the quiet scandal of most schedule risk practice.

The standard input, and its problem

Most SRA guidance says: ask the engineer for minimum, most-likely, and maximum durations, fit a triangle, simulate. The output looks rigorous. The input is the same optimism Chapter 1 described, now wearing error bars. Decades of estimation research say expert ranges are too narrow – actuals fall outside stated min-max far more often than the stated confidence implies. Simulating uncalibrated ranges produces distributions that are precise, plausible, and systematically tight: the P80 you compute is the world's P60, and you find out at the worst possible time.

What calibration means, concretely

Take historical programs where the outcomes are known. Replay them through the model: with the priors you propose, would the claimed percentile bands have contained the actual outcomes at the claimed rate? If your P20–P80 band is honest, roughly 70% of actuals should land inside it – not 95% (bands too wide, uselessly conservative), not 40% (too narrow, dangerously confident). Tune the priors until the claim and the history agree. That closed loop – claim, replay, tune – is the entire idea. A model that has never been replayed against outcomes has a precision claim and no accuracy claim.

The objections, taken seriously

"Calibrated against whose programs?" A fair question with an honest answer: whatever history was available – which is always a sample, always from somewhere. The defense is not that the sample is perfect; it is that the alternative input is calibrated against nothing at all. A rough map, honestly labeled, beats a confident blank page.

"Every program is unique." True at the task level, mostly false at the distribution level. Programs are unique in content and depressingly similar in failure statistics – optimism, [merge bias](/guide.html#merge-bias), supplier slips, test-retest cycles. This regularity is why reference-class forecasting works anywhere. And the objection contains its own endgame: as your organization accumulates its own actuals, the same replay loop can run on *your* history, and uniqueness flips from objection to asset.

Provenance: the label that earns the trust

Calibration evidence is never uniform across a distribution. The middle percentiles, where most outcomes land, can be disciplined by modest history. The deep tails – P90, P95 – rest on distribution-shape assumptions that thin samples cannot check. An honest model says so, per number: this percentile is backed by replayed outcomes; that one is a model estimate. In front of a review board, "here is which numbers carry evidence" beats "trust the software" every time it is tried – because it converts the board's skepticism from an attack into the model's own stated structure.

> **How we do it in Epoch SRA** – Priors are field-calibrated by replaying historical space-program schedules, tuned so the claimed P20–P80 band contains roughly 70% of actual outcomes. Every percentile in every output carries its provenance label: field-calibrated (P50/P80) or model estimate (the extended percentiles). Calibration against a customer's own actuals is on our roadmap – the mechanism does not care whose history it eats.

--- In practice: find one finished project in your organization with a preserved baseline. Compare its final actual finish to the range anyone would have stated at baseline time. That single data point, honestly faced, is the beginning of a calibration set — and usually the end of confidence in bare expert ranges.

Chapter 8 – Cost through the schedule

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

Schedule risk and cost risk are usually analyzed by different people with different tools, then stapled together in a review. The staple is the weakness: on most programs, the dominant cost risk *is* the schedule. Standing armies bill by the day whether or not the chamber is free; every week of slip is payroll, facilities, and overhead with no offsetting progress. A cost model that does not ride on the schedule distribution is modeling a different program.

The mechanism: one simulation, two readouts

The honest construction requires no second model. Each simulated future already contains every task's duration. Give tasks a burn rate – cost per working day – and a fixed cost where one exists, and each iteration prices itself: task cost equals fixed plus rate times that future's duration; program cost is the sum, plus the cost impacts of whichever discrete risks fired in that future. Twenty thousand futures yield a cost distribution exactly as they yielded a date distribution – same draws, same correlations, no stapling.

Two numbers fall out that reviews immediately adopt. The *cost P80*: the budget figure with a stated confidence, replacing the traditional single estimate plus arbitrary contingency percentage. And the *delay price*: across the joint samples, the expected cost per week of finish slip past P50 – schedule risk translated into the only unit some steering committees hear. "This risk moves P80 by six weeks" lands differently when the next line reads "at €85k per week."

Where the rates come from, and what that means

Burn rates are the user's numbers – loaded labor, facility costs, level-of-effort overheads. This is both the method's practicality (every program controller has them) and its honesty boundary: the cost distribution inherits its *shape* from the calibrated schedule simulation, but its *scale* from your rates. It is therefore *derived*, not calibrated – no replay of historical cost outcomes stands behind it unless one has actually been run. A tool or analyst who labels cost percentiles with the same confidence as backtested schedule percentiles is borrowing credibility the numbers have not earned. Say "derived from the schedule simulation through your rates" and the number is defensible; say "calibrated" and it is not.

What this replaces

The traditional alternative is contingency-by-percentage: estimate the cost, add 15% because the last program added 15%. The joint construction replaces a folk number with a distribution:

contingency becomes the distance from your point estimate to your chosen cost percentile, and it moves when the schedule risk moves – which is the whole point. When a mitigation kills a risk, its cost consequence is measurable in the same run that measures its schedule consequence, and the mitigation's business case writes itself.

> **How we do it in Epoch SRA** – Optional cost-per-day and fixed-cost columns on the schedule, cost impacts on register risks; the same 20,000 iterations produce cost P50/P80, a cost histogram, per-risk cost contribution in the tornado, and the delay price. Every cost output carries the derived-not-calibrated label, enforced by a build-time check – the honesty rule is code, not a style guide.

--- In practice: put a burn rate on your five largest level-of-effort tasks — payroll divided by working days is enough — and compute what one month of program slip costs. That single number, however rough, upgrades every schedule-risk conversation your program will have this year.

Chapter 9 – Running an SRA your review board believes

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

The mathematics of the previous chapters fail for a non-mathematical reason more often than any other: the room does not believe the numbers. Belief is not won by more decimal places. It is won by reproducibility, provenance, and a cadence that survives contact with a real program. This chapter is about the operating discipline.

Reproducibility, or it did not happen

An analysis that cannot be re-run is an anecdote. The minimum standard: a fixed random seed, recorded inputs, and a stamped output – run timestamp, iteration count, seed, and input fingerprint on every chart and table that leaves the tool. When a stakeholder asks "is this the current picture?", the stamp answers; when two analyses disagree, the fingerprints say why. Boards have been burned by stale slides too often to extend trust without this, and they are right.

Answer the provenance question before it is asked

Chapter 7's labels earn their keep in the meeting. Present percentiles with their evidence class stated – calibrated versus model estimate – and the predictable challenge ("how do you know these numbers?") has already been answered in the artifact itself. The same applies to limitations: an analysis that states what the import could not translate, which quality checks failed, and which risks were unmeasurable is an analysis that has done the board's due diligence for it. Stated limitations read as competence; discovered ones read as concealment.

Cadence: the analysis is a verb

A one-off SRA is theater with better graphics. The value compounds only on a rhythm: re-import the current plan, re-run, and compare against the previous run at every review cycle. The comparison is the product – *which way did P80 move since last month, and which risks moved it?* – because a program manages deltas, not snapshots. This is also the quiet test of your toolchain: if re-running next month's plan means rebuilding mappings and re-entering risks, the cadence dies by friction, and the analysis with it. Task identities and register mappings must survive a re-import, or the second month never happens.

The review agenda that works

Four items, in order, before any percentile is shown. One: schedule quality – what the checks found, what was repaired. Two: what changed since last run – plan deltas, register deltas.

Three: the distribution – P50, P80, and the gap, trended against previous runs. Four: the tornado – measured contributions, and the mitigation decisions they imply. Ending on decisions rather than dates is what separates an analysis the board *uses* from one it *receives*.

> **How we do it in Epoch SRA** – Every output is stamped: timestamp, seed, iterations, schedule signature, provenance per percentile. Re-import is identity-stable – task IDs and register mappings survive plan changes, with an orphan report for anything that no longer resolves – so the monthly cadence costs minutes, not an afternoon. The quality-check summary opens every results pane, in the same order this agenda recommends.

--- In practice: put a recurring thirty-minute slot in the calendar the day before each program review, titled "re-run and diff". The habit, not the tool, is what makes the numbers believed by month three — because by then the board has watched the distribution respond to reality twice.

Chapter 10 – Choosing tools: the questions that matter

The Quantitative Schedule Risk Analysis Handbook · Edition 1, August 2026 · by the team behind [Epoch SRA] (https://epochsra.com), an Excel add-in for schedule risk analysis · corrections welcome at contact@epochsra.com

This chapter is deliberately vendor-neutral, ours included. The SRA tool market is small and genuinely differentiated – enterprise platforms, scheduler add-ins, Excel-based tools, and general simulation engines all coexist because they answer different organizational shapes. The wrong question is "which tool is best?" The right question is "who will operate it, on what schedule data, under what constraints?" – and the checklist below turns that into vendor conversations.

Where does my schedule data go? Local computation, vendor cloud, or on-premise server are different answers with different consequences – for export-controlled programs, often decisive ones. Ask precisely; "secure" is not an architecture.

What happens to constructs the tool cannot handle? Every import or integration has edges – link types, calendars, constraints, resource logic. The differentiating answer is not "we support everything" (no one does) but "here is what we do with what we don't support." Silent approximation and honest reporting are different products at the same feature list.

Which of your numbers are calibrated, and against what? If the vendor's answer to "how do you know your distributions are honest?" is a description of the input dialog, Chapter 7 applies. Ask whether any percentile has been backtested against actual outcomes, and whether the output distinguishes evidence-backed numbers from model estimates.

What does the monthly cadence cost? Not the license – the labor. Re-import a changed plan: do mappings survive? Re-run: are results reproducible and diffable against last month? A tool that makes the first analysis easy and the twelfth expensive optimizes for the demo, not the program.

Can my schedule quality be assessed before my risk is? Chapter 4's argument in procurement form: ask what schedule-health checks run, and what happens when they fail.

What does a seat cost per year, all-in? Published pricing, quote-based pricing, per-seat versus suite, maintenance terms. None of these answers is wrong; unavailable answers are.

Can I evaluate on my own data without a sales process? The fastest tool evaluation is your real schedule through a real trial. Vendors who make that possible in an afternoon are making a statement about their product; vendors who route it through three calls are making a different one.

Weight these by your organization's shape. A risk office running a thousand-task portfolio weights integration depth and enterprise features. A program manager who *is* the risk process

weights time-to-answer, cadence cost, and data locality. Both are right, and most tool disappointment traces to buying for the other organization's shape.

> **How we do it in Epoch SRA** – Our answers to this checklist are published on our site, alongside comparisons with the other tools in this market that state where each is stronger. We think the checklist, honestly answered, is the fairest sales pitch available to anyone – including our competitors.

--- In practice: send this checklist, verbatim, to every vendor on your shortlist and to your own team about the current process. The current process is also a tool, and it should survive the same questions.