THE EVOLUTION OF MANUFACTURING SOFTWARE · 1950–2026

From Punch Cards to Predictive Plants

Seventy-five years of teaching the factory to remember — and now to think

Matthew Mortensen

Founder, Reliance Systems Group

First edition, 2026.

Dedication

For the planners, schedulers, buyers, operators, and shop-floor supervisors who keep the world's factories running on a daily basis. And for the operations leaders who risked grafting new systems onto the spine of their profitable (and predictable) plants without fully seeing the path and its obstacles, but with total faith in their people to make it happen. And most importantly, for my mother, who got me my first factory job.

Epigraph

> *The biggest risk in manufacturing software has never been adopting the new thing too early. It has always been waiting until the new thing was safe — by which time your competitor had already learned to run it.*

Table of Contents

Preface — Why I Wrote This Book

1. The Machine That Remembers
2. The Paper Plant
3. Iron, Cards, and the Nightly Run
4. The Order Behind the Order: MRP
5. Closing the Loop: MRP II
6. One Spine for the Enterprise: ERP
7. The Nervous System: MES
8. Thinking Ahead: APS
9. The Plant in the Cloud
10. The Six Rivers: Manufacturing's Core Business Processes
11. Proof, Not Promises: Quality and Compliance
12. The Plant That Learns: AI and Machine Learning
13. 2026 and the Self-Driving Plant

Glossary and Acronyms

About the Author

Preface — Why I Wrote This Book

Any success I have in this life can be traced back to a decision my mother made in 1988. Tired of not having the resources necessary to give her two boys a life firmly in the middle class, and no doubt eyeing the earnings ceiling of her cashier position at the local K-Mart, she took a factory job. That was not an uncommon decision to make in 1980's America; in fact, by that time it was a proven path to a higher standard of living. I certainly didn't grasp the weight of that decision at the time, beyond the fact that it meant she worked odd hours and I didn't see her as much anymore. She grew that first job into a career that has lasted nearly 4 decades in that same company. That decision made back in 1988 helped me pay for college, then grad school, and eventually helped me get my first "real" job. I didn't want to work in manufacturing. In fact, I thought I was too good to be in a factory. The global recession of 2009 and some personal happenings gave me few other choices for work. Along the way something magical happened - I got to see my mom through a professional lens, and I got to know her longtime coworkers. I caught glimpses of this amazingly intricate system they were orchestrating in order to make and deliver medical devices that were saving lives everyday around the world. The glimpses became total immersion inside this orchestra when the company decided to implement an ERP system. I was hooked.

I was assigned to be the lead of the validation team for that ERP implementation. I learned about all of the processes flowing through a plant to get raw materials in and finished goods out. I learned maybe a thousand new acronyms so I could speak the language of the big consulting firm leading the configuration. I was a Quality Engineer and an organic chemist by education, but I had shown a knack for understanding large systems and how to map them, test them, train on them. My team eventually helped prevent a brittle, not-quite-ready system from going live. But the biggest takeaway for me from that project was how knowledgeable all these people were about the inner workings of their factory. They were professional and brave in ways I had not encountered up to that point. Going into that factory job, I had a certain image of a talented professional painted in my mind, an image molded by 10 years in higher education institutions. Leaving that first factory job, I had been transformed into someone with tremendous respect for the magic that happens to keep all the plates spinning in that beautiful orchestra called a manufacturing plant.

This book is the story of how that orchestra has evolved over time. I say orchestra deliberately because this is not a story about engineering of one discipline or another; it isn't about a specific type of product or market segment it goes into; it isn't about CAD or PLCs and firmware on a machine. This story is about the business processes and the associated software that make it possible to order a multitude of raw materials and send an equally impressive amount of whatever "widget" being made out the door. It begins in 1950, in a factory where the most advanced information technology was a wall of pigeonholes and a clerk with a sharp pencil, and it ends in 2026, in plants that are beginning to schedule, diagnose, and correct themselves with almost no human keystrokes at all. Between those two points runs one of the great untold engineering stories of the modern economy: the slow, expensive, frequently painful project of teaching the factory to remember what it was doing, and then to reason about what it should do next.

I am, by temperament, a futurist. I believe that the manufacturers who win the next two decades will be the ones willing to adopt powerful technology before it is comfortable — to absorb the risk, survive the go-live, and compound the advantage. That conviction colors every page that follows. But I have also seen failed implementations and respect how hard this is. I have no patience for hype that ignores the operator who still has to hit a production number on Tuesday morning.

So I have tried to write a history book (with one eye on the future) that honors my former coworkers and the many other factory workers all over the world: a history that is honest about the pain, generous about the people, clear about the technology, and unembarrassed about its optimism for what artificial intelligence is about to do for the companies that make the physical world. If you run a plant, sell to one, build software for one, or simply want to understand how the things around you get made, I wrote this for you.

Let's begin where the factory's memory began — on paper.

— M.M., 2026

Chapter 1 — The Machine That Remembers

Walk onto any factory floor at three in the morning and you will hear two kinds of memory at work. There is the memory in the steel — the worn jig that only fits one way, the lathe that drifts a thousandth of an inch when the building gets cold, the operator who knows by the sound of the spindle that a tool is about to break. And there is the other memory, the one we built on purpose: the card, the ledger, the database, the model. This book is the story of that second memory — seventy-five years of teaching the plant first to remember, and then, at long last, to reason. I have spent my career standing between those two memories, and I am here to tell you the most interesting chapter is the one we are living through right now.

The thesis in one sentence

Manufacturing software is the seventy-five-year project of teaching the plant first to **remember**, then to **reason**.

That is the whole book, compressed. Everything that follows — the punch cards and the nightly batch runs, the bill-of-material explosions and the closed loops, the single database and the shop-floor terminals, the cloud tenants and the edge gateways, and finally the models that learn — is a variation on those two verbs. Remember. Reason. We spent the first six decades getting very, very good at the first verb. We are now, in 2026, in the early and exhilarating years of the second.

I want to be precise about what I mean, because the words *remember* and *reason* are doing a lot of work and I do not use them loosely.

When I say a plant **remembers**, I mean it can answer the question *what is true right now, and what was true before?* How many of part number 4471-A are sitting in the bin on the east wall? What did we promise the customer, and when? What did this lot of stainless cost us, and which heat number did it come from? Who ran the third shift on the press last Tuesday, and what did the gauge read at 2:14 a.m.? A plant that remembers well can be audited, can be trusted, can be reconstructed after a fire or a recall. A plant that remembers poorly is flying on rumor, on the expeditor's gut, on the planner's private spreadsheet that nobody else can open.

When I say a plant **reasons**, I mean something harder. I mean it can answer the question *what should we do next, and why?* Given everything it remembers — and given everything it can forecast, simulate, and weigh — the reasoning plant proposes a decision. Build this order before that one. Reorder now, not next week, because the supplier in Ohio is about to slip. Pull the press offline for maintenance Thursday at noon, because the vibration signature says the bearing has four days left, not four weeks. Reasoning is not just recall. Reasoning is judgment rendered into action.

For most of the history I am about to tell you, our software could remember magnificently and reason barely at all. We built systems of **record** — vast, disciplined, expensive machines for remembering. And then we asked human beings to do all the reasoning on top of that memory: the planners, the schedulers, the buyers, the quality engineers, the plant managers who carried the whole factory in their heads. The software remembered. The people reasoned. That division of labor defined the industry for fifty years.

What is changing now — what makes this the most consequential moment in the history of manufacturing software — is that we are finally building systems that **reason** alongside us. Not instead of us. Alongside us. The arc of these seventy-five years bends, slowly and then suddenly, from the system of record toward the system of decision. That is the bend this book is about, and it is the bend I have bet my career on.

Who is telling you this, and why you should care that I am an optimist

My name is Matthew Mortensen. I founded Reliance Systems Group because I believed, and still believe, that the manufacturers who win the next twenty years will be the ones who treat new technology as a lever rather than a threat. I have spent my working life on the seam between the people who make things and the systems that are supposed to help them. I have seen go-lives that went beautifully and go-lives that did not. I have never forgotten the impact to real people when systems didn’t work as intended, and I never hand-wave the cost of change. Manufacturing is hard in a way that software people who have never smelled cutting fluid do not always appreciate. The machines do not care about your sprint cadence. The customer's truck shows up at the dock whether or not your integration is finished.

So let me be clear about my disposition, because it colors every page that follows. **I am a risk-positive futurist.** I do not mean reckless. I mean that, after watching this industry for a long time, I have concluded that the single largest risk in manufacturing software is not adopting the wrong system. It is standing still. The firms I have watched compound advantage over decades are not the ones who waited for the technology to be proven, safe, and cheap. They are the ones who adopted early, absorbed the pain, learned the new tool before their competitors did, and were three years up the learning curve when everyone else finally arrived. Risk, in this business, is not a bug to be minimized to zero. Handled with eyes open, risk is a feature — it is the toll you pay for first-mover position, and the firms that refuse to pay it do not avoid risk. They simply trade the visible risk of change for the invisible, slower-acting risk of obsolescence.

I will make that argument again and again in this book, in different costumes, because it is the through-line of my whole professional life. But I am not asking you to take it on faith. I am going to show you, era by era, what the early adopters got and what the laggards lost. The pattern is remarkably consistent. The technology changes. The shape of the bet does not.

The credo, stated plainly

Since the style of this book is to say things plainly, here is my credo without ornament.

I believe technology is a lever, not a threat. I believe the upside of a new capability is almost always larger, and arrives sooner, than the people protecting the status quo will admit. I believe the human being on the shop floor is not being replaced by software — has never once, in seventy-five years, actually been replaced by software — but is instead being continuously promoted, freed from the work a machine can do so they can do the work only a human can. I believe that the biggest mistake a manufacturer can make is to wait for certainty, because certainty in this field arrives only after the advantage has been claimed by someone braver. And I believe, with particular conviction, that artificial intelligence is not the end of the manufacturing professional but the most powerful tool ever handed to one.

That is the credo. The rest of the book is evidence.

The four-era arc

To understand where we are, you have to understand how we got here, and the cleanest way I know to organize seventy-five years is into four great eras. Each era is defined not by a single product but by where the plant's memory physically lived and how the data moved to the decision.

The first era was **paper**. From the dawn of organized manufacturing through the 1960s, the plant's memory lived on cards, in ledgers, on dispatch boards, in the traveler packet that rode along with the job. The data was visible, tactile, and resilient — a fire could not corrupt a Kardex card the way a disk crash corrupts a database — but it was slow, error-prone, and stubbornly local. To know the true state of the plant, you had to physically walk it. The reasoning lived entirely in human heads.

The second era was the **mainframe**. Beginning in the late 1950s and dominating the 1960s and 1970s, centralized batch computing pulled the plant's memory off the floor and into a glass-walled, air-conditioned room tended by a priesthood of data-processing professionals. The IBM 1401 and later the System/360 could remember more than any room full of clerks, and they could compute reorder points and explode bills of material at a scale no human team could match. But they remembered in batches. You fed the machine punch cards, you waited, and the next morning — or, for the big runs, after the whole weekend — you got your answers. The plant's memory was now powerful and centralized, but it was also delayed, and it spoke a language only the priesthood understood.

The third era was the **onsite, on-premise server**. From the 1990s into the 2010s, the client-server revolution put real computing power on company-owned hardware in the company's own building, behind the company's own locked door. ERP and MES systems ran on servers in a back room, with terminals — and then PCs — on every desk and, eventually, on the shop floor itself. The plant's memory became a single integrated database, queryable in something close to real time, owned and controlled entirely by the company. This is the era that built the modern enterprise spine. It is also the era of the

brutal big-bang implementation, the eight-figure project, the consultant army, and the scars that many manufacturers still carry.

The fourth and current era is **cloud, and now edge-plus-cloud**. Beginning in the 2010s and accelerating straight through to this writing in 2026, the plant's memory moved off company hardware and into shared, elastic, internet-delivered infrastructure — and then, in a fascinating swing of the pendulum, partway back toward the floor in the form of edge devices that compute locally and sync to the cloud. The memory is now everywhere and nowhere: in a data center two states away, on a gateway bolted to a machine, in a phone in a supervisor's pocket. And critically, this is the era in which the memory finally became large enough, fast enough, and connected enough to support genuine reasoning — to host the machine-learning models that turn the system of record into a system of decision.

Four eras. Paper, mainframe, onsite server, cloud-and-edge. Notice the shape of it: the plant's memory traveled from the floor, up into a central temple, out onto the company's own servers, off into the cloud, and now partway back down to the floor again. That is not an accident. It is the first of the five great themes that run through this entire book, and I want to lay all five out now so you can watch for them.

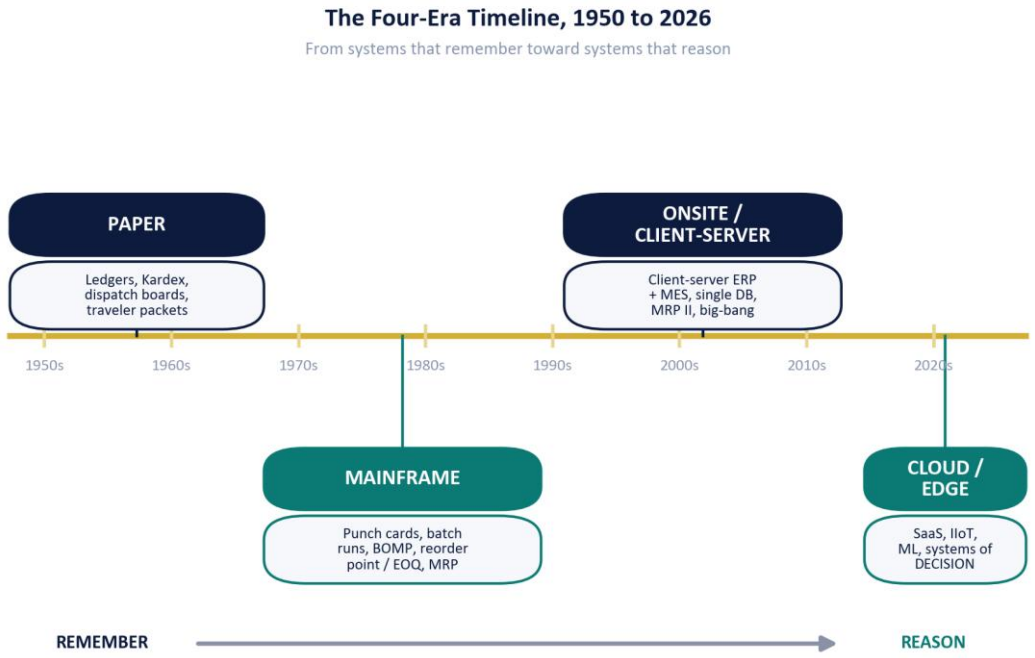

Figure 1.1 — *The four-era timeline, 1950 to 2026. The plant's memory migrates from paper on the floor, up into the centralized mainframe, out onto company-owned servers, and finally into the cloud and back partway toward the edge. The software families — MRP, MRP II, ERP, MES, APS — are layered onto the eras that birthed them. The bottom rail marks the long shift from systems that remember toward systems that reason.*

I will spend a full chapter on each of these eras, and several chapters on the software families that grew within them, so I will not exhaust them here. The timeline above is a map for the journey, not the

journey itself. But hold its shape in your mind: a memory that keeps moving, getting larger and faster and more connected at every step, until it is finally ready to do more than remember.

The five recurring themes

If the four eras are the *what*, the five themes are the *why* — the deeper currents that explain why the industry kept moving in the direction it did. I will thread these through every chapter, and I will name them when they surface, but you should meet them here at the start.

Theme one: the pendulum of centralization

The first theme is that the locus of the plant's intelligence swings back and forth between the center and the edge like a pendulum, and it never stops swinging. Paper was radically decentralized — every card, every board, every traveler was local. The mainframe yanked everything to the center, into one room, one machine, one priesthood. The client-server era pushed computing back out to desks and terminals while keeping the database central. The cloud re-centralized the data into enormous shared facilities. And now the edge is pushing computation back toward the machine itself, because some decisions — a vision system rejecting a bad part in fifty milliseconds — simply cannot wait for a round trip to a data center.

People love to narrate this as progress in a straight line, from primitive-local to advanced-central or the reverse. It is not a line. It is a pendulum, and the swing is driven by the eternal trade-off between the power of pooling everything in one place and the speed of deciding close to where the action is. Every era resolves that trade-off differently, with the tools it has. When you understand the pendulum, you stop being surprised that "edge computing" — which sounds like a brand-new 2020s idea — is in some ways a return to the locality of the dispatch board, only with a GPU bolted to it.

Theme two: the data gravity of manufacturing

The second theme is that every era is, underneath everything, a fight to get the right data to the right decision at the right time. I call this the data gravity of manufacturing, because data in a plant behaves like mass: it accumulates, it resists being moved, and it bends everything around it. The hardest problem in manufacturing software has never been computing the answer. The hardest problem has always been getting clean, current, trustworthy data from the place it is created — the receiving dock, the spindle, the inspection bench — to the place the decision is made, in time for the decision to matter.

Paper lost this fight to lag and transcription error. The mainframe lost it to batch delay. The onsite server narrowed the gap but fought integration wars between islands of data. The cloud and the edge are the latest combatants, and they are winning more of the fight than ever before — but the fight is the same fight. When you read about MRP's "nervousness" in Chapter 4, or the ERP integration scars in Chapter 6, or the IIoT data pipelines in Chapter 9, you are watching the same battle against data gravity, refought with new weapons.

Theme three: the human in the loop

The third theme is the one I care about most, and the one most often gotten wrong by people selling the next big thing. **Software has never removed the human from manufacturing. It has only ever changed what the human decides.**

Consider the planner. In the paper era, the planner spent the day physically reconciling cards and chasing shortages, and the *decision* they made was which fire to fight first. The mainframe took over the arithmetic of reorder points, and the planner was promoted: now they decided which of the machine's recommendations to trust and which to override. MRP took over the bill-of-material explosion, and the planner was promoted again: now they managed the planning parameters and wrestled with the nervousness of the system. APS took over the finite-capacity scheduling math, and the planner became a scenario analyst, asking what-if and choosing among optimized options. At every single step, the software ate the work below and the human moved up to the work above. The job title stayed the same. The job changed completely.

This matters enormously for how you should read the AI chapters, because the breathless version of the AI story is that the machines are coming for the jobs. The honest version, the version seventy-five years of evidence supports, is that AI is the next and largest promotion in a long series of promotions. The expeditor with the clipboard became the planner with the terminal became the analyst with the dashboard, and is now becoming the decision-architect who supervises a system that reasons. The human stays in the loop. The loop just keeps getting more interesting.

Theme four: risk as a feature

The fourth theme is my credo turned into a historical claim. In every era, a set of firms adopted the new technology early, paid the price in disruption and failed projects and 2 a.m. go-lives, and survived. Another set waited for the technology to mature, the price to fall, and the risk to be wrung out. And in every era, the early adopters who survived compounded an advantage that the laggards never caught.

This is not survivorship bias hand-waving. The mechanism is concrete. The early adopter does not just get the tool sooner. They get the *organizational learning* sooner — the rewired processes, the staff who actually understand the system, the data history that the models will later feed on, the painful but permanent shift in how the company thinks. By the time the laggard buys the now-safe, now-cheap technology, the early adopter has three to five years of compounding learning, and learning compounds in exactly the way that money does. Risk, paid for deliberately and absorbed with discipline, is the price of that head start. The firms that refuse all risk do not end up safe. They end up behind. I will show you this pattern in the mainframe era, in the MRP era, in the ERP era, and I will argue it is about to repeat, at higher stakes, in the AI era.

Theme five: AI as the connective tissue

The fifth theme is the destination. The long arc of these seventy-five years bends toward systems that decide, not merely systems that record — and artificial intelligence is the connective tissue that finally makes the bend possible. Every prior era added memory and added a little reasoning, but the reasoning

was brittle, rule-based, and narrow. The reorder point is a rule. The MRP explosion is a rule. The finite-capacity scheduler is a sophisticated bundle of rules. They are all reasoning of a kind, but they cannot learn, cannot generalize, and cannot handle the messy, high-dimensional reality of a real plant the way a human can.

Machine learning changes the nature of the reasoning. A model that learns from the plant's own accumulated memory — the very memory we spent sixty years building — can sense demand instead of merely forecasting it from a static formula, can predict a machine failure instead of waiting for it, can see a defect a human eye would miss, and can propose a schedule no rule could discover. AI does not replace the systems of record. It sits on top of them, drinks from them, and turns their memory into judgment. That is why I call it connective tissue: it is the layer that finally connects everything the plant remembers to the decisions the plant must make. The whole book builds toward this, and Chapter 12 is its synthesis, but you will find a marked moment in every single chapter — including this one, just below — where I point to exactly where AI bends the arc.

The layered capability map

The four eras tell you *when*. The five themes tell you *why*. But to understand the systems themselves — MRP, ERP, MES, APS, and the rest — you need a third lens, and that lens is a layered map of what the software actually *does*. I find it useful to picture the plant's software not as a timeline but as a stack of four capability layers, each one resting on the one below.

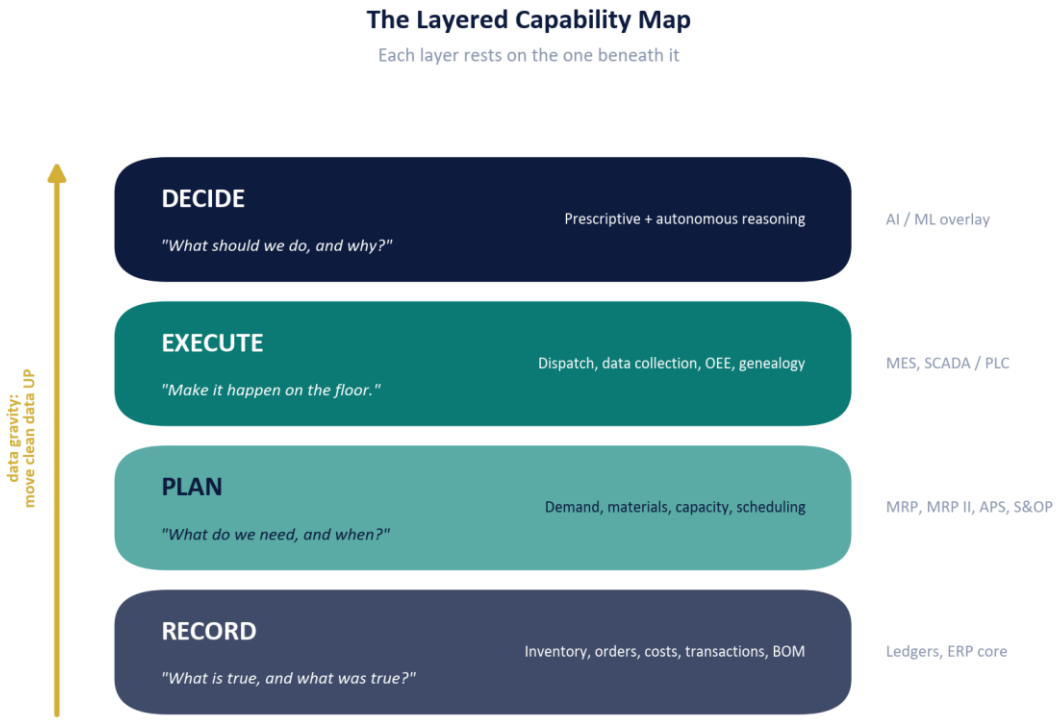

Figure 1.2 — *The layered capability map: Record, Plan, Execute, Decide. Each layer rests on the one beneath it; a plant cannot plan what it cannot record, cannot execute a plan it has not made, and cannot truly decide without all three layers feeding it. The systems of the four eras map onto these layers, and the arrow on the left is the data-gravity fight — the perennial struggle to move clean data upward to where each decision is made. AI and ML are arriving as an overlay on the Decide layer, drinking from everything below.*

The layers are cumulative, and they arrived roughly in order. The plant first learned to **record**: this is the work of the ledger, the Kardex card, and later the ERP database — the canonical answer to *what is true.* On top of recording, the plant learned to **plan**: reorder points, then MRP's dependent-demand logic, then MRP II's capacity loops, then APS's finite-capacity optimization — the answer to *what do we need and when.* On top of planning, the plant learned to **execute**: the MES layer that dispatches the work order to the machine, collects the data back, computes overall equipment effectiveness, and stitches together the genealogy of every part — the answer to *make it happen and prove it happened.* And now, on top of all three, the plant is learning to **decide**: the AI and ML overlays that take everything the lower layers remember and turn it into prescriptive, and increasingly autonomous, judgment — the answer to *what should we do, and why.*

You cannot skip a layer. This is the single most important thing to understand about the map and about the industry. A plant that cannot record cleanly cannot plan, because the plan is only as good as the inventory and BOM data beneath it. A plant that cannot plan cannot meaningfully execute, because execution without a plan is just reaction. And — this is the lesson of the current moment — a plant that cannot record and plan and execute cannot truly *decide* with AI, because the models that do the deciding are fed entirely by the memory of the layers below. The companies rushing to buy AI without having gotten their record-and-plan layers clean are trying to build the fourth floor of a building with no foundation. It happens constantly. It does not work, and Chapter 12 will explain in detail why.

Where AI changes the game

> **Where AI changes the game:** For seventy-five years, the four layers of the capability map had a hard ceiling. Record, Plan, and Execute were software's territory. *Decide* belonged almost entirely to people. The software could compute a reorder point or explode a bill of material, but the genuine judgment — which fire to fight, which recommendation to trust, which order to expedite, which machine to pull for maintenance, which lot to quarantine — lived in human heads, supported by dashboards but never made by the machine.

Artificial intelligence is the first technology that climbs into the Decide layer and stays there. And the reason it can is precisely the memory we spent six decades building. A machine-learning model is, at its heart, a thing that learns patterns from history — and manufacturing, after sixty years of obsessive recording, is drowning in history. Every transaction, every routing, every gauge reading, every defect, every late shipment, every machine cycle has been written down somewhere. The systems of record were, all along, unknowingly assembling the training data for the systems of decision.

Here is the bend, made concrete. A static safety-stock formula is a rule that never learns; a demand-sensing model reads point-of-sale signals, weather, and order velocity and adjusts itself continuously —

the Plan layer stops guessing and starts perceiving. A preventive-maintenance schedule based on calendar intervals wastes good parts and misses surprise failures; a predictive-maintenance model listens to the vibration and temperature of the actual machine and tells you it has four days, not four weeks — the Execute layer stops following a calendar and starts reading reality. A human inspector who can hold attention for only so many hours becomes a computer-vision system that examines every single part at line speed and never blinks — the quality function stops sampling and starts seeing everything. And a planner who could evaluate three scheduling scenarios before lunch becomes a planner who supervises a reinforcement-learning scheduler that evaluated three million overnight and surfaced the two worth a human's judgment.

In every one of those cases — and I will give you dozens more across this book — the AI does not remove the human. It removes the *ceiling*. The work that was impossible because no person could hold enough of the plant's memory in their head at once becomes possible, because the machine can hold all of it and reason over it in real time. This is why I am an optimist, and not a naive one. I have watched what happens when you take the ceiling off a good manufacturing team. They do not become unemployed. They become formidable.

The six rivers

There is one more lens I owe you before we begin the historical journey in earnest, and it is the one that keeps this book grounded in the business rather than the technology. Underneath all the software families and all the eras, a manufacturing enterprise is really just six great flows of work and value moving through it continuously, the way rivers move through a landscape. I call them the six rivers, and they are the deep subject of Chapter 10, but you must meet them now because every system we discuss exists to serve one or more of them.

A river is a business process: it has a source, it flows in one direction, it gathers tributaries along the way, and it empties into an outcome. Manufacturing has six of them, and together they are the entire metabolism of the enterprise.

The first river is **Order-to-Cash (O2C)**. It begins when a customer wants something and ends when the money is in the bank. Quote, order, fulfill, invoice, collect. This is the river that pays for everything else, and it is the river the customer actually experiences. When O2C runs clean, the customer gets what they ordered, when they were promised, and the invoice is correct. When it runs dirty, you have angry customers, disputed invoices, and cash that should be yours sitting in someone else's account.

The second river is **Procure-to-Pay (P2P)**. It is O2C run in reverse, from the company's side as a buyer. Requisition, purchase order, receipt, supplier invoice, payment. This is how the plant feeds itself — the raw material, the components, the services, the everything-you-don't-make-yourself. P2P is where the data gravity of the supply base lands on you, and in an era of fragile global supply chains, it is the river that keeps plant managers up at night.

The third river is **Record-to-Report (R2R)**. It is the financial conscience of the enterprise: transaction capture, the period close, financial reporting. R2R is the river that turns all the other rivers' activity into

a truthful account of what happened and where the money went. It is the river the auditors swim in, the river the bank and the board and the tax authority care about. R2R is pure system-of-record work — the very heart of the Record layer — and its evolution from hand-posted ledgers to real-time consolidation is one of the great quiet stories of this book.

The fourth river is **Hire-to-Retire (H2R)**. It is the flow of human beings through the enterprise: recruit, onboard, develop, and eventually offboard the workforce. People often forget H2R when they think about manufacturing software, because it feels like an HR concern rather than a plant concern. That is a mistake. The plant runs on skilled people, and the river that brings them in, grows them, schedules them, certifies them, and lets them go is every bit as operational as the river that moves steel. In a labor-constrained age, H2R may be the most strategically important river of all.

The fifth river is **Plan-to-Produce (P2P-Prod)** — and to avoid confusion with Procure-to-Pay I will always spell it out or mark it clearly. This is the river that runs straight through the heart of the factory: demand, plan, schedule, execute, and cost the work on the floor. It is the river that MRP, MRP II, APS, and MES were all built to serve. When people picture "manufacturing software," this is usually the river they are picturing. It is the most technically elaborate of the six, and it touches the Plan and Execute layers of our capability map most directly.

The sixth river is **Issue-to-Resolution (I2R)**. It begins with a defect, a complaint, an incident, a deviation — something went wrong — and it ends with the problem genuinely closed: triaged, root-caused, corrected, and prevented from recurring. In the quality world this is the home of corrective and preventive action, CAPA, and it is the river that compliance regimes care about most. I2R is the plant's immune system. A plant with a strong I2R river learns from every failure; a plant with a weak one keeps making the same mistake and calling it bad luck.

The six rivers form an interlocking circuit — supply feeds the plant, the plant satisfies demand, finance accounts for all of it, quality learns from what breaks, and people run the whole thing. (The formal swim lane and value-circuit diagrams for these rivers appear in Chapter 10 as Figures 10.1 through 10.3.)

I introduce the rivers here, lightly, for one reason: so that as we travel through the eras, you can always ask the grounding question — *which river does this system serve, and is it making that river run cleaner or dirtier?* Technology that does not eventually make one of the six rivers run cleaner is technology a manufacturer should be suspicious of. The rivers are the point. The software is only ever a means.

The system families you are about to meet

Let me now name, briefly, the cast of software families that will occupy the middle of this book, so the acronyms do not ambush you. Each one will get its proper introduction in its own chapter; this is only a roll call.

MRP — Material Requirements Planning. The 1970s breakthrough, built on Joseph Orlicky's insight that the demand for components is not forecast but *calculated* from the demand for finished goods. MRP explodes a bill of material, nets gross requirements against what you already have, and time-phases the

result into a schedule of what to order and make, and when. It was the first software that truly served the Plan layer for dependent demand. Chapter 4 is its home.

MRP II — Manufacturing Resource Planning. The 1980s evolution that closed the loop, adding capacity planning, shop-floor control, sales-and-operations planning, and financial integration to MRP's material logic. Where MRP asked *what do we need,* MRP II asked *and can we actually make it, and what will it cost.* Chapter 5.

ERP — Enterprise Resource Planning. The 1990s integration wave that put all the rivers — finance, procurement, sales, HR, and manufacturing — onto a single database with a single version of the truth. ERP is the great system of record, the enterprise spine, and the system that defined the onsite-server era. Chapter 6.

MES — Manufacturing Execution System. The system that lives in the gap between the ERP's plan and the machine's reality. MES dispatches work orders to the floor, collects data back from operators and machines, computes effectiveness, and maintains the genealogy that lets you trace any part back to its origins. It is the heart of the Execute layer. Chapter 7.

APS — Advanced Planning and Scheduling. The system that fixed MRP's most embarrassing assumption — that capacity is infinite — by scheduling against finite, real-world constraints and optimizing toward business objectives. APS is what happens when the Plan layer gets serious about reality. Chapter 8.

QMS and the compliance regimes. Quality management systems, and the standards they serve — ISO 9001, IATF 16949, AS9100, ISO 13485, FDA 21 CFR Part 820, and the rest — turn the Issue-to-Resolution river into auditable, provable practice. This is the world of electronic batch records, audit trails, and validation. Chapter 11.

AI and ML overlays. The newest family, and the one that does not replace any of the above but sits atop all of them, drinking from their memory to produce decisions. Supervised, unsupervised, and reinforcement learning; predictive maintenance; computer-vision quality; demand sensing; generative copilots; digital twins; and the early agentic systems that begin to act on their own conclusions. Chapter 12, with a marked moment in every chapter before it.

That is the cast. Notice that it is cumulative, not sequential — MES did not replace ERP, and APS did not replace MRP. Each new family was added to the stack, taking over a layer of work and promoting the humans above it, exactly as the human-in-the-loop theme predicts. The plant of 2026 runs all of them at once, and the interesting questions are now about how they connect, where the boundaries between them are dissolving, and what the AI overlay does to the whole arrangement.

Era at a glance

Before we descend into the paper plant of the 1950s, let me give you one consolidated reference — a single table you can return to whenever you lose your bearings in the chapters ahead. It compares the four eras across the dimensions that matter most: when they ruled, the dominant technology, what the

plant could remember and decide, and the key limitation that eventually drove the industry to the next era.

Table 1.1 — *Era at a glance: what each era could remember and decide.*

Era	Approx. dates	Dominant technology	What it could remember / decide	Key limitation
Paper	1950s–1960s	Ledgers, Kardex cards, dispatch boards, traveler packets, hand posting	Remembered the local state of inventory, jobs, and accounts on physical media; decisions made entirely by people reading the artifacts	Slow, error-prone, stubbornly local — to know the plant you had to walk it; no real-time visibility
Mainframe	1960s–1970s	Centralized batch computers (IBM 1401, System/360), punch cards, BOMP, reorder-point and EOQ automation	Remembered far more than any clerical staff and computed reorder points and BOM explosions at scale	Batch delay (answers came the next morning or after the weekend); a costly priesthood stood between users and data
Onsite / on-prem server	1990s–2010s	Client-server ERP and MES on company-owned hardware; single integrated database; three-tier architecture	Remembered the whole enterprise on one queryable database in near-real time; could plan, execute, and report in an integrated way	Capital-intensive, brutal big-bang implementations; integration wars between systems; company owned all the risk and maintenance
Cloud + edge	2010s–2026	SaaS, multi-tenancy, mobile, IIoT, edge gateways, ML/AI overlays	Remembers everywhere at once, in real time, and finally *reasons* — predictive, prescriptive, and increasingly autonomous decisions	Connectivity and cybersecurity dependence; data-governance and model-trust challenges; the discipline AI demands of the layers beneath it

Read across any row and you can see the bargain each era struck. Read down any column and you can watch the long progression: the technology gets more powerful, the memory gets larger and faster and more connected, and — only in the final row — the verb finally changes from *remember* to *reason.* That change in the last row is the whole reason I wrote this book. Everything before it is the seventy-year run-up. Everything after it is the future we are now building.

A scene from each era, so the abstractions have a body

I have spent a lot of words on arcs and layers and themes, and I owe you something more concrete before we go on. Let me put a single human scene in each of the four eras, because the whole argument of this book lives or dies on whether you can feel the difference between a plant that remembers poorly and one that remembers well, between a plant that only records and one that begins to reason.

In the **paper plant**, picture the expeditor. She comes in before the first shift with a clipboard and a list that is already out of date, because the list was printed yesterday and the world moved overnight. Her job is to walk the floor and find out what is actually true — which jobs are stuck waiting on a part, which machine is down, which order the customer just called screaming about. She is, in the most literal sense, the plant's query engine. The data exists, scattered across travelers and cards and the heads of the operators, and her legs and her memory are the only mechanism for assembling it into a picture. A good expeditor is worth her weight in gold and utterly irreplaceable, which is precisely the problem: the plant's intelligence walks out the door every night and retires one day for good. The reasoning is superb and entirely human; the memory is real but trapped in artifacts that cannot talk to each other.

In the **mainframe plant**, picture the same shortage, but now picture the planner waiting for the nightly run. He has fed his changes into the system on punch cards or, later, a terminal, and now he waits. The machine in the glass room is chewing through the reorder-point calculations and the bill-of-material explosion for the entire plant, and it will not be done until morning. He cannot ask it a question now. He cannot say "just show me part 4471-A." Batch computing does not work that way; you get the whole report or you get nothing, and you get it on the machine's schedule, not yours. When the report lands on his desk at 7 a.m., it is magnificent — pages and pages of requirements no clerical team could ever have computed — and it is already a few hours stale, and it is written in a format only he and the data-processing priesthood can read. The memory just got vast. The reasoning is still entirely his, and now he must reason against a snapshot taken in the dark.

In the **onsite-server plant**, picture the planner again, but now she has a terminal on her desk and the answer is *right there.* She types the part number and the system tells her, in something close to real time, the on-hand quantity, the open orders, the lead time, the cost, the customers waiting on it. The integration wars are real — the quality data lives in one system and the maintenance data in another and getting them to agree is somebody's full-time job — but the core enterprise memory is finally unified, queryable, and current. She no longer waits for the nightly run; she interrogates the database whenever she likes. The promotion is visible: she spends almost no time assembling the picture and almost all of it deciding what to do about it. The memory is now fast and integrated. The reasoning is still hers, but the software has cleared the underbrush so she can do more of it.

In the **cloud-and-edge plant**, picture the supervisor with a phone. The vibration sensor on the stamping press has been feeding a model in the cloud for eight months, and this morning the model sends an alert: bearing degradation, estimated four days of useful life, recommend pulling the press for service Thursday at noon when the schedule has slack. The supervisor did not compute this. No human could have — it required holding eight months of high-frequency sensor data in mind and comparing it against

the failure signatures of a thousand similar machines. The supervisor's job is now to *judge the recommendation:* is Thursday really the slack window, is the spare bearing in stock, is there a customer order that changes the calculus. The memory has become so deep and so fast that it can finally support reasoning the software does itself — and the human has been promoted, one more time, into the supervisor of a machine that reasons. That is the arc, in four scenes, with a body on it.

The pendulum, looked at more closely

I want to return to the first theme, the pendulum of centralization, because it is the one most often mistaken for simple progress, and getting it wrong will make the later chapters confusing.

The naive story is that computing moved steadily from primitive and local toward advanced and central, or, in the breathless cloud-era version, that everything is "moving to the cloud" as a final destination. Neither is true. What actually happens is a repeated negotiation between two real and permanent forces. The first force pulls toward the center: pooling data and computation in one place makes it cheaper, more consistent, more powerful, and easier to govern. One database, one truth, one place to secure and back up and reason over. The second force pulls toward the edge: deciding close to where the action happens is faster, more resilient to a broken connection, and sometimes the only physically possible option. A vision system that must reject a defective part before it leaves the station has milliseconds; it cannot wait for a round trip to a data center two states away. A plant that loses its internet connection cannot stop making parts.

Every era resolves this negotiation with the tools it has, and the resolution swings back and forth. Paper was pure edge — maximally local, maximally resilient, hopeless at pooling. The mainframe swung hard to the center — maximal pooling, at the cost of batch delay and a priesthood. Client-server split the difference, pushing presentation and some logic out to the desktop while keeping the database central. The cloud swung back hard to the center, pooling data at a scale and elasticity no company could achieve alone. And now the edge is swinging back out, not because the cloud failed but because some decisions are too fast or too critical to centralize, and because the sensors on the floor now generate more data than it makes sense to ship anywhere. The modern answer is not cloud *or* edge. It is cloud *and* edge — pool what benefits from pooling, decide locally what must be decided locally, and synchronize the two. That hybrid is the current resting point of a pendulum that has never once stopped, and it will swing again.

Why does this matter for a manufacturer making decisions in 2026? Because it tells you not to fall in love with a topology. The vendors will tell you that their architecture is the final word, that everything belongs in their cloud or on their edge box. The history says otherwise. The right question is never "central or local" in the abstract; it is "where does *this particular decision* need to be made, given how fast it must happen and how much it would cost to be wrong." Some of your intelligence belongs in the cloud and some belongs bolted to a machine, and a mature plant deliberately places each capability where the physics and the economics say it should go. The pendulum is not a problem to be solved. It is a trade-off to be managed, forever.

Data gravity, and why clean data is the whole ballgame

The second theme deserves the same closer look, because if there is a single lesson I would tattoo on the forearm of every manufacturer chasing the AI dream, it is this one: the bottleneck has never been the algorithm. The bottleneck has always been the data.

Think again about what "data gravity" means in a plant. Data is created at thousands of points — every receipt at the dock, every transaction at the stockroom window, every cycle of every machine, every reading at every inspection bench, every clock-in and clock-out, every customer call. Each of those points creates a small fact. And every decision worth making requires assembling many of those small facts, cleanly and currently, into a picture. The fight of every era is the fight to overcome the gravity that wants to keep each fact stuck where it was created, siloed and stale and slightly wrong.

Paper lost this fight at the transcription. A number written on a traveler had to be carried, read, re-keyed, and posted, and at every hop it could be smudged, transposed, delayed, or simply lost. The mainframe won the computation but lost the timeliness, because the facts could only be assembled in the nightly batch. The onsite server won the timeliness within a single system but fought a war between systems — the ERP knew the order but not the machine's condition, the MES knew the machine but not the customer's credit, and reconciling them was a permanent, expensive chore. The cloud and the edge are winning more of the fight than ever, because they can ingest high-frequency data from everywhere and assemble it continuously. But notice: they have not changed the *nature* of the fight. They have just brought heavier weapons to it.

And here is the part that matters for the future. Machine learning does not relax the data-gravity problem. It makes it the *only* problem. A model is a thing that learns from history, and it learns exactly the history you feed it — including all the gaps, errors, biases, and staleness. Feed a demand-sensing model dirty sales history and it will sense demand confidently and wrongly. Feed a predictive-maintenance model sensor data with unlabeled downtime and it will predict failures it should not and miss the ones it should. The single most common reason an AI project fails in a plant is not that the model was bad. It is that the data beneath it was never clean enough to reason over, because the Record and Plan layers of the capability map were never properly built. This is why I keep insisting that you cannot skip a layer. The AI dream is real, and it is closer than the skeptics think — but it is built entirely on the unglamorous, decades-long discipline of getting your data right. The firms that win the AI era will be the ones that won the data-discipline era first, quietly, while no one was watching.

The human in the loop, defended at length

Of all five themes, the human-in-the-loop is the one I will defend most fiercely, because it is the one the doom-merchants get most wrong, and because getting it wrong leads manufacturers to make genuinely bad decisions out of fear.

Here is the historical claim, stated as strongly as I can make it: in seventy-five years, no wave of manufacturing software has ever, on net, removed the human from the plant. Every wave has moved

the human up. The arithmetic clerk did not vanish when the mainframe computed reorder points; the clerk became the planner who set and judged the parameters. The planner did not vanish when MRP exploded the bill of material; the planner became the manager of planning policy and the tamer of system nervousness. The scheduler did not vanish when APS optimized the finite-capacity schedule; the scheduler became the scenario analyst choosing among optimized options. At every step, the software ate the layer of work directly below the human, and the human stepped up onto the layer above — work that was previously impossible because there was no time or capacity to do it.

Why does this keep happening, era after era, with such regularity? Because the work above is always harder, more judgment-laden, and more valuable than the work below, and there is always more of it than anyone could previously get to. Manufacturing is not a fixed pie of tasks that software slices away until none are left for people. It is an effectively bottomless well of decisions, most of which never got made well because no one had the time. When software takes over the routine, it does not leave a vacuum. It exposes the next layer of decisions that were always there, waiting, unattended. The plant manager who is freed from chasing shortages does not go home early. He starts working on the problems he never had time for — the supplier strategy, the layout, the training, the thousand improvements that were always one level of abstraction above the firefighting.

This is exactly why I am unafraid of AI in the plant, and why you should be unafraid too. AI is the largest promotion in the series, not the end of it. It takes over the deepest layer of routine reasoning we have yet automated — the demand sensing, the failure prediction, the visual inspection, the schedule optimization — and it exposes a new layer of human work above it: judging the machine's recommendations, governing the models, deciding what the plant should *become*, handling the genuinely novel situations no model has seen. The human in the loop does not disappear. The loop gets larger, the stakes get higher, and the human stands at the most interesting point in it that has ever existed. I have watched good teams get this promotion, and I will tell you what it looks like: not unemployment, but a kind of professional exhilaration, the feeling of finally being able to work on what actually matters.

There is a hard-nosed corollary, and I will not pretend otherwise. The promotion is real, but it is not automatic, and it is not free. The clerk who would not learn to be a planner did struggle. The plant that treated each wave as a way to cut headcount rather than to elevate capability did get the worst of both worlds — it lost the people who held the tacit knowledge and gained a system no one was elevated to run well. The human-in-the-loop theme is a description of what *can* happen and, on net across the industry, *has* happened. Whether it happens in your plant depends on whether you treat the new technology as a way to promote your people or a way to be rid of them. I have always counseled the former, not out of sentiment, but because the plants that promote their people are the ones that actually capture the advantage. The tacit knowledge in your operators' heads is, increasingly, the training data and the judgment layer that makes the AI work. Discard the people and you discard the very thing that makes the machine valuable.

Risk as a feature, in the cold light of evidence

Let me sharpen the fourth theme into something you could actually act on, because "embrace risk" is useless as advice if it is just a slogan.

The claim is not that all risk is good or that early adoption always pays. Plenty of early adopters of plenty of technologies lost their shirts, and I am not going to insult your intelligence by pretending otherwise. The claim is narrower and, I think, more useful: across the four eras of manufacturing software, the firms that adopted a *fundamentally sound* technology early — and had the operational discipline to survive the disruption — compounded an advantage that the late majority never recovered. The two conditions matter. The technology has to be fundamentally sound, not a fad, which requires judgment about what is real. And the firm has to survive the adoption, which requires operational discipline and a tolerance for the 2 a.m. go-live. When both conditions hold, early adoption is not gambling. It is investing at the bottom of a learning curve.

The mechanism, again, is compounding learning. When you adopt a sound new system three years before your competitor, you do not just get three years of the system's direct benefits. You get three years of *learning how to use it* — three years of rewiring your processes around it, of training your people on it, of accumulating the data history that the next generation of tools will feed on, of making and recovering from the mistakes that the laggard has not yet made. Learning compounds. By the time the laggard buys the now-mature, now-cheap, now-safe version of the technology, you are not three years ahead in features. You are three years ahead in *organizational capability*, and that gap is far harder to close than a feature gap, because the laggard now has to climb the entire learning curve you have already climbed, while you have moved on to the next curve.

This is why I say the biggest risk in manufacturing software is standing still. The manager who waits for certainty is not avoiding risk. He is choosing a particular risk — the slow, invisible, deferred risk of falling behind on the learning curve — over the fast, visible, immediate risk of a hard implementation. He feels safer because the risk he chose does not announce itself with a 2 a.m. crisis. But it is, in my experience, the deadlier of the two, precisely because it is silent. By the time the consequences of standing still become visible, the gap is often too large to close.

So when I tell you to be risk-positive, I am not telling you to be reckless. I am telling you to judge soundness rigorously, build the operational discipline to survive adoption, and then *move* — earlier than feels comfortable, because comfort in this field is a trailing indicator of advantage already lost. That posture is about to be tested harder than ever, because the AI wave is the steepest learning curve any of us has faced, and the firms that start climbing it now, while the tools are still rough, will be at the top of it while their competitors are still debating whether it is real.

AI as connective tissue, and what "systems of decision" really means

The fifth theme is the destination, so let me say more precisely what I mean by the bend from systems of record to systems of decision, because the phrase is easy to say and easy to misunderstand.

A system of record answers *what is true.* It is a disciplined, trustworthy memory. The ERP that knows your inventory, the MES that knows your machine's genealogy, the financial ledger that knows where the money went — these are systems of record, and they are magnificent achievements. But a system of record is fundamentally passive. It waits to be queried. It will tell you the truth if you ask, but it will not tell you what to do about it. The reasoning, the judgment, the *so what,* has always had to come from somewhere else — historically, from a human being.

A system of decision answers *what should we do, and why.* It takes everything the systems of record remember, adds everything it can forecast and simulate, and produces a recommendation — or, increasingly, an action. It is active. It does not wait to be asked; it watches, it reasons, and it proposes. The demand-sensing system that adjusts the plan before you ask it to. The predictive-maintenance model that opens a work order before the machine fails. The scheduler that re-optimizes the floor when a rush order lands and tells you what it changed and why. These are systems of decision, and they are the new thing under the sun.

Artificial intelligence is the connective tissue that makes the bend possible, and the word "connective" is exact. AI does not sit beside the systems of record as a separate tool. It sits *on top* of them and *connects* them — connects the inventory data to the demand signal to the supplier's reliability to the customer's priority, and reasons across all of it at once, which no human ever could because no human can hold that many facts in mind simultaneously. The systems of record are the organs; the AI is the nervous system that finally lets the organs act as one body. And the reason it can do this now, and could not before, is the simplest reason of all: the organs are finally rich enough. Sixty years of obsessive recording built a memory deep enough to reason over. The systems of record were, without knowing it, assembling the substrate for the systems of decision the whole time.

I want to be careful here, because optimism without honesty is just salesmanship. The bend toward decision is real, but it is gradual, and it passes through stages. First the machine describes what happened — that is the dashboard, the report, the descriptive layer we have had for decades. Then it predicts what will happen — that is the forecasting and the failure prediction, the predictive layer we are deep into now. Then it prescribes what to do about it — the prescriptive layer, where the machine recommends and the human approves, which is where the leading plants live today. And finally, in narrow and carefully bounded domains, it acts on its own conclusions — the autonomous layer, which is just beginning and which Chapter 13 is devoted to. The bend is not a cliff. It is a curve we are climbing one stage at a time, and the human stays in the loop at every stage, with the machine handling more of the reasoning and the human handling more of the judgment about the reasoning. That is the future I believe in, and it is the future this entire book is built to explain.

How to read this book

A few practical notes before we begin. This book moves roughly chronologically, but not slavishly — the eras overlap, the software families bleed across decades, and I will frequently jump forward to show you where a thread leads. Chapters 2 through 9 walk the four eras and the five software families in order: the paper plant, the mainframe, MRP, MRP II, ERP, MES, APS, and the cloud. Chapter 10 steps out of the chronology to follow the six rivers across all four eras at once, which is the best way to see how the business process — not the technology — is the thing that actually evolved. Chapter 11 takes up quality and compliance, the discipline that turns the Issue-to-Resolution river into provable practice. Chapter 12 is the synthesis on AI and machine learning, where all the threads converge. And Chapter 13 is the futurist payoff — my best account of the self-driving plant we are building and the bet on risk that will decide who gets there first.

In every chapter, watch for three things. Watch for the five themes surfacing — I will name them. Watch for the six rivers running underneath the technology. And watch for the marked moment, "**Where AI changes the game**," where I point to exactly how the newest layer bends the old arc. If you read for those three things, you will come away with the structure of the whole field in your head, which is worth far more than any individual fact about any individual system.

And read it, please, with my disposition in mind. I am not a neutral chronicler. I am a partisan for the future, a man who has bet his working life on the proposition that the manufacturers who lean into new capability will out-compete the ones who hide from it. I will give you the costs honestly — the failed projects, the 2 a.m. go-lives, the scars — because pretending change is painless is a lie that helps no one. But I will always come back to the same conclusion, because seventy-five years of evidence keeps pointing at it: the biggest risk in this business is standing still, and the plant that learns to reason will eat the plant that only remembers.

Let us begin where the memory began — on paper.

The thread forward

For all the power of the mainframe and the cloud, the plant's memory did not begin on silicon. It began on cards and boards and in the ledgers that clerks posted by hand. Before we can appreciate what the machine gave us, we have to stand in the plant that had no machine at all — the paper plant, with its traveler packets riding along with every job, its Kardex drawers, its dispatch board glowing with colored tickets, and its expeditors who carried the schedule in their heads and their gut. It was slower than anything we would tolerate today. It was also, in ways we have half-forgotten and are now trying to recover, astonishingly resilient. Chapter 2 takes us onto that floor.

End of sample