CHAPTER 7

70/30 as an Architectural Principle — building the two layers in practice

> *Full automation fails one way; a human process with an AI garnish fails another. The savings appear only where both layers — the machine layer and the social layer — are designed together, each running at its own pace, both closing into one loop of accountability. That is the 70/30 principle as architecture, and it can be built in eight deliberate steps.*

Bain & Company showed in a 2025 study: companies that combine generative AI with a genuine redesign of their processes cut costs by up to 25 percent. Companies that lay AI on top of existing processes, leaving the processes untouched, harvest single-digit percentages or nothing at all.[1] That 25-percent figure is the strongest external evidence for the 70/30 approach. The technology on its own yields no economics; the economics come from constructing the human and technical layers together.

One system, two layers

Most architecture diagrams of AI systems show the technical layer: the model, the data pipeline, the infrastructure, the integrations — sometimes with the user interface on top. A 70/30 architecture needs a diagram on which the technical and the social layer appear together and in relation: model, data pipeline, infrastructure on one side; roles, escalation paths, mandates, thresholds, what gets logged and who reviews it on the other. Both layers are full, equal components of the system. Only together do they form an AI system — a technical layer alone remains an automated process

without control, and a social layer alone remains a collection of human decisions without automated support. Norbert Wiener described this as early as 1950 in *The Human Use of Human Beings*: an automated system acquires its meaning only in combination with a human layer of oversight; without it, the cybernetic loop closes on itself and loses its corrective circuit.[2]

The technical layer holds the model — or several models working together and merging their outputs into one decision, a set the field calls an ensemble. It holds the data pipeline that cleans and normalizes incoming data and builds the features the model reads. It holds the serving infrastructure, the scoring and threshold logic that turns model output into decisions or into flags for human review, the logging — what the system records automatically — and the monitoring: performance metrics and the detection of gradual shifts in the incoming data over time.

The social layer holds the role definitions — training, control, accountability, with actual names attached. It holds the escalation paths: who is notified, in what order, under what conditions. It holds the manual-override procedures — how humans overrule system decisions and what gets logged when they do. It holds the mandates: who may take which decisions, with explicit limits of authority. It holds the oversight routines — how often the escalation thresholds, the numeric limits at which a decision passes from machine to human, are reviewed and who takes part. And it holds the documentation duties: what must be recorded, by whom, by when.

Crew Resource Management as the institutional precedent. Aviation reached this conclusion in the 1970s. By 1979, accident reports showed a persistent pattern: technically sound aircraft were crashing through coordination failures in the cockpit — a senior pilot who took no objection from a junior, hierarchy bias, a refusal to delegate. The NASA psychologist John Lauber coined the term Crew Resource Management in 1979 — the trained management of the cockpit crew as a system of roles rather than a set of individuals. Within a decade the FAA, the United States aviation authority, made this training mandatory for every US carrier through

Advisory Circular 120-51; the current edition documents five generations of the method, extended to multi-member crews and single-pilot operations. [3] The lesson carries directly to AI: aviation formally recognized that even the most refined aircraft is only half the system. The other half is the trained social architecture of the cockpit — roles, escalation paths, the junior pilot's mandate to say *stop this*. That is the social layer as an equal component. AI systems today stand roughly where aviation stood in 1978: technical refinement is growing fast, and the social layer lags behind it.

The eight-step process: from process map to escalation drill

What follows is a practical sequence for building a 70/30 system. Each step answers a category of failure that repeats across companies precisely when that step is skipped. Cutting the list down to three or four items looks attractive at the start and sharply lowers the odds that the system still works after go-live. Each of the eight steps closes a specific failure class, and without all of them no 70/30 architecture scales beyond the pilot — the limited trial on one section of the business that precedes a full roll-out.

Step 1: the process map. The first step describes the current process — no model yet, no AI at all. The team records the process as it runs today with people in it: every step, every decision, every hand-off between participants. The work takes time, often feels excessive, and produces a standing temptation to shorten exactly this phase. Without it, the team does not know what it is about to automate and risks laying AI over a distorted picture of its own work. The result is a process map — a graphic or tabular representation with every decision point marked, and everything that follows builds on it.

Step 2: identifying the decision points. On the map from the previous step, every point at which a decision is taken gets singled out. Not all of these points are candidates for automation — some stay with a human regardless of technical feasibility, for regulatory, ethical or contextual reasons.

For each point the team answers four questions: what does an error cost here, what regulatory weight sits on this decision, how often is it taken in a normal working rhythm, and how variable are its inputs. The answers decide whether a point should be considered for automation at all, and they become the selection criteria for the next step.

Step 3: defining the 70 percent. From the identified points, the team selects the subset the system should handle automatically. The criteria: high frequency, low variability of inputs, an acceptable cost of error, and no regulatory requirement for a human signature. The set of points that meets all four criteria at once is what the 70 percent of automation refers to. The share varies between companies — disciplined processes sometimes reach 80 percent, heavier regulatory environments land visibly lower. If the achievable share comes out far below 70 percent, that is a signal to redesign the processes rather than to shrink the ambition: processes have to be rebuilt until they can meet these criteria, otherwise the economics of the 70/30 approach stay out of reach.

Step 4: defining the 30 percent. The points that stay with humans form the second half of the system in full standing, and they need the same design attention as the automated half. For every human decision point, three parameters are recorded: who exactly takes the decision, under what conditions it reaches that person, and within what time window it must be taken. Named individuals are the point — functional labels are not enough. An entry such as *the responsible line manager* or *the expert on duty* fails at the moment of an incident, because behind such a formula nobody feels personally addressed. A sufficient entry names the person, their position and a designated deputy for absence, vacation or shift change, so that at the moment of escalation there is no ambiguity about whose desk the decision lands on.

Step 5: setting the thresholds. For every automated point, the team defines the thresholds at which the matter passes to a human. Typical formulations: *if model confidence falls below 85 percent, the decision escalates to the AI operator*, or *if the decision affects a batch above 500 units, human confirmation is required*. Thresholds must be specific, measurable and written into the operating documentation rather than buried in code — otherwise they stay invisible to the people who work by them every day, and no threshold review is possible without a dive into the source.

Step 6: designing the logging. Logging covers every system decision, every escalation and every manual override — with name, timestamp and the reason for the action. Separately recorded: where the logs live, who has access, and how long they are retained. For high-risk systems under the EU AI Act the minimum retention is five years; internal policy may set longer periods where industry standards or the risk profile demand it.

Step 7: the operating documentation. Before go-live, three signed documents exist. The *system description* records whom the system serves, what it does, which escalation thresholds are set, and who answers for training, operational control and final decisions. The *operating procedure* describes what the operator does in normal mode, on escalation and on failure. The *failure recovery plan* defines what happens when the system is fully unavailable and how the process continues by hand until the system returns to normal operation.

Step 8: the escalation drill. Before go-live, the team simulates an incident — it picks an escalation scenario and walks it in real time: who gets notified, who reacts, what that person does, what enters the log. The elapsed time is measured right there, during the drill. The acceptable length of an escalation path depends on the type of decision: for critical incidents it is usually a matter of minutes, for routine ones tens of minutes, and the concrete limit belongs in the operating procedure. In a real incident that time will stretch further — stress, night hours, a key person missing — so a drill under calm conditions only sets the lower bound of the estimate.

NASA's "Lock the Doors" as the precedent for the drill. NASA Mission Control has a documented procedure called "Lock the Doors", declared by the Flight Director at the moment of a critical incident. The doors of the control room are physically locked, outside access stops, and every station switches into a mode the field describes as freeze, isolate, protect: freeze the current state of the system, isolate it from new changes, protect the documentation and the logs. The procedure is not for the incident itself — it is for what comes after. It guarantees that when the investigation starts, the data is in its original state instead of being overwritten during a chaotic response.[4] For AI operations the equivalent is direct: when an escalation arrives, the first move is to record which state of the system led to it — with whom, at what time, on which inputs — and only then to fix anything. Without that, the incident review a week later is speculation instead of analysis. A drill that skips the freeze phase trains the reaction and teaches nothing about diagnosis.

The documentation standard: what exists in writing before the model goes to production

This is the minimal mandatory list, with no claim of completeness. High-risk systems under the EU AI Act require substantially more.

System card. Name and version of the system. Components: model, version, infrastructure, integrations. Classification under the EU AI Act. Date of last update and a signature.

Roles and contacts. Training: name, position, contact. Operational control: name, position, contact, deputy's contact. Accountability: name, position, contact. Approval date and the signature of every person involved.

Operating thresholds. Every automatic escalation threshold, as a number. Every contextual escalation trigger. Date of the last review and the person who approved it.

Escalation path. Scenario → who is notified → response time → what that person does. Per scenario: standard escalation, night and weekend escalation, critical escalation.

Manual-override procedure. How is the override technically performed? Which fields must the log entry contain, in what format? Who reviews the override logs, and when?

Failure recovery plan. What happens on full system failure? The manual substitute procedure. The criterion for returning to the system after recovery.

Typical mistakes

Mistake 1: concentrated mandates. An experienced technician receives all the roles — training, control, accountability, plus *general AI responsibility*. On paper it looks clear. In operating reality the person is overloaded, cannot be everywhere at once, and at the moment of an incident becomes the single point of failure. The distribution of roles must reflect real availability rather than organizational convenience.

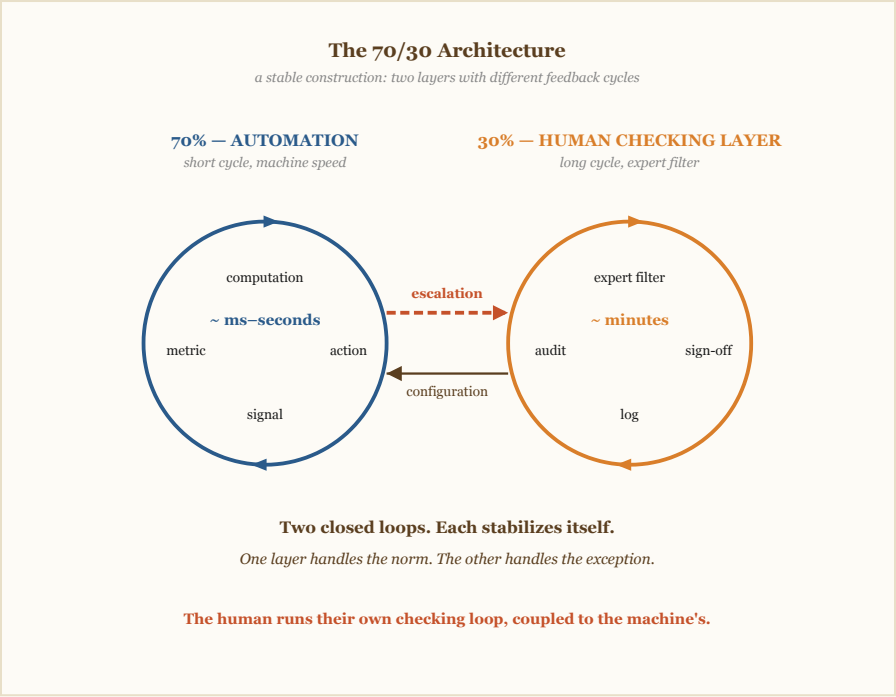
Mistake 2: technical thresholds without human context. *Escalate when model confidence falls below 85 percent* sounds precise. But if the operator receiving the escalation does not know what *confidence of 84 percent on this type of decision* means in practice, no considered decision is possible. Technical thresholds need a human explanation: *this means the system is unsure of its own decision — an unfamiliar document format, say, or a new supplier. In that case the operator checks the document type and the key fields, then either confirms the decision or returns the document for reprocessing.*

Mistake 3: documentation nobody reads. Excellent documentation in a shared folder on the corporate portal, unopened since go-live, is not documentation. Operating documentation must be available where people need it: at the operator's workstation, inside the system interface, in the first window that opens when an escalation arrives.

Mistake 4: oversight without a date. *Thresholds are reviewed as needed* means they are not reviewed. Oversight belongs in the calendar: once a quarter — or after every significant incident — a concrete meeting with concrete people that reviews concrete metrics and decides whether thresholds or role assignments need to change.

Mistake 5: no hand-over plan for staff changes. The AI operator goes on vacation. Or resigns. Who takes over the function? Has that person been briefed? Do they know where the operating documentation lives? Knowledge transfer must be a planned component of the system rather than an improvisation.

> *Bain & Company, 2025: companies that combine generative AI with process redesign cut costs by 25 percent. The economics appear only together with the redesign — and the 70/30 approach is precisely a method of that redesign.*



Two closed loops: the machine loop handles the norm in milliseconds to seconds, the human loop handles the exception in minutes. Each stabilizes itself at its own pace — coupled through escalation one way and configuration the other.

Three Questions for Your Own Context

1. **The layer diagram.** Draw your AI system in two layers — technical and social. If the social layer cannot be drawn, it does not structurally exist.

2. **The threshold review.** When were your system's operating thresholds last reviewed? Who was in the room? What was changed? If the answer is *can't remember* or *never* — act.

3. **An escalation drill, now.** Pick a scenario and measure the time. Who was notified? Who reacted? What took longest? The drill result is the agenda for the next oversight meeting on the system.

Sources

1. Bain & Company, research on generative AI adoption, 2025: companies combining generative AI with process redesign report cost reductions of up to 25 percent; AI layered onto unchanged processes yields single-digit gains.
2. Norbert Wiener, *The Human Use of Human Beings: Cybernetics and Society*, Houghton Mifflin, 1950.
3. John K. Lauber, "Resource Management on the Flightdeck", NASA Conference Publication 2120, June 1979; FAA Advisory Circular 120-51E, "Crew Resource Management Training", January 22, 2004 — current edition.
4. NASA Johnson Space Center, Flight Director Operations Handbook; the "Lock the Doors" procedure as described in Gene Kranz, *Failure Is Not an Option*, Simon & Schuster, 2000.