

TEAM TESS PLAYBOOK · THE SUBSCRIPTION EDUCATION SERIES

Raise the Rate

The Complete Lesson Plan Book

Project management & Lean Six Sigma, taught from the industries that proved it. Fourteen modules. One build each. One honest number, built piece by piece on your own work, until, at the capstone, your gauge page reads your rate and the distance it has travelled.

FINAL EDITION · OCTOBER 2026

Agentic Armor LLC

Team Tess Playbook subscription education series

What this book is

This book prints all fourteen modules complete — the lesson plan and the full lesson for every module, Module 1 through the Module 14 capstone. Nothing in this edition is a plan standing in place of a lesson: for each module you will find its complete lesson plan — its one build, its objectives, its source story and the evidence grade that story carries, the piece it builds in your gauge, its worked example, and its field exercise — followed by the full lesson itself, taught in the module's four beats.

Module 1, “Your Number,” is printed complete, as it was in the first edition of this book. Modules 2 through 14 follow in their taught editions, each printed whole: lesson plan and full lesson together, in course order. The book closes with the summary that assembles your gauge, the delta, and the closing page.

How this book is organized

Read the introduction first. Part I states the doctrine the whole course teaches, in eight articles. Part II explains how the course runs: one build per module, in the parts printed for every module — the lesson (Part A), the worked example (Part B), and the field exercise (Part D); Part C, the module's audio narration, belongs to the taught editions and is not reprinted. The course is worked at your own pace, with one artifact logged per module and your gauge built piece by piece across all fourteen modules. A note on what the subtitle does not promise stands in Part II: this book teaches a Lean operating discipline and borrows two Six Sigma terms (yield, and defect counting); DMAIC, statistical process control, capability analysis, FMEA, and inferential statistics are not in this book, and project management's cost, schedule, and risk machinery appears only where near-miss reporting touches it. Then read the modules in order: each module's exercise builds on the artifact of the module before it, and the capstone assembles the gauge page from parts you have already been running for weeks.

What this book is not

It is not a credential. This course issues no belts and no certificates of attendance. Its one completion instrument is your own gauge page and the change in it — your delta — read at the capstone against the baseline you set in Module 1.

A note on evidence grades, used throughout this book. Every source story is graded exactly as the study behind this course graded it: RECORD: a named, checkable source; INFERENCE: a reasoned transfer from the record, labelled as such; FOLKLORE: widely repeated, with no primary source, told only when it is instructive and labelled folklore on its face. Where a story carries a split grade, the plan prints the split. No plan in this book upgrades a grade. The sources behind the graded stories are cited in the Sources and Bibliography at the back of this book.

Final Edition, October 27, 2026: the October 27 revised edition, corrected on the evidence throughout: claims conformed to their sources, a bibliography added, and the closing conformed to the course as it now runs. Every correction is logged in the edition's fix record.

Contents

Part I: The Doctrine: why nobody downstairs knows why, and the eight articles	Introduction	4
Part II: How the course runs: the four parts, the pace, the gauge you build, the completion instrument	Introduction	6
Module 1: Your Number The house composite; build your gauge.	Chapter 1	9
Module 2: Commitments You Can Count	Chapter 2	19
Module 3: Less Started, More Finished	Chapter 3	27
Module 4: Make the Error Impossible	Chapter 4	34
Module 5: Stop the Line	Chapter 5	42
Module 6: Report the Almost	Chapter 6	49
Module 7: Read It Back	Chapter 7	55
Module 8: Verified by Another	Chapter 8	62
Module 9: First Pass, Every Pass	Chapter 9	68
Module 10: The Ten-Minute Turn	Chapter 10	77
Module 11: One Plan, Four Gates	Chapter 11	84
Module 12: Small Batches, Fast Signal	Chapter 12	91
Module 13: The Weekly Review	Chapter 13	98
Module 14: Registers That Remember (Capstone)	Chapter 14	105
Your gauge, assembled: the one-page summary	After the plans	113
The delta	After the plans	114
What happens next, and what this book cost to make	Closing page	115
Sources and Bibliography: the published sources the graded stories rest on	Back of the book	116

Every module in this contents list is printed complete in this book: its lesson plan and its full lesson. Module 1 through Module 14, in order, with the introduction before them and the summary, the delta, and the closing page after them.

The Doctrine

The standard the whole course teaches, stated once, in plain words, before the first lesson

The compartment disease

In a compartmentalized organization, decisions are made in separate rooms, at separate levels, and the reasons for them do not travel. Information moves in fragments, if it moves at all. So the people who carry the work (six levels down, and all the way across) end up asking whether they are in compliance: with a regulation, a standard, a directive. They ask whether, when they should be able to say why, why the requirement exists, in their own words, at their level.

Understanding sits at the top, held between a management layer and the bodies that govern, and it never reaches the floor. That is not a training gap. Training can hand a person an instruction; it cannot, by itself, make the reason for the instruction a thing the whole building holds. The gap is a doctrine gap: no standard in the building makes the why a deliverable owed to every level the work touches.

This course exists to close that gap in one operation at a time, yours. The doctrine below is the standard it teaches. Eight articles. Together they fit in your head, because a standard a practitioner cannot carry in their head will not survive contact with their building.

The eight articles

Article 1: End state first. Every effort opens from a committed end state: a named owner, a written end state, a date. Work missing any of the three is activity, not a project, and the doctrine does not pretend to manage it.

Article 2: One form, unchanged. All work reports in one uniform form (**Status · Owner · Next milestone · Gate · Metric**) and a fact written at the bottom reaches the top in the words it was written in, never summarized into somebody else's credit. The form is how truth travels; a workstream that reports in its own shape has left the standard.

Article 3: First-pass truth. Work counts when it crosses its delivery point clean the first time. Work that lands late, is never handed over, or lands with an error found afterward is an escape, written down permanently. Repairs are new work, never erasures.

Article 4: The gauge is the scoreboard. Every operation running this doctrine keeps one number: a fixed-formula composite of stage yields, counted under a permanent rule, impossible to improve by redefining it. The formula changes only in the open, by a dated amendment printed on the gauge's face, with the old series preserved.

Article 5: Claim bars beside the rate, held by named humans. What may be claimed (published, advertised, spent against) is governed by claim bars: standards stated in advance, counted in units, that print beside the number and never blend into it. (This book keeps its words separate: a claim bar is this article's instrument; a gate is a ruling conference on an effort's plan, Module 11; the step from one module to the next is the opening.) And four decision classes never leave human hands: **spend, send, publish, change**. The system recommends, prices, and reports; a named person rules.

Article 6: Improvement arrives one small practice at a time. Change enters as one small, sourced practice at a time, restated by every process it touches, and judged on whether it moves the number it was adopted to move. An adoption that does not earn its place within its trial window is set aside, in the open, without blame. The list of practices an operation has declined (and why) is half the product: it is the record of fashions the operation will never have to un-adopt.

Article 7: The Tendrils Principle: WHY at every level. The name is the shape of the requirement: a reason reaches outward, to everyone the work touches, and downward, into every level that carries it, or it has not reached at all. Comprehension is a deliverable of the system, owed at every level the work touches, down through the whole organization and all the way across it. A requirement is not discharged when it is transmitted; it is discharged when the person at the working level can state why it exists, not only that it exists. The doctrine's test is named **the Level Test**: pick any level, ask why, the answer must exist there, in that level's own words, or the system above it has failed, not the person. This article is the doctrine's answer to the compartment disease: decisions made in sealed rooms produce questions at the floor; reasons carried outward (by the one form of Article 2 and a standing rhythm of review) produce understanding at the floor. An operation in which, six levels down, a practitioner must ask whether they are in compliance has broken this article. The doctrine's aim is the operation where they can say why they are.

Article 8: Adoption by thread. No one is rebuilt at once. A practitioner, a team, an organization adopts this doctrine one thread at a time (one commitment practice, one report form, one honest number) in their own order, at their own pace, whole between steps. The person pulls the threads; the system never pulls the person.

What the doctrine is not. It is not a survey of project management, and it does not compete with the established bodies of knowledge on their ground. It teaches public methods (Toyota's system, the Last Planner, aviation's reporting culture, independent verification, queueing discipline) with their sources named, and it teaches the discipline of adopting them honestly. Its claim is narrow: an operation run on these eight articles will know its own number, will know why at every level, and will improve on evidence. That is the whole claim.

How the Course Runs

One build per module · one artifact per module · one number the whole way

One build per module, in four parts

Each module makes one build: one artifact, built whole, in your operation. Three modules carry a named second thread, and their plans say so on their faces rather than smuggling it (Module 4 builds the exception queue beside its validation, Module 11 carries standard work beside its plan of record, Module 13 names the steering inputs its review runs on); the rest teach one thing. Every module runs the same four parts, because the structure is itself a lesson in standard work:

1. **Part A: the lesson.** The source story, the concept taught from the industry that proved it, in that industry's real history, the plant, the construction site, the hospital floor, with the evidence graded on its face: RECORD, INFERENCE, or FOLKLORE, as defined at the front of this book; and the translation, the concept restated for your operation, naming the gauge component the concept moves, so the course never drifts into inspiration detached from measurement.
2. **Part B: the worked example.** A worked operation, shown at the WHAT level: what was produced, and what it counted. (WHAT level, defined once here: a claim stated at the WHAT level rests on the visible product and its count, never on the internal machinery behind it.) The form, the numbers, the defects, anonymized, with its real figures, including the unflattering ones. Where an attempt failed or is incomplete, the example says so.
3. **Part C: the narration.** The lesson read aloud, one voice. Part C belongs to the taught editions; it is not reprinted in this book, and nothing in this book depends on it.
4. **Part D: the field exercise.** One exercise, run on your own work, producing one artifact. The artifact is the module. A reader who only reads has completed nothing: the app logs the artifact, the next module opens, and its exercise builds on yours.

The pace, and the opening

One module at a time, at your own pace. The next module opens when you have logged the current module's field-exercise artifact, **logged, not graded**. The app records the artifact; nobody marks homework. Modules already opened stay open; the library is cumulative.

Why not all at once? Each exercise needs real work to be real. A reliability score requires a week of commitments kept or missed; a census of open work means something only after time lived under a limit. The exercises are sized in weeks of your operation's life, but the calendar is yours: the opening follows the artifact, never the date.

Formats per module (design figures, stated from the course's format specification, the series has not yet run, and these are design figures, not measured results): the written lesson, about **20 minutes** of study reading · an audio version for the commute, about **19-24 minutes** per module, one voice, at a measured narration pace · the worked example, inside the lesson and as a one-page sheet in the app · the exercise sheet, sized per module and stated on its own page (most run 30-60 minutes). Across the series: about 5 hours of audio and about 20 hours of total engagement. Module 1's print figures are counted from its own text and are printed in its chapter.

The gauge you build, piece by piece

The course is one long build. You assemble your own gauge across the fourteen modules, in the order the house's gauge is assembled: production first, because the composite can never exceed it; delivery second; reporting throughout.

Stage	Modules	What your gauge gains
Skeleton	M1: Your Number	Component A (production yield), counted by hand · the formula, printed · the rule against redefinition, adopted in writing
Reliability	M2-M3	The commitment line (kept ÷ committed, with reason codes for misses) · the flow fields (work in process, aging, completions)
Reporting	M4	Component E (reporting conformance) + the exception queue that defends it
Candor	M5-M6	The in-process catch line · the near-miss and alert counts as leading indicators
Delivery	M7-M9	Component D (first-pass delivery yield) formalized · delivery discipline: read-back on irreversible steps, independent verification before claim
Speed	M10-M12	Changeover cost and batch-size lines, the levers that raise A without working harder
Governance	M13	One controllable input metric named per component · the weekly review rhythm
Capstone	M14	The full gauge page assembled · the formula placed under change control · the delta read against the M1 baseline · the gauge page issued as the completion instrument

Your gauge is complete on paper by Module 9. Modules 10-14 teach you to raise it and to govern it, and the capstone assembles the final page from parts you have already been running for weeks. That sequencing is deliberate: a gauge handed over finished would be furniture; a gauge built piece by piece is an instrument the builder trusts.

The completion instrument: your number is the credential

This course issues no belts and no certificates of attendance. A printed certificate attests that a person once sat through a course; this method exists because that attestation is not enough. So the course issues exactly one completion instrument: **your gauge delta**.

At Module 1, your first component is baselined and the page dated. At the capstone, the app assembles your full gauge page (components, formula, claim bars, and the delta against your Module 1 baseline, as of that date) and that page, printed from your own instruments, is the completion instrument. It says an operation measured itself honestly across fourteen modules and its number moved by this much, under a formula that was never redefined along the way. A small delta and a large delta earn the same instrument; the page reports, it does not rank.

The subscription, the app, and this series

The series is included whole with the Team Tess Playbook subscription. There are no per-module unlocks for sale and no second paywall for the teaching: a subscriber who is paying to learn the method is never charged again for the next lesson in it.

The Playbook app holds the instruments the method needs, the uniform report form, the commitments board, and your gauge. **This series is the app's WHY layer:** the app is where the work is run; the series is where you learn why it is run that way, level by level, in the spirit of Article 7. And the app is **fully customizable:** its forms, boards, and gauge are yours to shape to your operation. The doctrine fixes the discipline (one form, one honest number, claim bars beside the rate) not the shape of your work.

Claims discipline: what may be said about this course. The course makes no claim to be the most efficient, the fastest, or the best of anything; no such league table exists. What may be said is: the method, the sources, and your own number. That is the whole advertisement, and it is enough.

Your Number

Build the first component of your own gauge, and the rules that keep it honest

LESSON PLAN: MODULE 1

The one concept	The house composite, $RATE = A \times D \times E$; build your own gauge.
Objectives	By the end of this module you can: (1) state why the composite multiplies rather than averages, say what it is and is not (it is not rolled throughput yield, and it is not a survival probability), and compute it for three stage yields; (2) define a countable unit (finished and verified) in one sentence two people would apply the same way; (3) write a counting rule, including its exclusion clause, before counting anything; (4) count one production line by hand and publish Component A to one decimal, with exclusions printed.
Source story	Construction counts its promises, Glenn Ballard's reliability research RECORD · Six Sigma defect counting RECORD · Sources are cited in the bibliography at the back of this book.
Builds in your gauge	Component A (production yield), counted by hand · the formula, printed · the rule against redefinition, adopted in writing. (Capstone arc stage: Skeleton.)
Worked example	A real gauge, anonymized, printed in full in this chapter: A 27.2% · D 94.4% · E 100% · rate 25.7% · claim bar 0 of 51, claimable figure 0.0% printed beside the rate · the day's three escapes, by class.
Field exercise	Build Component A: one production yield, counted by hand. Artifact: your eight-line Component A page, logged in the app. Printed in full in this chapter.
The opening	Artifact logged → Module 2 opens, at your own pace. Logged, not graded.
Formats & time	Counted for this module: the lesson and the worked example run about 2,910 words of print, counted from this edition's own text, about 20 minutes of study reading · the audio narration of the same lesson (Part C) is not reprinted in this book, and no word count or running time is claimed for it here · field exercise 30–60 minutes on your own work.

The Lesson

1. The promises nobody counted

Construction is an old industry, and for most of its history it ran on a sentence spoken in a trailer every Monday morning: here's what we'll get done this week. The sentence sounded like a plan. Nobody scored it. Then, in the 1990s, a researcher named Glenn Ballard started doing something almost rude in its simplicity, he wrote the promises down and counted how many were kept.

The number he found, across project after project, was this: without a system for making and scoring commitments, weekly plan reliability in construction typically ran between **30 and 55 percent**. Fewer than half the promises made on Monday were true by Friday. These were not lazy crews. They were competent people, working hard, inside a system that had never once asked whether its plans were real. The plans were weather reports nobody verified, and because nobody verified them, nobody could improve them, and week after week the same gap between promise and performance was treated as a surprise.

When Ballard's Last Planner System spread (commitments stated small, completable, and scored) reliability commonly rose to around **70 percent**, and there, honestly, it has tended to plateau. We will study that system in full in Module 2. For now, hold the founding fact of this entire series: **an industry that felt busy and professional was, by its own count, keeping fewer than half its promises, and it could not have told you that, because it had never counted**. (Evidence: RECORD: Ballard's dissertation research and two decades of replications. The figures above are the published ranges, not a vendor's brochure.)

Your operation has a version of that Monday sentence. It may be a release plan, a case queue, a content calendar, a client promise. Somewhere in it is a gap between what gets said and what gets done, a gap you currently experience as anecdotes: the rough week, the vendor who was late, the thing that almost shipped. This module replaces the anecdote with a number. Not a scorecard for someone else to grade you with. A gauge (an instrument you build, you own, and you read) whose first job is to tell you the truth about your own operation before anyone else does.

2. What a gauge is

A gauge is not a dashboard. A dashboard shows you everything and commits to nothing. A gauge answers **one question with one number**, built from components that each answer a plainer question, under rules written down before the counting starts.

The gauge you will build across this series asks three questions about your operation, in the order work actually flows:

1. **PRODUCE**. Of everything your operation is committed to produce, how much stands finished and verified? Call this **Component A: production yield**.
2. **DELIVER**. Of everything you handed over, how much landed clean the first time, on time, no error found afterward? Call this **Component D: first-pass delivery yield**. You will formalize it in Module 9.

3. **REPORT.** Do your own books foot? Do your records reconcile, and do they tell the truth about your position, including when the position is bad? Call this **Component E: reporting conformance**. You will build it in Module 4.

This week you build Component A, because A is the anchor: it is the component the whole gauge can never exceed, and it can only move when real work finishes. But the frame you build it in (the formula, the claim bars, the rule against redefinition) is the frame the finished gauge will live in. Build the frame carefully; you are not allowed to rebuild it quietly later. That, too, is part of the lesson.

3. Why the formula multiplies

Suppose your operation produces at 90 percent, delivers at 90 percent, and reports at 90 percent. What is your operation's rate? The comfortable answer is 90, the average. The gauge's answer is **72.9**: the product, and the product is the stricter master.

Be precise about what the product is, because a famous term sits nearby and this book does not claim it. The three yields are counted on three different populations: A on the committed inventory, D on the period's handovers, E on the period's records. No single cohort of units is tracked through all three stages, so the product is **not** the share of one batch of work that survives the whole chain, and it is not a probability. Lean Six Sigma's rolled throughput yield is exactly that tracked figure: the probability that a unit passes every step of one process defect-free, computed across the serial steps of a single process. RATE is this book's own instrument, the **house composite**: three yields, three plainer questions, multiplied by design, so that a weakness in any stage taxes the whole number and cannot be averaged away. Every soft dashboard you have ever seen uses the average, and the average is how a chain of three mediocre stages gets reported as excellence. The product refuses that comfort; that refusal, and not a survival claim, is what it is for.

The formula (yours, from this week forward)

$$\text{RATE} = A \times D \times E = \text{production yield} \times \text{first-pass delivery yield} \times \text{reporting conformance}.$$

Until you build D and E, your rate is a production rate, A alone, labelled as such. The label is part of the honesty.

Multiplication has a second virtue: it forbids trading. Under an average, a great month in production can paper over a delivery collapse. Under a product, it cannot, a defect in any stage taxes the whole number, in the open, and a zero anywhere is a zero everywhere. When you see the worked example in Part B, watch for the second figure the example prints beside its rate: the claimable figure under the page's own claim bar, 0.0%. The example prints that figure itself, beside the headline, on purpose. Your gauge will print its uncomfortable figures too. That is not pessimism. That is what makes the headline worth believing on the day it finally rises.

4. Component A, defined tightly enough to count

A production yield needs three things, and most home-made metrics die for lack of one of them:

- **An inventory.** A bounded, enumerable set of things your operation is committed to produce, cases, claims, documents, tickets, builds, files, units. If you cannot list it, you cannot count it, and “we’re about 80% done” is the sound of an unlisted inventory.
- **A unit, defined so two people count it the same.** Not “worked on”, finished and verified. The definition of done must include the check: a unit counts when it has passed whatever verification your operation owes it, not when effort stopped.
- **A counting rule, written before the count.** What counts, what does not, when the count is taken, and who takes it. Written before, because a rule written after the count is not a rule; it is an alibi.

Component A is then one division: **verified units ÷ countable inventory units**: the inventory minus the items excluded under the rule, with the exclusions printed, never silently dropped. Counted by hand, the first time. Not sampled, not estimated, not pulled from a tool’s cheerful default report. By hand, because the act of counting your own inventory, item by item, is where you will discover what your inventory actually contains: the duplicates, the zombies, the items that were never real. Every operation that has done this exercise has found them. Yours will too, and that discovery is worth more than the percentage.

5. Claim bars ride beside the rate

A rate tells you how you are doing. A **claim bar** tells you what you may claim. They are different instruments, and confusing them is how operations end up advertising things that are not true yet.

A claim bar is a standard stated in advance, counted in units: how many units stand at 100 percent, complete and cleared, out of all the units there are. Bars usually sit at zero for a long time, and that is the bar working, not failing: it exists to mark the distance between “well underway” and “done,” and to forbid the claim of done until it is done. Your gauge will carry its claim bars beside the rate, always, never blended into it. Progress toward a bar is measured by a component (A climbs as units finish); the bar’s count moves only when a unit crosses the whole standard. In the worked example, the bar stands at 0 of 51, next to a production yield of 27.2%, and the claimable figure the page prints beside its 25.7% rate is 0.0%: the share of the inventory the operation may yet call finished to the full standard. All three numbers are true at once. Learning to hold them at once (momentum and distance, without letting either one lie about the other) is most of what this series is for.

6. The formula is a promise

Here is the rule that separates a gauge from a mood. **Your formula is printed, dated, and fixed. It may be changed only in the open: the new formula, the reason, and the date written on the gauge’s face, with the old series kept and the break marked.** A change may never be applied backward to raise a number already reported. If a future formula would

read higher on the same facts, you report both numbers for a full cycle and let the lower one stand beside the higher until the new series has earned its own history.

Why so strict? Because the first time any measured operation feels pain, the temptation arrives, not to do better work, but to redefine the denominator. Exclude the hard cases. Count the drafts as done. Average instead of multiply. Each redefinition is small, reasonable, and fatal: the moment the formula bends, the gauge stops measuring the operation and starts measuring the operator's comfort, and it will never tell the truth again. Six Sigma's real inheritance is not the Greek letter; it is this refusal. Your number can only ever go up one way (by the work improving) and that constraint is precisely what will make the number valuable to you, to a client, or to anyone you ever show it to.

7. Reading your first number

Your baseline will probably be lower than you expect. Construction's professionals found 30 to 55 percent. The worked example in Part B (a real operation's real gauge, anonymized) reads **25.7%** on the day it was first assembled, a day that operation's own register counts among its most productive ever: fifty-four deliverables closed, its books reconciled to the cent. Both things are true. The number was low because the standard was absolute and the inventory was honestly counted, not because the operation was idle.

So read your first number the way an instrument-maker reads a first calibration: not as a verdict, as a zero point. Everything this series does from here (the commitments, the work-in-process limits, the error-proofing, the candor practices) exists to move this number by moving the work, and your gauge, honestly kept, is how you will know whether any of it actually worked. A baseline you flinched from is a baseline you cannot improve on paper or in fact. Write it down exactly as it counts. Date it. That page is the first honest document your operation has produced about itself, and it is the instrument this whole series will eventually be judged by, yours, in Module 14, when we read the delta.

8. Where the series goes from here

Next, Module 2 puts your commitments under the same discipline: one week of promises stated so that done is binary, scored as kept ÷ committed, Ballard's instrument, on your work. Module 3 caps the work you start, because your flow time is set by your inventory of open efforts, not by your effort. Module 4 builds Component E in your books. By Module 9 your gauge has all three components and reads whole: $A \times D \times E$, all three components in. The rest of the series raises it, and Module 14 places the formula under change control and reads your delta. One build per module, one artifact per module, one number the whole way. Open the exercise sheet (Part D) now, while the definitions are fresh, your inventory is waiting to be counted.

PART B

The Worked Example: a Real Gauge, Anonymized

The house's own operation, shown at the WHAT level: components, formula, claim bar, and the day's defects by class. Names and unpublished detail are withheld; every figure is real, from the gauge's inaugural census of

October 24, 2026. The operation is a documentation-and-reconciliation program: a fixed inventory of items across multiple jurisdictions, each item counted only when reconciled and verified against its source of record.

25.7%

THE RATE (THE HOUSE
COMPOSITE)

= 27.2% production × 94.4% first-
pass × 100% reporting

0 of 51

CLAIM BAR · UNITS AT
100%, COMPLETE &
CLEARED

Caps what may be
claimed or published. At
zero, the bar is the rule
working.

0.0%

THE CLAIMABLE FIGURE

The share of the inventory
finished to the full
standard, printed beside
the rate because the two
answer different
questions: the rate says
how the machine runs;
this says what may be
claimed.

Comp.	What it asks	Count	Value
A	Production yield: of the committed inventory, how much stands reconciled and verified	217 of 799 items	27.2%
D	First-pass delivery: of the day's closed deliverables, how many landed clean the first time	51 of 54 clean · 3 escapes (below)	94.4%
E	Reporting conformance: the books foot and reconcile to the cent	Chains footed; zero unexplained entries	100%
E'	Position: reported separately, never blended into E	Operating buffer, current period	-\$40.16

Read E and E' together, the way the example prints them: the reporting is perfect and the position is tight, in the same week, on the same page. A true book in a tight week conforms; a padded book in a flush week would not.

The day's three escapes: printed unsoftened, by class

1 · Finished, never handed over. A deliverable was completed and sat in its folder for hours while the desk waited on it, never delivered, never reported. Under the counting rule, a file that was never handed over is unlanded: built is not delivered.

2 · Prepared, not surfaced at its hour. A report owed at a named hour was finished on time (text and audio verified) and then never presented; the recipient had to ask for it. A deliverable the recipient must request is a late land.

3 · A wrong fact, caught in re-verification. A document delivered as final carried a factual error on its face, found when the desk re-verified the delivered document afterward. It had crossed the delivery point and the recipient held it: an escape under the doctrine's own definition (found after handover, it counts against D), however it came to light, and the page counts it as one. The corrected edition is owed and scheduled. Caught in-process work is counted separately (the same day logged **3 in-process catches, 0 escapes among them**) because the catching machinery deserves its own line, and because hiding catches would teach the desk to stop catching.

How to read this gauge: the headline (25.7%) is the product of the three yields, each counted on its own population; the rate is computed from the exact counts and reported to one decimal ($217 \div 799 \times 51 \div 54 = 25.65\%$, printed 25.7%). Component A is the road, it moves only when items reconcile, roughly a tenth of a composite point per item, so the climb is honest arithmetic, not a mood swing. The claim bar (0 of 51) says nothing is finished to the full standard yet, and the claimable figure stands at 0.0% beside the rate: momentum and distance, both true, neither lying about the other. When this operation's rate rises, it will rise the slow way: item by item, week by week, under a formula that has not moved since the day it was printed. That is the instrument you are building this week.

About the audio for this module. The taught edition's audio is the narration of this same lesson (one voice, signposted, with the arithmetic spoken slowly and the worked example's numbers walked through aloud). It teaches the same lesson by ear and is not reprinted here; this printed lesson is the document of record.

Check yourself

Q. Your operation produces at 80%, delivers clean at 90%, and reports true at 95%. What does the gauge read, and what is the number not?

A. $0.80 \times 0.90 \times 0.95 = 68.4\%$. It is not the share of any batch that survived all three stages, and it is not a probability: the three yields are counted on three different populations, and the product is the house composite, not rolled throughput yield.

Q. A unit is finished but has not passed the check your unit definition requires. Does it count in Component A?

A. No. A unit counts when it is finished and verified; “done but unchecked” is not done, and counting it is the first small lie a gauge dies of.

Q. Your claim bar stands at 0 of 51 while your rate climbs. Is the page contradicting itself?

A. No. The rate says how the machine runs; the bar says what may be claimed as finished to the full standard. Momentum and distance are both true at once, and the page prints both.

The Field Exercise: Count One Thing True

Your artifact: one Component A page, logged in the app. **Time:** 30–60 minutes, on your own work. Module 2 opens once this page is logged, at your own pace, logged, not graded. Nobody marks this. The count is the teacher.

The page you are building

Your Component A page carries exactly eight lines. Nothing else goes on it:

1. The operation and the one production line you are measuring (one line only this week).
2. The **unit**: your definition of one finished, verified item, in one sentence a stranger could apply.
3. The **counting rule**: what counts as inventory, what is excluded and why, when the count is taken, who counts.
4. **Inventory**: the full count of units the line is committed to produce.
5. **Excluded under the rule**: items removed by the rule's own terms, named and counted, never silently dropped.
6. **Verified**: units finished and passed the check your definition requires.
7. **Component A**: $\text{verified} \div \text{countable inventory}$ (inventory minus the items excluded under the rule), as a percentage, one decimal.
8. The **date** and your name as counter. A count without a counter is a rumour.

The six steps

1. **Pick one production line (≈ 5 min)**. The line where an outsider could ask “how many, and how do you know?” and you would currently have to wave your hands. Cases, claims, documents, tickets, builds, files, if your work feels uncountable, count deliverables or commitments; every operation finishes something. One line only: a broad count teaches nothing a narrow count doesn't teach faster.
2. **Define the unit (≈ 10 min)**. Write the sentence: “One unit = _____, finished and verified by _____.” If two honest people applying your sentence to the same item could disagree, the sentence isn't done. Rewrite until they can't.
3. **Write the counting rule (≈ 5 min)**. Before you count a single item. Include the exclusion clause now (duplicates, items that belong to another line, items opened in error) so that when you find them (you will), the rule, not your mood, decides what happens to them.
4. **Count the inventory by hand ($\approx 15\text{--}30$ min)**. The whole denominator. No sampling, no tool reports, no “about.” List them or tally them, but touch every one.
5. **Count the verified (≈ 10 min)**. Apply the unit definition item by item. “Mostly done” does not count. “Done but unchecked” does not count. The stringency is the point.

6. **Divide, date, sign, log (≈5 min).** One division. Write the percentage exactly as it comes out: 41.0% is a fine first number; a rounded-up 45% is a small lie that will compound. Enter the page in the app; your baseline is now of record.

Worksheet

#	Item (or tally block)	In inventory? (rule)	Verified? (definition)	Notes / exclusion reason
Totals		Inventory: _____	Verified: _____	Excluded under rule: _____ A = _____ %

A small worked example (so you can see the size of the thing)

A **constructed teaching example**, the team Modules 2 to 14 will follow: a four-person billing team takes its application queue. Its figures are the example's own, computed under the rules as printed, and they check. Unit sentence: "One unit = one application processed end-to-end and passed by the second-person check." Counting rule: everything in the quarter's queue log counts; duplicates and another team's misroutes are excluded under the rule and printed. Two afternoons of counting by hand: **inventory 1,240 · excluded under the rule 111 (61 duplicates, 38 misroutes, 12 opened in error) · verified 986**. Component A, on the countable inventory (the full inventory of 1,240 minus the 111 excluded under the rule, which is 1,129: **$986 \div 1,129 = 87.3\%$**) with the exclusions printed on the same page, so any reader can recompute it either way. The team lead's comment, recorded under the count: "We thought we were behind on processing. We're actually behind on queue hygiene. Different problem." That sentence is what this exercise is for.

Pitfalls, each one has ended a real gauge:

- **Denominator shopping:** choosing the inventory that makes the number kindest. The rule is written first precisely so the inventory can't be chosen after its contents are known.
- **Counting drafts:** “done except for the check” is not done. The definition includes verification or it includes nothing.
- **Mid-count redefinition:** if the definition proves broken, stop, rewrite the page, recount from zero under the new rule, and keep the broken page. A recounted baseline is honest; a patched one is fiction.
- **Sampling:** an estimate of your inventory is an opinion about your inventory. First count: everything, by hand. Later counts may use your systems, once your systems have survived comparison with your hands.
- **Rounding toward comfort:** report one decimal, as computed. The gauge's credibility is built from a hundred small refusals like this one.

Done looks like this: eight lines on a page, a real denominator you touched item by item, a percentage you did not round toward comfort, your name and the date at the bottom. Log it. Next, Module 2 (Commitments You Can Count) puts your weekly promises under the discipline Ballard brought to the trailer: stated so done is binary, scored as kept ÷ committed, with a reason code on every miss. Your gauge is about to get its second line, and your operation its first reliability number.

What this module teaches, shows, and never shows, stated once for Module 1 and in force for every module:

Teaches: the house composite gauge and its construction, with rolled throughput yield defined and distinguished in section 3, and Ballard's published reliability research. **Shows:** the house's gauge at the WHAT level (components, formula, claim bar, real figures, defects by class) anonymized. **Never:** the house's unpublished detail or any principal's private information. Teaching the method is the product.

Commitments You Can Count

Promises stated so that “done” is binary, and scored

This module's artifact: one week of binary commitments, with your first PPC scored and a reason code on every miss: your commitment page for the week, logged in the app.

Time: lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** the commitment line: kept ÷ committed, with reason codes for misses. (Capstone arc stage: Reliability.)

LESSON PLAN: MODULE 2

The one concept	PPC, percent plan complete: promises stated so done is binary, scored as kept ÷ committed.
Objectives	By the end of this module you can: (1) restate a week's work as binary commitments: each either done or not done, with no third state; (2) score the week's PPC as kept ÷ committed, to one decimal; (3) attach a reason code to every miss and name the week's most frequent code; (4) state two ways PPC gets gamed and the check that catches each.
Source story	The Last Planner System, Glenn Ballard RECORD · the ~70% plateau in plan reliability, and how PPC gets gamed RECORD .
Builds in your gauge	The commitment line: kept ÷ committed, with reason codes for misses. (Capstone arc stage: Reliability.)
Worked example	One week of commitments from a small billing team, the team from Module 1's exercise, with the week's PPC counted and every miss reason-coded.
Field exercise	Write one week of binary commitments before the week begins; score the first PPC. Specification in full in Part D.
The opening	Artifact logged → Module 3 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The promise, scored

In Module 1 you met Glenn Ballard's rude, simple act: he wrote the Monday promises down and counted how many were true by Friday. Without a system, construction's weekly plan reliability ran between 30 and 55 percent. This module is the system he built from that count, the Last Planner System, reduced to the one piece you can run this week: a week of commitments stated so that done is a yes-or-no fact, scored as a single percentage called **PPC, percent plan complete**. One honesty note on the transfer, printed here so the claim travels at its true strength: PPC was born counting weekly work-plan assignments in construction. Carrying it to any operation's promises, as this module does, is a reasoned transfer from the source, graded INFERENCE.

PPC asks one plain question: of everything you committed to finish this week, how much did you finish? Not how hard you worked. Not how close you got. Finished, or not. The score is kept ÷ committed, and it is the second line your gauge gains: the commitment line, standing beside Component A from Module 1.

2. Why the promise has to be binary

Most weekly plans fail the count before the week even starts, because they are not written as promises at all. They are written as intentions: work on the Johnson file, move the migration forward, look at the backlog. None of those can be scored. Is “move the migration forward” done on Friday? Nobody knows, including the person who wrote it. An unscorable promise is worse than no promise, because it lets everyone feel planned while nothing is measured.

A binary commitment has four marks. It names **one thing**, not a bundle. It is **small enough to finish inside the week**. Its done-state is **checkable by a stranger**: a reader holding the sentence on Friday can score it without asking you a single question. And it is made **before the week begins**, because a promise written after the work is a report wearing a costume.

The stranger test. Hand your commitment list to someone who knows nothing about your work. If they could score every line on Friday from the sentences alone, the list is binary. Every line they would have to ask you about is a line that is not yet a commitment. Rewrite it until they wouldn't.

3. How a commitment week runs

The rhythm is short enough to hold in your head, because it has to survive busy weeks, not just calm ones:

1. **Commit (before the week).** Write the week's commitments. Fewer than feels ambitious: a list you might actually keep beats a list that flatters you on Monday and indicts you on Friday.

2. **Work the list as written.** The list does not get edited mid-week to make it easier. Life will edit it for you. That is what reason codes are for.
3. **Score (at week's end).** Each commitment: kept, or not kept. No partial credit. A commitment 90 percent done is a miss, and the 90 percent is real work that will finish next week as a new commitment, not as a rounding of this one.
4. **Code every miss.** Beside each miss, a short reason code from your own small set: the work was waiting on someone else, it was bigger than stated, you were pulled onto other work, the material never arrived. The code is the lesson; the miss is just its envelope. (One divergence from the source practice, printed as one: Last Planner practice keeps reasons for non-completion in shared categories so codes aggregate across teams and projects. This book has each learner keep a small set of their own, which serves one operation well and aggregates poorly. The trade is named so you can reverse it deliberately when your operation outgrows one page.)
5. **Count the codes.** The week's most frequent code is the week's real finding. One miss is an event. Four misses with the same code is a system telling you its name.

4. The score, computed honestly

PPC = commitments kept ÷ commitments made, as a percentage, one decimal.

Counted over one week, on the list as written before the week began. A changed promise is a miss with a reason code, not a rewrite.

Two rules keep the division honest. First, the denominator is the list as written, never the list as it ended up after mid-week edits. Second, a commitment carried into next week counts as a miss this week and as a new commitment next week. That feels like double punishment. It is actually double information: the miss tells you this week's estimate was wrong; the new commitment gives the work a fresh, honest promise instead of an ageing one nobody believes.

5. The plateau: and what it is telling you

Ballard's published finding, carried in Module 1, is that reliability under this system commonly rises to around **70 percent** and there tends to plateau. Read that plateau correctly. It is not the ceiling of human effort; crews at 70 percent are not slacking the other 30. The plateau is where the easy causes are gone and the remaining misses share structural reasons: work released before it was ready, promises made for other people's capacity, inputs that arrive late as a matter of routine. Your reason codes are how you find out which structure is yours. A team that only watches its PPC sees a number stuck at 70. A team that counts its codes sees the three fixable reasons the number is stuck, and the series spends its next modules on exactly those fixes: limiting what you start (Module 3) and making the recurring error impossible (Module 4).

6. How PPC gets gamed: and the checks

Any scored promise invites gaming, and this system has been running long enough that the games are known by name. Learn them now, while your list is young:

- **Sandbagging:** committing only to what is already nearly done, so the score gleams. The check: commitments must be written before the week, and a PPC near 100 percent week after week is itself a finding. Real work, promised honestly, misses sometimes. A perfect score is a list that was never tested.
- **The quiet rewrite:** editing a commitment mid-week so Friday's version matches what happened. The check: the list as written is the list that is scored. Changes are allowed (work changes) but a changed promise is scored a miss with the code changed, so the record shows how often the plan itself moved.
- **The vague promise:** writing commitments nobody could score, so nothing is ever plainly missed. The check: the stranger test, applied before the week starts, not after it ends.

Notice what all three checks have in common: they protect the record, not the score. This series will tell you the same thing in every module: a gauge that can be flattered will be, and then it is furniture.

7. Reading your first PPC

Your first score will probably land somewhere between 50 and 80 percent. Score it exactly as it counts, code every miss, and write the week's most frequent code under the number. That code (not the percentage) is what you carry into Module 3, where you will count how much work you have open at once, and discover why so many of those promises were crowded out before the week began.

PART B

The Worked Example

The worked example in this module continues the teaching example from Module 1: the four-person billing team and its application queue, the team whose hand count produced Component A of 87.3% (986 verified out of a countable inventory of 1,129). The figures below are that team's first commitment week, counted under the rules in Part A.

The week as committed

Before the week began, the team wrote **18 commitments**. Each was small and binary. Three, as written:

- "All 214 priority applications received by Wednesday noon are processed and second-checked by Friday close."
- "The duplicate-check step is run on every new batch before processing starts, all week."
- "The misroute tray is emptied and handed to the receiving team by Thursday."

The week as scored

Line	Count	Figure
Commitments made (list as written, before the week)	18	
Kept	13	
Missed	5	
PPC = 13 ÷ 18		72.2%

The five misses, with their reason codes: **waiting-on-another-team, 2** (the receiving team could not take the misroute tray on Thursday, and one batch waited a full day on a supervisor's sign-off) · **stated-too-big, 2** (two commitments bundled a week's processing and a clean-up task; both finished half and scored as misses) · **pulled-away, 1** (one processor was borrowed for a day by another queue).

How to read this week. 72.2% sits right at the published plateau, and the codes say the plateau here is not effort. Four of five misses are about how work arrives and how it is stated, not how fast anyone processed. The team's most frequent code, waiting-on-another-team, names the fix to try first: agree the hand-over times before committing to them. That is a Module 3 and Module 11 conversation, and the team only knows to have it because the codes were counted.

One more honesty note from the same page: the team lead's first instinct was to score the two stated-too-big commitments as “mostly kept.” Under the rule, they are misses, coded. The page prints them as misses. Next week's list was written smaller, which is the system working, not the team failing.

PART D

The Field Exercise: One Week of Promises, Scored

Your artifact: one week of binary commitments, with the first PPC scored, logged in the app as your commitment page for the week. **Time:** 30–60 minutes across the week, on your own work. Module 3 opens once this page is logged, at your own pace, logged, not graded. Nobody marks this. The count is the teacher.

The page you are building

Your commitment page carries exactly six things:

1. The **week**: its start and end dates.
2. The **commitments as written before the week began**: each one binary, each one a stranger could score.

3. Beside each, at week's end: **kept** or **not kept**. No third state.
4. Beside every miss, a **reason code** from your own short code set, written on the page.
5. **PPC**: kept ÷ committed, as a percentage, one decimal.
6. The **week's most frequent code**, named in one line, and your name and the date as scorer.

The five steps

1. **Write the week (≈20 min, before it begins)**. List what you will finish this week, not start, not advance: finish. Apply the stranger test to every line and rewrite any line a stranger could not score.
2. **Fix the list**. Date it. From here the list is the record; it is not edited again this week, whatever happens.
3. **Work the week as it comes (≈5 min a day)**. When a commitment breaks, note the reason the day it breaks, while you still remember the true one.
4. **Score (≈15 min, at week's end)**. Kept or not kept, line by line. Attach the reason code to every miss. Count the codes.
5. **Divide, date, sign, log (≈5 min)**. PPC to one decimal, exactly as it counts. Write the most frequent code under the score. Log the page.

Worksheet

#	Commitment (as written before the week)	Kept? (Y/N)	Reason code if missed
1			
2			
3			
4			
5			
6			
7			
8			
9			
Totals		Kept: _____ of _____	PPC = _____ % · Most frequent code: _____

Pitfalls: each one has ended a real PPC:

- **Partial credit:** “90 percent done” is a miss with good news attached. Score the miss; carry the work forward as a new commitment.
- **The mid-week rewrite:** editing the promise to fit the week. A changed promise is a miss, coded changed.
- **Committing other people's capacity:** promising a finish that depends on a hand-over you never agreed. The miss will code itself waiting; agree the hand-over first, or do not commit.
- **Codes so kind they say nothing:** “busy week” is not a reason code. A code names a cause another person could act on.

Done looks like this: a dated list a stranger could score, one PPC figure computed on the list as written, a reason code on every miss, and the week's most frequent code named. Log it. Next, Module 3, Less Started, More Finished, counts everything you currently have open, and shows you why the crowd of open work, not your effort, sets how long anything takes.

Check yourself

Q. A commitment reads “move the migration forward.” Does it pass the stranger test?

A. No. A stranger could not score it on Friday; it names no finish state. Rewrite it until done is a yes-or-no fact.

Q. You kept 13 of 18 commitments. What is PPC, and what else must the page carry?

A. 72.2%, and a reason code on every one of the 5 misses, with the week's most frequent code named: the code, not the percentage, is the finding you carry forward.

Q. Mid-week, a commitment's terms change. How is it scored?

A. As a miss, coded changed. The list as written is the list that is scored; the change is information, not an edit.

THE OPENING: MODULE 2

Artifact logged → Module 3 opens at your own pace. The opening is the logged commitment page: the list as written, the PPC as counted, a reason code on every miss. Logged, not graded. Nobody marks this; the count is the teacher. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: percent plan complete and commitment scoring (Ballard's published Last Planner method), with the published plateau and the known ways the score gets gamed. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 2 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Less Started, More Finished

Limit the work in process, and let flow time fall

This module's artifact: your work-in-process census, with your first cap set in writing, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** the flow fields: work in process, aging, and completions. (Capstone arc stage: Reliability.)

LESSON PLAN: MODULE 3

The one concept	Work-in-process limits; Little's Law: flow time is set by how much is open, not by how hard people work.
Objectives	By the end of this module you can: (1) census every open effort in your operation and state the count; (2) state Little's Law in your own words and apply it to your own open work; (3) set a first work-in-process cap in writing, with the date it takes effect; (4) name what happens to new work when the cap is full, before the situation arises.
Source story	Queueing theory's identity, Little's Law RECORD : the identity as published (Little, 1961), with the conditions it needs stated in the lesson · Toyota's kanban system RECORD .
Builds in your gauge	The flow fields: work in process, aging, and completions. (Capstone arc stage: Reliability.)
Worked example	The billing team's open-work census: the count as found, the flow time the arithmetic predicts, and the cap as set.
Field exercise	Census your own work in process; set your first cap in writing. Specification in full in Part D.
The opening	Artifact logged → Module 4 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The busiest trap in work

Every operation knows the feeling: everything is started, nothing is finished, and everyone is busy all the time. Module 2's reason codes usually point here first. A promise misses not because nobody worked on it, but because it was one of thirty open efforts, each getting a slice of attention too thin to finish any of them. This module teaches the one law that governs that trap, and it carries a scientist's name because it is a theorem, with conditions, not advice.

2. Little's Law, in plain words

In 1961 the researcher John Little proved a relationship so simple it is easy to miss how much it controls. For any steady flow of work, three numbers are locked together: how many items are **open** (in process), how fast items **finish** (completions per week), and how long an average item **takes** from start to finish (flow time).

Flow time = work in process ÷ completions per week

The identity holds for long-run averages in a stable system: items conserved (what enters eventually leaves, by finishing, parking, or killing), and averages taken over a horizon long enough to mean something. Inside those conditions there is no third possibility. Outside them, the division estimates, and section 7 says how to read the estimate.

Read the formula the uncomfortable way. If your operation finishes 6 items a week and has 24 open, the average item takes 4 weeks from start to finish, no matter how busy everyone feels, no matter how many hours go in. Effort does not appear in the equation. The count of open things does. Read the result as the law's statement about the average item, not as a stopwatch on any single one. An operation that wants faster finishes has exactly two honest levers: finish more per week (hard, slow to change) or **keep fewer things open** (a decision, available today).

3. Kanban: the law with a fence around it

Toyota's kanban system is Little's Law given a physical form. A kanban is a card: a fixed number of cards circulate, and work may only start when a card is free. The number of cards is the cap on work in process. No card, no start. The system feels strict because it is: it refuses the comforting move of starting one more thing to feel responsive, because every extra start lengthens the wait of everything already open.

Translated out of the factory, the card can be anything with a fixed count: a board column with a stated limit, a tray that physically holds only so many files, a rule that this team runs at most five live efforts and the sixth waits, visibly, in a named queue. The form does not matter. The fixed count does.

4. What a cap is: and what it is not

- **A cap is a number, written down, with a date it takes effect.** “We should start less” is a sentiment. “From Monday, at most 12 open efforts; the 13th waits in the intake list” is a cap.
- **A cap needs its full-rule stated in advance.** When the cap is full and urgent work arrives, what happens? One item is finished first? One is formally parked, by name, by a named person? Decide it in the calm, in writing, because in the moment, the answer will otherwise always be “just this once,” and a cap with exceptions on demand is a decoration.
- **A cap is not a workload cut.** Nothing is refused forever; work waits in a visible queue instead of an invisible one. Waiting in the open is honest: the requester can see their place. Waiting inside a pile of started work is the same wait, hidden, and it slows everything else down while it hides.

5. The census comes first

You cannot cap what you have not counted, and most operations have never counted their open work. The census is deliberately plain: list every open effort, write when it started and when it was last touched. Three numbers fall out: the **open count**, the **oldest age**, and **last week's completions**, and they become your gauge's flow fields, counted the same way every week from now on.

Expect the census to find what Module 1's hand count found in the inventory: zombies. Efforts started months ago, untouched for weeks, still counted as “in progress” by everyone who reports on them. A zombie is not progress; it is open work inflating your flow time while doing nothing. The census does not judge them. It prints them, with their ages, and lets the print do the judging.

6. Aging: the field that tells the truth early

Of the three flow fields, watch **aging** most closely. The open count can look stable while individual items quietly rot: a list of 20 open efforts where 6 have not moved in a month is sicker than a list of 24 where everything moved this week. Age is measured two ways on your page: the age of the oldest open item, and the date each item was last touched. An item untouched across two weekly reviews is named at the review: finished, formally parked, or honestly killed. What it may not do is drift, unnamed, forever.

7. Reading your first census

Do the division on your own page before you set the cap: $\text{open count} \div \text{last week's completions}$ is your average flow time, in weeks, whether you ever wrote it down before or not. One reading discipline before you divide: last week's completions is a short window, so treat your first flow-time figure as an estimate with wide error bars; it steadies as the weekly counts accumulate, which is one more reason the census repeats weekly. If the number surprises you (and it usually does, on the high side) that surprise is the module working. Then set the cap low enough to change the arithmetic and high enough to actually hold. A cap nobody keeps teaches the operation that caps are theatre. A cap kept for a month teaches it that finishing is the job.

The Worked Example

Continuing the teaching example: the billing team from Modules 1–2 (Component A 87.3% · first PPC 72.2%, with waiting and pulled-away among its miss codes) counts everything it currently has open.

The census as found

Flow field	Count
Open efforts (every started piece of work, listed item by item)	24
Oldest open item	11 weeks since it was started; untouched for 6 weeks
Completions in the last full week	6

Little's Law on those figures: **flow time = $24 \div 6 = 4.0$ weeks**. The average piece of work in this team takes four weeks from start to finish, in a team that feels, and is, busy every day. The census also named what the count alone could not: **5 of the 24** open items had not been touched in three weeks or more. They were not being worked. They were being carried, and every one of them was lengthening the wait of the 19 that were real.

The cap as set

The team's cap, as written: "From next Monday, at most **12 efforts open at once**. New work waits on the intake list, in the open, in arrival order. When the cap is full and urgent work arrives, the team lead may park one open item by name to make room. The parking is written beside the census, never silent." At the team's current completion rate of 6 per week, the cap predicts a flow time of **$12 \div 6 = 2.0$ weeks**, the same people, the same pace of finishing, half the wait, bought entirely by starting less. The prediction carries the law's own warning: it holds only if completions hold at 6 a week under the cap. The law relates the three numbers; it does not promise the cap leaves throughput untouched, and a cap that starves a step will show up as falling completions inside a month. The team watches both numbers and believes the pair, not the prediction.

Notice the honest remainder on the same page: the cap does not say what happens to the 5 zombies. The team ruled separately: two were finished that week, two were parked by name, one was killed as no longer needed, and the census for the following week starts from 19 real items on its way to 12. The cap is the fence; the rulings are the housekeeping the fence made unavoidable.

A second operation, so the law is not mistaken for a billing trick: a solo electrician's job board (a constructed example, built for this page). Open jobs: **9**. Completions last week: **3**. The division: 9

$\div 3$ = an estimated **3 weeks** from call to finished job, which is what his customers already tell him. He sets a cap of **6** open jobs, new calls waitlisted in the open where callers can see their place: the arithmetic predicts $6 \div 3 = \mathbf{2 \text{ weeks}}$, if completions hold at 3 a week, which the law does not promise and his first month under the cap will test.

PART D

The Field Exercise: Count What Is Open: Then Cap It

Your artifact: your work-in-process census, with your first cap set in writing, logged in the app.

Time: 50–70 minutes, on your own work. Module 4 opens once this page is logged, at your own pace, logged, not graded.

The page you are building

1. The **census**: every open effort, listed, with the date it started and the date it was last touched.
2. The **open count**: the census's total, as the first line of the artifact.
3. The **flow fields**: open count $\cdot$ oldest age $\cdot$ completions in the last full week $\cdot$ and your computed flow time (open $\div$ completions).
4. Your **cap**: the maximum number of efforts open at once, in writing, with the date it takes effect.
5. The **full-rule**: what happens to new work when the cap is full, stated before the situation arises.
6. Your name and the date as census-taker.

The four steps

1. **Census ($\approx 20\text{--}30$ min)**. List every open effort. All of them, including the ones nobody has touched in weeks and the ones you would rather not show anyone. An effort is open if it has been started and not finished, parked by name, or killed.
2. **Write the flow fields (≈ 10 min)**. Open count, oldest age, last week's completions.
Divide: open $\div$ completions = your average flow time in weeks. Write it down exactly as it comes out.
3. **Set the cap (≈ 10 min)**. Choose a number that changes the arithmetic and that you can actually hold. Write it with its start date.
4. **Write the full-rule, date, sign, log (≈ 10 min)**. State what new work does when the cap is full. Then log the page.

Worksheet

#	Open effort	Started	Last touched
1			
2			
3			
4			
5			
6			
7			
8			
9			
Open count: ____ · Oldest age: ____ · Completions last week: ____ · Flow time = ____ weeks			
Cap: at most ____ open from (date) ____ · Full-rule: _____			

Pitfalls:

- **The flattering census:** leaving the zombies off the list because “they don't really count.” They count in Little's Law whether you list them or not.
- **The heroic cap:** a cap so low it breaks in the first week and teaches everyone that caps are theatre. Holdable beats heroic.
- **Starting to beat the cap:** rushing a start in on Friday because the cap lands Monday. The census date is the honest line; the count stands as found.
- **A queue nobody can see:** work waiting for a slot must wait in the open, in a named list, or the cap has just moved the hiding place.

Done looks like this: a complete census you can defend item by item, three flow fields and the division that follows from them, and a cap with a date and a full-rule on it. Log it. Next, Module 4, Make the Error Impossible, takes your most recurring error and builds the check that stops it from ever shipping again, and your gauge gains Component E.

Check yourself

Q. Your census finds 24 open efforts and 6 completions last week. What is the average flow time?

A. 4.0 weeks: flow time = work in process ÷ completions per week, whatever anyone's effort felt like.

Q. What two conditions does Little's Law need before the division is a law and not an estimate?

A. Long-run averages in a stable system, with items conserved: what enters eventually leaves. Short windows, and parked or zombie items inflating the open count, make the figure an estimate, read as one.

Q. The cap is full and urgent work arrives. What happens?

A. Whatever your written full-rule says, decided in the calm in advance: the work waits in the open queue, or a named item is parked by name. "Just this once" is not a full-rule.

THE OPENING: MODULE 3

Artifact logged → Module 4 opens at your own pace. The opening is the logged census page: the full list, the flow fields, and a cap with a date and a full-rule. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: Little's Law (published queueing theory, with its conditions stated) and work-in-process limits as practised in Toyota's kanban system. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 3 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Make the Error Impossible

Design the check into the work, and keep the books continuously true

This module's artifact: one validation, built and working, that makes one recurring error impossible, with the error, the validation, and the proof logged in the app; in the same week, your Component E rule written and your exception queue opened. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** Component E (reporting conformance) + the exception queue that defends it. (Capstone arc stage: Reporting.)

The one concept	Poka-yoke: error-proofing; and continuous close: records reconciled as work happens, not reconstructed afterward.
Objectives	By the end of this module you can: (1) name one recurring error in your operation and state how often it has occurred in a stated window; (2) build one validation that makes that error impossible (not discouraged, impossible) and show it working; (3) state your operation's reporting-conformance rule: what "the books foot" means, line by line; (4) open an exception queue that holds every item your conformance rule cannot yet clear.
Source story	Payer first-pass yield, the claims discipline of counting work that passes clean the first time RECORD as a discipline; any exact band for it INFERENCE : the book teaches the counting discipline and prints no borrowed band · The practice of reconciling continuously, carried in this book under the name continuous close; healthcare's medication reconciliation is a cross-industry patient-safety practice RECORD . ThedaCare's own documented contribution, Collaborative Care, is carried in Module 11.
Builds in your gauge	Component E (reporting conformance) + the exception queue that defends it. (Capstone arc stage: Reporting.)
Worked example	The billing team's recurring error, its validation, and its first Component E count, books, exception queue and all.
Field exercise	Find one recurring error; build one validation that makes it impossible; write your Component E rule and open the exception queue. Specification in full in Part D.
The opening	Artifact logged → Module 5 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. Two kinds of fix

When an operation finds an error, it reaches for one of two fixes. The first is a warning: a reminder, a sterner instruction, a poster, a note in the file saying be careful here. The second is a design: the work is changed so the error cannot proceed: the form will not accept it, the step will not complete with it present, the wrong thing physically does not fit. Toyota's name for the second kind is **poka-yoke**, error-proofing, and this module's whole argument fits in one sentence: **warnings depend on attention, and attention is the one resource your**

busiest week does not have. The errors that recur in your operation are not recurring because people are careless. They are recurring because the work, as designed, lets them through.

2. What counts as error-proofing

A real validation has three marks:

- **It sits at the point of the error.** The check runs where the mistake is made, while it can still be stopped, not in a review three days later, when stopping means repair.
- **It blocks, it does not advise.** While the error is present, the work cannot proceed. A message you can click past is a warning wearing a validation's clothes.
- **It can be demonstrated.** You can run the old error at it, on purpose, and watch it stop. If you cannot show it stopping, you have a hope, not a check.

The forms are humble, and that is the point. A field that will not accept a number with the wrong number of digits. A checklist step the system will not mark complete until the count matches. A template with the totals already built as formulas, so a hand-typed total that disagrees cannot be saved. A second-person check, Module 1's billing team built one into its very definition of a unit, where the passer is never the processor. None of these requires new software. All of them require deciding that this error, specifically, is finished.

3. Count the error before you kill it

Error-proofing starts with a count, in a stated window: this error, this many times, in these weeks. The count does two jobs. It proves the error is recurring, worth a design fix rather than a one-off correction. And it gives you the before-figure, so that when the error stops occurring you can say by how much, on the record, instead of by feel. An error you never counted can never be shown to have died.

4. The second half of the module: books that stay true

The same discipline (reconcile at the point of work, not afterward) builds the third component of your gauge. **Component E, reporting conformance**, asks: do your records foot? Do they reconcile, line by line, against what actually happened, and do they tell the truth about your position, including when the position is bad?

The practice behind it is the **continuous close**. Instead of reconstructing the books at the end of a period from memory and piles, the records are reconciled as the work happens, each entry matched to its source while both are fresh. Healthcare learned this the hard way with medication lists. Medication reconciliation, the discipline of keeping a patient's medication list continuously matched to the orders at every hand-off, is a cross-industry patient-safety practice, carried in accreditation standards for years, and it exists because a list rebuilt from memory at discharge carried errors that a list maintained continuously did not. This book borrows its shape for your books: reconcile as the work happens. (ThedaCare's own documented contribution to this family of practice, its Collaborative Care redesign, is carried in Module 11, where it belongs.) Your books are the same list, in a different uniform.

5. Your Component E rule: and the exception queue

Component E needs a rule stated as plainly as Component A's counting rule. It has three parts:

1. **What “footing” means, line by line.** Which records must agree with which sources, to what tolerance: to the cent, to the item, to the date. Write it so a stranger could run the check.
2. **When the check runs.** Continuously, or at a stated short interval. “At year-end” is reconstruction, not conformance.
3. **What happens to what will not clear.** Every item the rule cannot reconcile goes into an **exception queue**: a named, open list that holds the item, its age, and who owns clearing it. The queue is what makes a high E honest: the conforming share is high because the rest is standing in the open, counted, not because it was swept somewhere uncounted.

Component E = records reconciled clean ÷ all records in the window, under your stated rule.

The items that do not reconcile are not deleted from the denominator: they stand in the exception queue, named and counted, until they clear or are written off in the open.

6. Position is not conformance

Carry forward the distinction Module 1's worked example printed: **E measures whether the books are true; the position (E' in the house's gauge) measures whether the position is good.** A true book in a tight week conforms at 100 percent and reports a hard position beside it. A padded book in a flush week fails conformance no matter how comfortable the position looks. Report the two on separate lines, always, and never let a good position talk you out of checking the books, or a true book talk you into calling a hard position fine.

7. Reading this module's two builds

By the end of this week your gauge holds two new things: one error that used to recur and now cannot proceed, with the stopped attempt on record, and Component E with its exception queue standing open. Notice that both builds share one shape: stop it where it happens, and print what you could not clear. Modules 5 and 6 take that shape into the two places errors hide best: work still in process, and the almos that never quite happened.

PART B

The Worked Example

Continuing the teaching example: the billing team (Component A 87.3% · PPC 72.2% · work-in-process cap 12) takes its most recurring error, and builds its Component E in the same week.

The error, counted

Reviewing the last full month's rework, the team found its champion: **an application keyed to the wrong account number**, caught later, every time, but only after processing had been done against the wrong account. Counted in the stated window: **17 occurrences in 200 applications**, **8.5%** of the month's work touched by one error. Two “be careful” reminders had already been issued for it in earlier months. The error had outlived both.

The validation, demonstrated

The team built the check into the intake step: the account number is now verified against the source document before processing begins, by format and check-digit, and the processing screen will not open the application while the number fails. A mismatch stops the work at intake, where fixing it takes a minute, instead of after processing, where it took an afternoon. The demonstration, run and recorded as this module requires: the lead deliberately keyed one of the month's known wrong numbers at the validation. **It stopped. The application could not proceed.** That stopped attempt is line one of the team's proof.

Component E, first count

Line	Count	Figure
Records in the window (one month of the team's books)	120	
Reconciled clean under the team's stated rule	114	
Standing in the exception queue (named, owned, dated)	6	
Component E = 114 ÷ 120		95.0%

How to read the 95.0%. It is not “we lost 5 percent somewhere.” The six exceptions are listed by name with owners and ages; two cleared within the week. The queue is the reason the 95.0% can be believed: everything that does not reconcile is standing somewhere you can point to. The day the queue is empty and the count says so, E reads 100%, and not one day before.

PART D

The Field Exercise: Kill One Error: and Open the Queue

Your artifact: one validation, built and working, that makes one recurring error impossible, with the error, the validation, and the proof logged in the app. In the same week: your Component E rule written and your exception queue opened with its first items. **Time:** 30–60 minutes. Module 5 opens once this is logged, at your own pace, logged, not graded.

The artifact carries six things

1. The **error**, named plainly, with its count in a stated window: this error, N times, in these weeks.
2. The **validation**: where it sits, what it checks, and how it blocks. (A reminder, poster, or sterner instruction does not meet this specification: the error must be unable to proceed.)
3. The **demonstration**: the old error run at the validation once, and the result, recorded.
4. Your **Component E rule**: what footing means line by line, when the check runs, and what happens to what will not clear.
5. Your **exception queue**, opened: its first items listed with owners.
6. Your first **Component E figure** if your window allows a count this week, or the date of the first count, stated, if it does not.

The five steps

1. **Find and count the error (≈15 min)**. Pick the error your operation has made more than once. Count it in a stated window. If you cannot count it, pick the one you can. The count is part of the kill.
2. **Build the validation (≈20 min)**. Put the check at the point of the error, blocking, in whatever material your work allows: a form rule, a template formula, a physical fit, a named second-person check.
3. **Demonstrate it (≈5 min)**. Run the old error at the validation. Record what happened, in one line, with the date.
4. **Write the E rule and open the queue (≈15 min)**. State the rule; list the first exceptions by name with owners.
5. **Date, sign, log (≈5 min)**.

Worksheet

Field	Yours
Error (named) · count in window	
Validation: where it sits · how it blocks	
Demonstration: date · result	
E rule: footing means · check runs · unclear items go	
Exception queue: first items · owners	

Pitfalls:

- **The warning in validation's clothes:** a message that can be clicked past, a note that can be ignored. If the error can proceed, it will, in your busiest week.
- **Proof by description:** “it should stop it.” Run the old error at it. The demonstration is the artifact's spine.
- **An exception queue that empties by fog:** items leave the queue by clearing or by an open write-off, never by ageing off the page.
- **Blending position into E:** a comfortable month is not conformance. Count the records, not the mood.

Done looks like this: an error that used to recur, a validation that stops it cold, the stopped attempt on record, an E rule a stranger could run, and an exception queue holding what remains. Log it. Next, Module 5, Stop the Line, gives everyone on your team the authority the validation has by design: the authority to halt the work the moment a defect appears.

Check yourself

Q. Your validation shows a warning the user can click past. Does it meet this module's specification?

A. No. A validation blocks: while the error is present, the work cannot proceed. A message that can be clicked past is a warning wearing a validation's clothes.

Q. Six of 120 records stand in the exception queue, named and owned. What is Component E, and why does the queue make the figure believable?

A. $114 \div 120 = 95.0\%$. The queue holds everything that does not reconcile, in the open and counted, so the conforming share is not achieved by sweeping.

Q. Your books are true and your position is bad. How do E and the position line read?

A. Separately, always: E can stand at 100% while the position line prints the hard position beside it. Conformance measures truth; the position measures comfort; neither may borrow the other's number.

THE OPENING: MODULE 4

Artifact logged → Module 5 opens at your own pace. The opening is the logged artifact: the counted error, the working validation with its recorded demonstration, the E rule, and the opened exception queue. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: poka-yoke error-proofing (Toyota's published method), continuous close (this book's name for reconciling as work happens), and medication reconciliation as a cross-industry patient-safety practice; the payer discipline is taught as a discipline, with any exact band labelled INFERENCE on the plan's face. **Shows:**

worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 4 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Stop the Line

Authority to halt the work, held by the people doing it

This module's artifact: your team's stop rule, written, with the save log opened, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** the in-process catch line: defects caught while the work can still be saved, counted separately from escapes. (Capstone arc stage: Candor.)

LESSON PLAN: MODULE 5

The one concept	Andon authority: any person who sees a defect may stop the work, and the stop is honoured, not punished.
Objectives	By the end of this module you can: (1) state, from the source story, what changed when stop authority reached the floor; (2) write your team's stop rule, who may stop what, for what, and what happens next, in one paragraph; (3) open a save log that records each stop, what it caught, and what it cost; (4) state the rule that protects the person who stopped the line in good faith.
Source story	NUMMI: General Motors' worst-performing plant, reopened under Toyota's system, transformed within one year RECORD .
Builds in your gauge	The in-process catch line: defects caught while the work can still be saved, counted separately from escapes. (Capstone arc stage: Candor.)
Worked example	One week on the billing team's floor: the stops as logged, what each caught, and the catch line as counted, beside the week's escapes.
Field exercise	Write your team's stop rule; open the save log. Specification in full in Part D.
The opening	Artifact logged → Module 6 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The plant that changed by stopping

In 1982, General Motors closed its Fremont, California plant, by the company's own account its worst-performing factory, with quality to match. Two years later the same building reopened as NUMMI, a joint venture run under Toyota's production system, staffed in large part by the same workforce. Within about a year, the plant's quality had transformed, from worst in the company to among its best. The equipment was not new. The people were substantially the same. What changed was the system they worked inside, and its most visible difference hung above the line on a cord.

The cord is the **andon**: any worker who sees a defect pulls it, and the pull signals: a light and a tone sound, and the team leader comes to the spot at once. Worker and leader work the problem at the station; the line keeps moving until the piece reaches the end of its zone, and stops then if the problem is still unresolved. Restarting is part of the design, not an afterthought: the stop has a stated end, or it is a stall. Management's job at a signal is to help, in person, fast. The worker who pulled the cord is thanked. That is the whole technology: **authority to halt the work, held by the people doing it, honoured every time it is used.**

2. Why stopping beats catching later

Every defect has a cheapest moment: the moment it is made. Left alone, it travels, into the next step, the next person's work, the delivery, the customer's hands, gathering cost at every stop and arriving, eventually, as what Module 1 taught you to call an **escape**: finished work that landed wrong. The arithmetic of stopping is brutal and simple. A stop costs minutes, in the open, where the cause is still warm. An escape costs rework, apology, and trust, and it teaches everyone watching that seeing a defect and saying nothing is a survivable strategy, because it was.

Most operations already have the knowledge to stop their escapes. It lives in the people closest to the work, who see the wrong figure, the mismatched file, the step that smells off, and keep going, because stopping is not their place, or because the last person who stopped something was treated as the problem. Andon is not a detection system. Detection was never the scarce resource. **Permission is.**

3. The catch line: what your gauge gains

This module adds a line to your gauge: the **in-process catch line**: defects caught while the work can still be saved, counted separately from escapes, forever. The two lines are never blended, for the same reason E and position never blend: they answer different questions. Escapes ask what got out? The catch line asks is the catching machinery alive?

Catch share = defects caught in process ÷ (defects caught in process + escapes), counted over the same window.

A rising catch line beside a falling escape count is the system learning. A catch line at zero beside escapes above zero is not a clean operation. It is a silent one.

Read the zero case carefully, because it fools managers everywhere. If your operation reports no catches and some escapes, the defects were there to be caught and nobody caught them, or caught them and said nothing. Silence is not quality. This series counts catches on their own line precisely so that catching is visibly valued: the desk that catches three defects in-process has done something the gauge should honour, and Module 1's worked example printed its own catches on exactly this principle: 3 in-process catches, 0 escapes among them, counted separately so the catching machinery gets its own line.

4. The stop rule, in one paragraph

Authority that lives in custom dies in the first busy week. Authority written down survives. Your stop rule is one paragraph, and it must answer five questions:

1. **Who** may stop the work? (The honest answer in this system: anyone who sees a defect, including the newest person on the team.)
2. **What** may they stop? (Name the scope: a step, a batch, a delivery, the whole line.)
3. **On seeing what?** (A defect, a mismatch, a step that cannot be verified, name the trigger in words a busy person can apply.)
4. **By what act?** (The literal move: the word said, the queue flagged, the file held. A stop nobody can perform is a slogan.)
5. **What happens next, and how does work restart?** (Who comes, what gets fixed or contained, and who calls the restart, so a stop is an event with an end, not a stall with a smell.)

5. The protection is the policy

One sentence in the rule does more work than the other five combined: **a good-faith stop is recorded as a save, never as a fault**, even when the stop turns out to have been unnecessary. Especially then. The first time a person stops the line for what proves to be nothing and is made to feel foolish for it, the cord goes dead for a year, quietly, in a way no dashboard will show you until the escapes arrive. You are not buying stops. You are buying the next stop: the real one, three months from now, seen by a junior person at a bad moment. Protect the mistaken stopper in writing, in advance, by name of the rule, or do not pretend to have a stop rule at all.

6. The save log: what a stop is worth

Every stop is logged in the **save log**: date · who stopped · what was caught · what it would have cost if it had escaped · how work restarted. The fourth column is the argument-ender. Stops will be questioned: they cost visible minutes in a world that counts visible minutes and invisible

escapes. The log answers with the ledger: here is what these minutes bought, in errors that never shipped. Keep the cost column honest (estimate plainly and label estimates as estimates) because a save log that inflates becomes a story, and the operation has enough stories already. That is what the gauge is for.

PART B

The Worked Example

Continuing the teaching example: the billing team's first week under its written stop rule (Component A 87.3% · PPC 72.2% · cap 12 · Component E 95.0%).

The rule, as the team wrote it

“Any team member who sees an application being processed against the wrong account, a missing second check, or a batch that does not match its intake count may hold that work, by flagging the batch and saying ‘hold’ in the team channel. The lead comes to the batch before it moves. Work restarts when the mismatch is resolved or the batch is set aside by name. A hold called in good faith is logged as a save, never as a fault, including a hold that turns out to be unnecessary.”

The week as logged

Line	Count
Stops called (holds)	7
Of the stops: defects confirmed and fixed in process	6
Of the stops: no defect found, logged as saves, no fault recorded	1
Total defects caught in process this week (stops + routine second checks)	14
Escapes this week (errors found after handover)	2
Catch share = 14 ÷ (14 + 2) = 14 ÷ 16	87.5%

The eight routine catches, itemised from the team's own second-check record so the 14 is a count and not an assertion: 3 applications keyed with a transposed digit in the amount, caught at the second check · 2 batches whose intake count disagreed with the batch header, recounted and corrected in process · 2 files routed to the wrong work queue, caught before processing · 1 duplicate application caught at the second check before a second payment ran. Stops 6 + routine catches 8 = the 14 the catch line counts.

Read the three numbers the way the gauge reads them. The **7 stops** are the authority being used: a first week in which nobody dared pull the cord would be the failure, however quiet the floor felt. The **one stop that found nothing** is printed without apology; it is the premium paid

for the six that found something. And the **87.5% catch share beside 2 escapes** is the honest whole picture: the catching machinery is alive and most defects die at home, and two still got out, named in the escape record with their classes, for Module 6's register and Module 9's counting to learn from.

PART D

The Field Exercise: Write the Stop Rule: Open the Save Log

Your artifact: your team's stop rule, written, with the save log opened, logged in the app.

Time: 30–60 minutes. Module 6 opens once this is logged, at your own pace, logged, not graded.

The artifact carries four things

1. The **stop rule**, in one paragraph: who may stop what, on seeing what, by what act; what the team does when a stop is called; and how work restarts.
2. The **protection**, stated in the same artifact: a good-faith stop is recorded as a save, never as a fault.
3. The **save log**, opened, with its columns standing ready: date · who stopped · what was caught · what it would have cost if it had escaped · how work restarted.
4. **Proof of publication:** the rule delivered to the team it governs, with the date it was delivered. A rule nobody has seen stops nothing.

Working alone. The stop rule runs with one pair of hands: anyone becomes you, and the hold becomes a named pause, the work flagged and not handed over until you have re-checked it against its source, after a break where the work allows. The save log runs the same way, and its first line may be a stop you called on yourself. The form is weaker (no second pair of eyes) and the artifact labels it the solo form; the protection sentence matters more, not less, when the person you must protect from contempt for a false alarm is you.

The four steps

1. **Write the rule (~20 min).** One paragraph. Answer the five questions from Part A, section 4, in your operation's own words and with its own literal act of stopping.
2. **Write the protection (~5 min).** One sentence, unhedged, naming the good-faith stop as a save, including the stop that finds nothing.
3. **Open the save log (~10 min).** Five columns, in whatever material your team actually uses. If a stop has already happened informally in recent memory, you may enter it as the first line, dated as memory, labelled as such.
4. **Publish, date, sign, log (~10 min).** Deliver the rule to the team it governs. Then log the artifact.

Worksheet: the save log, opened

[illegible]

Pitfalls:

- **Authority for some:** a rule where only senior people may stop. Defects do not check seniority before appearing; the newest eyes often see them first.
- **The stop with no restart:** a rule that halts work and names no way back teaches people never to halt it.
- **Protection with an asterisk:** “good-faith stops are protected, within reason.” The asterisk is read by the floor as the rule. Write it plain or expect silence.
- **Costs inflated for effect:** the save log's cost column estimated wildly to win an argument. Label estimates; keep the log believable, or lose it.

Done looks like this: a stop rule your team has actually received, protection written in plain words, and a save log standing open for its first entry. Log it. Next, Module 6, Report the Almost, goes one step earlier than the catch: the defect that nearly happened, and the register that learns from it before it ever ships.

Check yourself

Q. A stop is called and no defect is found. How is it logged?

A. As a save, never as a fault. The protection covers the good-faith stop that finds nothing, especially that one: punish it, and the cord goes dead for a year.

- Q.** Your operation reports zero catches and three escapes this week. What does the gauge say?
- A.** That the catching machinery is silent, not that the work is clean: the defects were there to be caught. A catch line at zero beside escapes above zero is a finding.
- Q.** You work alone. What is your stop rule's honest form?
- A.** A named hold on your own page: the work is flagged and does not proceed to handover until you have re-checked it against the source, ideally after a break; the save log runs the same way, labelled as the solo form it is.

THE OPENING: MODULE 5

Artifact logged → Module 6 opens at your own pace. The opening is the logged artifact: the written stop rule, the protection stated, the save log opened, and the rule published to the team. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: andon stop authority as practised in Toyota's system and documented in NUMMI's published record.

Shows: worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 5 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Report the Almost

Near misses are free lessons, if the system thanks the reporter

This module's artifact: a near-miss register, opened, with the first three near misses filed, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** the near-miss and alert counts as leading indicators, beside the catch line. (Capstone arc stage: Candor.)

LESSON PLAN: MODULE 6

The one concept	Near-miss learning; safety alerts: report what almost happened, count it, and learn before the escape arrives.
Objectives	By the end of this module you can: (1) distinguish an escape, an in-process catch, and a near miss, and classify three examples correctly; (2) open a near-miss register with a filing rule a busy person can actually follow; (3) file the first three near misses from your own operation, each with what almost happened and what would have followed; (4) state why a rising near-miss count is read as a leading indicator, not as deteriorating work.
Source story	NASA's Aviation Safety Reporting System, operating since 1976 RECORD · Virginia Mason's patient safety alert counts RECORD .
Builds in your gauge	The near-miss and alert counts as leading indicators, beside the catch line. (Capstone arc stage: Candor.)
Worked example	The billing team's first quarter on the register: the counts by month, and one almost that changed a rule before it could become an escape.
Field exercise	Open a near-miss register; file the first three. Specification in full in Part D.
The opening	Artifact logged → Module 7 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The lesson that costs nothing

Every escape your operation has ever shipped was, some hours or days earlier, an almost: a moment where the wrong thing nearly happened and somebody, or something, or plain luck, intervened. The almost carries the same information as the escape: the same weak step, the same missing check, the same trap in the work. It differs in exactly one way. **It costs nothing.** No rework, no apology, no customer holding the error in their hands. A near miss is the escape's lesson, delivered free, to any operation organized enough to collect it.

Most operations are not. The almost happens, the person involved exhales, fixes it quietly, and moves on, rationally, because reporting it has historically bought them a conversation about their competence. This module builds the machinery that reverses that trade: a register, a filing rule that protects the reporter, and a way of counting that reads a rising pile of almos as the health sign it is.

2. Three kinds of almost-trouble: keep them separate

Your gauge now carries three distinct lines, and this module's first skill is never confusing them:

Kind	What happened	Where it counts
Escape	Defective work crossed the delivery point and was found afterward, by you, or worse, by the recipient.	Component D's count (Module 9); the escape record.
In-process catch	A defect existed in the work and was caught before delivery, by a check, a second person, a stop (Module 5).	The catch line (Module 5).
Near miss	No defect shipped and none was in finished work, but the conditions for one assembled, and only an intervention or luck prevented it.	The near-miss register: this module.

Classify three, to fix the boundaries. A processor keys the wrong account number and the Module 4 validation blocks it: that is a **catch**: the defect existed and a designed check caught it. A processor reaches for the wrong file, notices the name a half-second before opening it, and takes the right one: **near miss**: nothing was wrong in any work product; the trap was simply armed and a human reflex disarmed it. The wrong statement goes out and the customer calls: **escape**. The register takes the second kind. Its value is precisely that nothing went wrong, yet.

3. The system that proved reporting works: ASRS

Aviation faced this problem at its starkest. Pilots and controllers saw the almos, the misread clearance, the runway nearly entered wrong, and stayed silent, because reporting your own almost to your regulator is a good way to end a career. In 1976 NASA opened the **Aviation Safety Reporting System**: reports filed to NASA, not to the regulator; identifying details stripped; reporters protected from enforcement use of their own reports in the ordinary case.

The almoses poured in, well over a million of them over the system's life, and aviation got something no accident investigation can provide: a map of its traps, drawn while the traps were still only almoses. The design lesson is the protection, not the form: **people report almoses to a system that thanks them and shields them, and stay silent to a system that grades them.** Your register is small. The human arithmetic inside it is identical.

4. Alerts in a hospital: Virginia Mason

Virginia Mason, a Seattle hospital system, carried the same design into care with its Patient Safety Alert system: any staff member who sees a condition that could harm a patient raises an alert, leadership responds, and (the part this course asks you to notice) **the alert counts are reported as a sign of health.** When alerts rose after the system opened, that rise was read correctly: not as care getting worse, but as seeing getting better. The traps had always been there. Now they had witnesses.

5. Leading, not lagging: how to read the count

Escapes and catches are lagging indicators: they count events that already reached, or nearly reached, the work. The near-miss count is a leading indicator: it moves before the escapes do. Two readings to internalize:

- **A rising near-miss count, early, is good news.** It almost always means reporting is becoming safe, not that work is becoming dangerous. Judge it alongside escapes: near misses up, escapes flat or falling is the signature of a system learning faster than its traps can fire.
- **A near-miss count of zero is a finding, not a comfort.** No operation runs a quarter with no almoses. A zero on the register means the register is not being used: the same silence signal as a catch line at zero (Module 5), one step earlier and cheaper to fix.

6. The filing rule a busy person can follow

Registers die of friction. If filing takes twenty minutes and a form with fourteen fields, the floor will file nothing, and they will be right to. The work is the work. The rule on the register's face answers four questions in a breath: **who may file** (anyone, including about their own almost) · **how long it takes** (two minutes; five fields) · **what happens to the reporter** (thanked, never blamed, in writing, same doctrine as Module 5's protection) · **what happens to the filing** (read at the weekly review; a filing that changes a rule says so on its own line). The five fields: date · reporter · what almost happened · what would have followed · what, if anything, changed as a result. That last field is allowed to say nothing yet, an honest nothing, reviewed weekly, beats a register where every entry performs a lesson it did not teach.

PART B

The Worked Example

Continuing the teaching example: the billing team's near-miss register across its first quarter of operation, beside the lines it stands next to on the gauge.

The counts, by month

Month	Near misses filed	Catches (Module 5 line)	Escapes
Month 1	3	9	3
Month 2	8	12	2
Month 3	12	14	2
Quarter	23	35	7

Read the quarter the way Part A taught: filings rose from 3 to 12 a month while escapes fell from 3 to 2. The operation did not get sloppier across the quarter. It got more willing to say what it saw. The 23 filings are 23 traps now named on paper. The 3 filings in month 1 are the honest measure of how many traps were being named before the register existed, in an operation that was having, by its later count, at least a dozen almosts a month all along.

The almost that changed a rule

Filing 14, month 2. A processor batching statements for two customers with similar names attached the right statements to the right envelopes, by eyesight, at speed, and only because the second customer's name caught her eye at the seal. What almost happened: two customers each receiving the other's statement. What would have followed: a privacy escape, the worst class this team can ship. What changed: statements are now matched to envelopes by the account-number check built in Module 4, not by name. The validation gained a second job, and name-matching was struck from the procedure in writing. One almost, costing nothing, bought the fix that an actual escape would have bought at full price, plus the apology, plus the disclosure the escape would have owed.

PART D

The Field Exercise: Open the Register: File the First Three

Your artifact: a near-miss register, opened, with the first three near misses filed, logged in the app. **Time:** 30–60 minutes. Module 7 opens once this is logged, at your own pace, logged, not graded.

The artifact carries four things

1. The **register itself**, with its five fields standing open: date · reporter · what almost happened · what would have followed · what, if anything, changed as a result.
2. The **filing rule**, written on the register's face: who may file, how long a filing takes, and the standing assurance that a filing is thanked, never blamed.

3. **Three near misses, filed:** real ones from your own work, however small. An “almost” you nearly forgot counts; that is the material this register exists for.
4. Your name and the date as the register's opener.

The four steps

1. **Open the register (≈10 min).** Five fields, in the material your team actually uses. A register kept somewhere nobody goes is a diary.
2. **Write the filing rule (≈10 min).** On the register's face. Keep it to what a busy person can hold: who, how long, thanked-never-blamed, read at the weekly review.
3. **File the first three (≈20 min).** Ask the team (and ask yourself) for the almosts of the last few weeks. Classify carefully: if a defect existed in work and a check caught it, it belongs on the catch line, not here. This register takes the ones where nothing shipped wrong but the trap was armed.
4. **Date, sign, log (≈5 min).**

Worksheet: the register's first page

Date	Reporter	What almost happened	What would have followed	What changed
Filing rule (on the register's face):				
Pitfalls: • Filing catches as near misses (or the reverse): the lines answer different questions and blending them corrupts both. Defect existed and was caught = catch. No defect in any product, trap merely armed = near miss. • A first filing about someone else: a register whose opening entries are all one person's near misses teaches the whole team to keep theirs private. File your own first. • Every filing must perform a lesson: forcing a “what changed” onto every entry manufactures lessons. Nothing yet is an honest entry; the review is where lessons get made. • Reading a rise as decline: reporting the growing count upward as deteriorating work. Print the escape line beside it and read them together, or the register will be punished into silence.				

Done looks like this: an open register, a filing rule that protects the reporter, and three honest almossts standing in it. Log it. Next, Module 7, Read It Back, takes the steps in your operation that cannot be undone, and puts a second voice on them before they are taken.

Check yourself

Q. A processor keys the wrong account and the Module 4 validation blocks it. Catch or near miss?

A. A catch: the defect existed in the work and a designed check stopped it. A near miss is when no defect stood in any product and only an intervention or luck disarmed the trap.

Q. Filings rise from 3 to 12 a month while escapes fall. How is the rise read?

A. As reporting becoming safe, not work becoming dangerous: a leading indicator, read beside the escape line, never alone.

Q. A quarter shows zero near misses filed. Comfort or finding?

A. A finding. No operation runs a quarter with no almossts; a zero on the register means the register is not being used.

THE OPENING: MODULE 6

Artifact logged → Module 7 opens at your own pace. The opening is the logged artifact: the opened register, its filing rule, and the first three near misses filed. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: near-miss reporting and safety-alert practice as documented in NASA's Aviation Safety Reporting System (operating since 1976) and Virginia Mason's published alert practice. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 6 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Read It Back

Checklists with read-back, on the steps that cannot be undone

This module's artifact: one checklist for one irreversible step, with one read-back run and recorded, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** delivery discipline: the first of the delivery-stage practices that keep Component D honest. (Capstone arc stage: Delivery.)

LESSON PLAN: MODULE 7

The one concept	Checklists with read-back on irreversible steps: the doer reads the step aloud; another voice confirms it before the step is taken.
Objectives	By the end of this module you can: (1) name the steps in your operation that are irreversible, and say what makes each one so; (2) write a checklist for one irreversible step that fits the moment it is used in; (3) run one read-back on live work and record what it caught or confirmed; (4) state the difference between a checklist that is performed and one that is merely possessed.
Source story	Aviation checklist culture and crew resource management RECORD .
Builds in your gauge	Delivery discipline: the first of the delivery-stage practices that keep Component D honest. (Capstone arc stage: Delivery.)
Worked example	The billing team's send step: the checklist as written, and the read-back log of its first month, catches and confirmations alike.
Field exercise	Pick one irreversible step; write its checklist; run one read-back. Specification in full in Part D.
The opening	Artifact logged → Module 8 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The steps that do not forgive

Every operation has a short list of steps after which there is no undo. The message is sent. The file is deleted. The payment leaves. The statement is sealed in the envelope, the dose is given, the announcement is published. Before those steps, an error is a catch, cheap, private, fixable. After them, the same error is an escape with a recipient. This module puts a practice on exactly those steps, borrowed from the industry that learned it at the highest possible price: aviation's checklist, performed as a **read-back**: one person reads each check aloud, a second voice confirms it from the actual state of the work, and only then does the step proceed.

2. Why aviation reads aloud

The checklist was born from a crash in 1935: the Boeing Model 299, the prototype of the B-17, an aircraft too complex for even a master test pilot to hold entirely in memory, lost because one control lock was forgotten. The industry's answer was not better pilots or sterner memory. It was a written list, worked aloud, by a crew. The wider discipline that grew around the list, crew resource management, is a development of the 1970s and 1980s, not of 1935, and this book dates it so: the junior voice is not decorative, the challenge-and-response is spoken, and the confirmation comes from looking at the thing itself, not from remembering having done it. The pattern has held for the better part of a century because it respects a fact every other module in this series keeps meeting: **attention fails, memory flatters, and a second voice at the moment of no return is the cheapest verification ever invented.**

3. Performed, not possessed

Most operations already possess checklists. They live in binders and shared drives, thorough, current-ish, and absent from the moment they were written for. A checklist that is possessed protects the organization's story about itself. A checklist that is performed protects the step. The difference is three properties:

- **It fits the moment.** It is short: the few checks that must pass, in the order they must pass, and it is physically or digitally present at the step, not retrievable after a search.
- **It is spoken.** Read-back means aloud: the doer reads the item, the second voice confirms from the work itself, the doer hears the confirmation before proceeding. Silent ticking by the doer alone is a memory test with boxes.
- **It can stop the step.** A check that fails halts the irreversible act: the envelope is not sealed, the send does not happen, under the same protection doctrine as Module 5: the voice that halts it has done the operation a service, on the record.

4. Choosing your irreversible steps

List your operation's steps and ask of each: if this is wrong, what does undoing it cost? Cheap and quick: the step needs no read-back; your catch machinery (Modules 4–5) covers it. Expensive, public, or impossible: the step is a candidate. Sent communications, deletions, filings,

payments, publications, administrations: most operations find between three and ten true irreversibles. That shortness is good news. Read-back is a strong medicine with a real cost, a second person's minutes, and it belongs exactly where the cost of the error dwarfs the cost of the minute: on the few steps that cannot be taken back.

5. Writing a checklist that survives a busy day

Write for the worst moment, not the best: tired, hurried, interrupted. Five rules from the floor:

1. **Checks, not instructions.** Each line names a state to confirm, “recipient matches the account on the work order”, not a task to perform. The doing happened before the list; the list verifies.
2. **Kill-or-confirm order.** The checks that would stop the step come first, while stopping is still cheap and unembarrassing.
3. **Short enough to be used at full speed.** If the list takes longer than the step it guards, the floor will route around it on the first busy day, correctly. Cut to the checks whose failure would actually stop you.
4. **One list per step.** A master checklist for everything is a binder again. The send step gets the send list; the delete step gets the delete list.
5. **The list is a record of record.** When a read-back catches something, the catch is written: it feeds the catch line (Module 5) and teaches the list's next revision. When a step escapes despite its list, the list itself is examined, in the open, at the review.

6. What this builds in your gauge

Read-back is the first of the delivery-stage practices: the discipline that will keep Component D honest when you formalize it in Module 9. D counts how your handovers landed. Practices like this one (and independent verification, next module) are why they land clean. Keep the run record as you go: date, step, who read, who confirmed, what it caught. A month of those records is the first evidence your delivery discipline is real rather than asserted, and assertion, as every module so far has said in its own way, is not evidence.

PART B

The Worked Example

Continuing the teaching example: the billing team takes its most irreversible step, the weekly statement send, and puts a read-back on it.

The step and its list

Once a week the team sends its processed statements out, the moment after which a wrong statement is a customer's problem and a privacy risk, not a file the team can quietly correct. The checklist, as written, in use order:

1. Batch count on screen matches the intake count for this send.

2. Every statement in the batch passed the account-number check (Module 4): no held items inside the batch.
3. Each statement is matched to its envelope by account number, not by name.
4. The send list names today's recipients and no others: last week's list is not reused.
5. Any batch flagged 'hold' under the stop rule (Module 5) is physically out of this send.

The first month of runs

Line	Count
Sends in the month	4 weekly sends, 25 batches total
Read-backs run	25 of 25
Catches at read-back	3: two batches carrying a held item, one batch whose count did not match intake; all stopped before sending
Escapes from the send step in the month	0

How to read the 3 catches. All three would have shipped. Two were Module 5 holds that had been set aside by name and then drifted back toward the send pile, a failure of housekeeping the checklist caught at the last cheap moment, and which the team then fixed at the source by moving held batches to a separate tray (a Module 4-style design change, prompted by a Module 7 catch: the practices compound). The read-back is not the deepest defence the team has. It is the last one, which is exactly why it stands on the step that cannot be undone.

PART D

The Field Exercise: One Step That Cannot Be Undone: One Read-Back

Your artifact: one checklist for one irreversible step, with one read-back run and recorded, logged in the app. **Time:** 30–60 minutes. Module 8 opens once this is logged, at your own pace, logged, not graded.

The artifact carries four things

1. The **step**, named, and one line on what makes it irreversible: what undoing a wrong version of it would cost.
2. The **checklist**: the few checks that must pass before the step, in the order they must pass, short enough to be used at full speed.

3. The **run record**: one read-back on live work: date, step, who read, who confirmed, and anything the read-back caught (or the plain statement that it confirmed all checks).
4. Your name and the date.

Working alone. A solo operator has no second voice, and this book does not pretend otherwise. The solo form is record-and-replay: read the checklist aloud into a recording, then play it back against the work itself, confirming each check from the thing, not from memory, before the step proceeds. It catches less than a live second voice (you bring your own expectations to both readings), and the run record labels it solo, record-and-replay so the weaker form is never dressed as the stronger one.

The five steps

1. **Pick the step (≈10 min).** From your operation's irreversibles: sent, published, deleted, administered, signed, paid. Choose the one whose error would cost most.
2. **Write the list (≈15 min).** Checks, not instructions; stoppers first; short enough for the busiest day. Test each line: if this check failed, would we actually stop? If not, the line is decoration. Cut it or make it real.
3. **Place the list at the step (≈5 min).** Where the hands are when the step happens. Not in a drive. At the step.
4. **Run one read-back on live work (≈15 min).** Aloud. One reads, one confirms from the work itself. Record the run.
5. **Date, sign, log (≈5 min).**

Worksheet: the run record

Field	Yours
Step (and why it is irreversible)	
Checklist: checks in order	
Run: date · reader · confirmer	
Caught, or confirmed clean	

Pitfalls:

- **The binder list:** forty thorough checks nobody performs. Short and performed beats complete and possessed, every week of the year.
- **Self-read-back:** the doer reading and confirming their own work in one voice. That is memory with extra steps. The second voice is the mechanism.
- **Confirmation from memory:** the confirmer answering from what they believe was done instead of looking at the work in front of them. The rule is: confirm from the thing, not from the story.
- **Read-back on everything:** putting lists on reversible steps too, until the practice is resented everywhere and performed nowhere. Reserve it for the steps that cannot be undone; that scarcity is its authority.

Done looks like this: a checklist short enough to survive contact with a busy day, standing at the step it guards, and one real read-back standing behind it in the record. Log it. Next, Module 8, Verified by Another, widens the second voice into a rule for whole deliverables: the verifier is never the builder.

Check yourself

Q. What three properties make a checklist performed rather than possessed?

A. It fits the moment (short, present at the step); it is spoken (read-back, confirmed from the work itself); and it can stop the step when a check fails.

Q. The doer reads the list and confirms their own items from memory. What is missing?

A. Both halves of the mechanism: a second voice, and confirmation from the thing itself rather than from the story of having done it.

Q. Working alone, what is the read-back's honest solo form, and how is it labelled in the run record?

A. Record-and-replay: read the checks aloud into a recording, play it back against the physical work, confirm each from the thing itself. The record labels it solo, record-and-replay, the weaker form named as weaker.

THE OPENING: MODULE 7

Artifact logged → Module 8 opens at your own pace. The opening is the logged artifact: the named irreversible step, its checklist, and one recorded read-back run on live work. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: checklist discipline and read-back as practised in aviation's checklist culture and crew resource management, both published and documented. **Shows:** worked examples at the WHAT level: what was

produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 7 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Verified by Another

The verifier is never the builder

This module's artifact: one live deliverable under independent verification, verifier named, standard stated, outcome recorded, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** delivery discipline, completed: verification standing between delivery and claim. (Capstone arc stage: Delivery.)

LESSON PLAN: MODULE 8

The one concept	Independent verification: work is verified by someone who did not build it: verifier $\neq$ builder, as a rule, not a courtesy.
Objectives	By the end of this module you can: (1) state why self-verification misses what independent verification catches, from the source record; (2) name a verifier who is not the builder for one live deliverable, with the verification standard stated in advance; (3) record the verification's outcome, passed, or returned with findings, without editing the record; (4) state what happens to a finding: who owns the repair, and why the repair is new work, not an erasure.
Source story	NASA's Independent Verification and Validation Program RECORD : the institutional record of verifier-separated-from-builder, kept by the agency since the early 1990s.
Builds in your gauge	Delivery discipline, completed: verification standing between delivery and claim. (Capstone arc stage: Delivery.)
Worked example	A month of the billing team's verifications: the standard as stated, the outcomes as recorded, returns included.
Field exercise	Name a verifier who is not the builder for one live deliverable. Specification in full in Part D.
The opening	Artifact logged → Module 9 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The eyes that built it cannot grade it

Every builder reads their own work with the intention still in their head. They see the figure they meant to type, the step they meant to include, the file they meant to attach. This is not carelessness. It is how human reading works: we perceive what we expect, and the builder expects their own work to be right, because they remember making it right. Self-verification therefore has a structural blind spot exactly where the errors live: in the gap between what was intended and what was actually done. An independent verifier has no intention to read from. They have only the work, and the standard. That poverty is their power.

This module makes the arrangement a rule, in the plainest possible form: **verifier ≠ builder**. Work is verified by someone who did not build it, as a standing rule of the operation, not as a favour asked when time allows.

2. Learned in writing, and at cost

The defense and aerospace world carries this rule as IV&V, independent verification and validation, and it carries it the way it carries most of its hard rules: written down after the failures that taught it. Again and again, the investigations of failed missions and flawed systems found the same shape underneath: the team that built the thing also certified the thing, and the certification inherited the build's assumptions, including the wrong one. The institutional answer was separation, and its standing form is NASA's IV&V Program, stood up in the early 1990s with its facility at Fairmont, West Virginia: a verifier organizationally apart from the builder, working to a standard stated in advance, reporting findings the builder is obliged to answer but not entitled to edit. You do not need an agency's scale to use the rule. A four-person team can run it with a rota. What you cannot do is run it with the builder holding the pen.

3. The standard comes first

Independent verification is not a second opinion. It runs against a **standard stated before the verifier begins**: what “passes” means, item by item: the figures that must foot, the sources that must agree, the checks (Modules 4 and 7) that must show as run. Stating the standard in advance protects both parties. The builder knows what will be examined, so verification is not an ambush. The verifier knows what they are certifying, so a pass means something specific instead of “looked fine to me.” And when the outcome is a return, the findings cite the standard's lines, so the argument, if there is one, is about the work and the standard, never about taste.

4. The outcome is recorded without softening

Verification ends in one of two recorded outcomes: **passed**, or **returned with findings**. Both are written down exactly as returned. Three disciplines keep the record worth having:

- **The record is the verifier's words.** The builder does not edit, summarize, or “contextualize” findings. Article 2 of the doctrine is standing behind this: a fact reaches the record in the words it was written in.

- **A return is not a verdict on the builder.** It is the system working: the same doctrine as the good-faith stop (Module 5) and the thanked near-miss filing (Module 6). An operation where verifiers soften findings to keep the peace has independent verification in name only.
- **Findings get owners, and repairs are new work.** Each finding names who repairs it. The repair is then built and verified as new work, it does not reach back and erase the return from the record. The record keeps both: returned on Tuesday, repaired and passed on Thursday. Erasing the return would teach the next verifier that findings disappear, and findings that disappear stop being written.

5. Small teams, honest compromises

The objection arrives on schedule: we are three people; everyone touched everything. The rule bends without breaking. The verifier is the person **furthest from the making** of that deliverable, and the artifact says so, plainly: “verifier built the intake section only; verified the remainder.” A partial independence, honestly labelled, beats a total independence falsely claimed. What the rule will not survive is the builder verifying their own portion and calling the arrangement independent because a colleague glanced at the cover page. Label the distance truthfully, shrink it where you can over time: cross-training so next quarter's deliverable has a genuinely uninvolved verifier is a fine use of a quarter, and print the label either way.

6. Where verification stands in your gauge

This module completes the delivery stage's delivery discipline: read-back (Module 7) guards the irreversible step, independent verification stands **between delivery and claim**: work that has not been verified by another is work your operation may use, but may not yet claim: not in a report, not to a client, not on your own gauge as finished-and-verified. Notice that Module 1's very definition of a unit already required this: a unit counted only when it had passed the check. This module names who must hold that check for it to count. Next module, you will count delivery itself, Component D, formalized, and these two practices are why its number will deserve belief.

PART B

The Worked Example

Continuing the teaching example: a month of verifications on the billing team's outgoing deliverables, under a standard stated in advance (figures foot against source · account check run · no held items inside · recipient list current).

The month as recorded

Line	Count	Figure
Deliverables put to independent verification	20	
Passed first verification	16	
Returned with findings	4	
First-pass verification rate = 16 ÷ 20		80.0%
Of the returns: repaired, re-verified, and passed as new work	4 of 4	

The four returns, by finding class: a figure that did not foot against its source (2) · a held batch item inside a deliverable (1), the same drift the read-back caught in Module 7, now caught one stage earlier · a recipient list carried over from the prior week (1). Every return stands in the record in the verifier's words, with the repair beside it as separate, dated work.

How to read 80.0%. It is not the team's shame line; it is the verification line working at full honesty. One deliverable in five needed another pair of eyes to become right, and became right before a customer saw it, which is the entire difference between a return and an escape. Watch the two numbers side by side over time: if returns fall while escapes stay at zero, the builders are internalizing the standard. If returns fall to zero while the work is unchanged, ask whether the verifiers have started being kind. The record, kept unsoftened, lets you tell those two stories apart.

PART D

The Field Exercise: One Deliverable, Another's Eyes

Your artifact: one live deliverable under independent verification, verifier named, standard stated, outcome recorded, logged in the app. **Time:** about 55 minutes, the verification included. Module 9 opens once this is logged, at your own pace, logged, not graded.

The artifact carries five things

1. The **deliverable**, named: one live item now in production.
2. The **verifier**, named: a person who did not build any part of it. If everyone touched it, the verifier is the person furthest from its making, and the artifact says so.
3. The **standard**, stated before the verifier begins: what “passes” means, item by item.
4. The **outcome**, recorded exactly as returned: passed, or returned with findings listed.
5. For any finding: its **owner**, and the note that the repair is new work. Your name and the date close the page.

The five steps

1. **Choose the deliverable (≈5 min).** One live item, real, current, and headed for a recipient.
2. **Name the verifier (≈5 min).** Not the builder. Apply the furthest-from-the-making rule if the team is small, and write the label honestly.
3. **State the standard (≈15 min).** Before the verifier sees the work. Item by item: what must foot, what must agree, what must show as run.
4. **Verify and record (≈20 min).** The verifier works the standard against the deliverable itself, not against the builder's account of it. Record the outcome in the verifier's words.
5. **Own the findings, date, sign, log (≈10 min).** Each finding gets an owner. The page is logged as it stands, returns included.

Worksheet: the verification record

Field	Yours
Deliverable	
Verifier (and distance from the making)	
Standard: pass means, item by item	
Outcome: passed / returned with findings	
Findings · owner of each repair	

Pitfalls:

- **The friendly glance:** a colleague “looking it over” against no stated standard. Without the standard written first, there is nothing to verify against, and the pass certifies nothing.
- **The edited return:** findings softened before they reach the record. The verifier's words stand. Always.
- **Repair as erasure:** fixing the work and deleting the return, so the month reads as flawless. The return and the repair are two lines, both dated. That pair of lines is the sound of the system working.
- **Verification after claim:** the deliverable announced, reported, or promised before it is verified. Verification stands between delivery and claim, or it stands for nothing.

Done looks like this: a named verifier who built nothing on the item, a standard written before the check, and an outcome recorded without softening. Log it. Next, Module 9, First Pass, Every

Pass, formalizes Component D, and your gauge reads whole for the first time: $A \times D \times E$, the composite, all three components in.

Check yourself

Q. Why does self-verification miss what independent verification catches?

A. The builder reads with the intention still in their head and perceives what they meant to do. The verifier has only the work and the standard, and that poverty is their power.

Q. Your team is three people and everyone touched the deliverable. What does the rule do?

A. It bends without breaking: the verifier is the person furthest from the making, and the artifact labels the distance plainly. Partial independence honestly labelled beats total independence falsely claimed.

Q. A deliverable is returned with findings, then repaired. What stands in the record?

A. Both lines, dated: the return in the verifier's words, unsoftened, and the repair as new work, verified again. The repair never erases the return.

THE OPENING: MODULE 8

Artifact logged → Module 9 opens at your own pace. The opening is the logged verification record: deliverable, verifier, standard, and outcome as returned. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: independent verification and validation as institutionalized in defense and NASA practice and documented in their published record. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 8 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

First Pass, Every Pass

Count delivery the way it landed: the first time

This module's artifact: Component D, formalized, one week of first-pass delivery, counted, logged in the app as your Component D page. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** Component D (first-pass delivery yield), formalized: the gauge now reads whole: $\text{RATE} = A \times D \times E$, the house composite with all three components in. (Capstone arc stage: Delivery, completing the gauge on paper.)

The one concept	First-pass yield: of everything handed over, how much landed clean the first time, with the four delivery keys, translated for any operation.
Objectives	By the end of this module you can: (1) define “landed clean” for your operation (on time, handed over, no error found afterward) in one sentence; (2) formalize Component D under a counting rule written before the count; (3) count one full week of first-pass delivery and state D to one decimal; (4) state, in writing, what an external frame can and cannot do for a number like D: read your own failure share and recovery time from the same register beside D, and state what the comparison does and does not entitle you to claim.
Source story	DORA (DevOps Research and Assessment): a decade of published delivery research, Forsgren, Humble & Kim, Accelerate (2018) and the DORA program's reports RECORD . DORA publishes performance levels per key; it publishes no band for a percentage like D, and this module borrows none.
Builds in your gauge	Component D (first-pass delivery yield), formalized: the gauge now reads whole: $RATE = A \times D \times E$, the house composite with all three components in. (Capstone arc stage: Delivery, completing the gauge on paper.)
Worked example	The billing team's first full Component D week, and its gauge, read whole for the first time: $A \times D \times E$.
Field exercise	Formalize Component D; count one week of first-pass delivery. Specification in full in Part D.
The opening	Artifact logged → Module 10 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The question delivery has been dodging

Your gauge can already answer two of its three questions. Component A says how much of what you committed stands finished and verified. Component E says whether your books tell the truth. This module formalizes the middle question, the one work actually flows through: **of everything you handed over, how much landed clean the first time?** That is Component D (first-pass delivery yield) and when it takes its place, your gauge stops being a production rate with footnotes and becomes the gauge whole: the rate, all three components in.

2. “Landed clean,” defined in one sentence

A handover lands clean when three things are all true: it was **handed over** (the recipient actually received it: built is not delivered, as Module 1's escapes taught), it was **on time** by the commitment it was owed under, and **no error was found afterward** within the counting window your rule states. Anything else is an escape, and escapes are classed, the way this series has classed them from the first module:

Escape class	What happened
Late	Handed over, but after the time it was owed.
Unlanded	Finished but never handed over, or surfaced only when the recipient asked.
Error found afterward	Handed over on time, and wrong: found by re-verification, by the recipient, or by the next step that trusted it.

Component D = handovers that landed clean ÷ all handovers in the window,
one decimal, under a counting rule written before the count.

One precision about the name, printed here so no reader is ambushed by another text. In Lean Six Sigma usage, first-pass yield counts units completed without defect, scrap, or rework; lateness is not a defect in that usage. This book's D is a **delivery yield**: under the doctrine's Article 3, a handover fails it for lateness and for non-delivery as well as for error found afterward, which makes D closer kin to a perfect-order rate than to textbook first-pass yield. The name is kept because the counting question is the same (did it land clean the first time?), and the difference is printed here, on the module's face, rather than left for a reader to discover against another book.

First-pass is the discipline in the name. A handover that landed wrong and was then repaired does not become clean retroactively. The repair is good work, recorded as such (Module 8), and the week's D still counts the escape. The moment repaired work counts as clean, D stops measuring delivery and starts measuring your willingness to forget, and every downstream reader of your gauge (starting with you) is reading a novel.

3. The four keys, translated

The source practice behind this module is the research program of **DORA (DevOps Research and Assessment)**: a decade of published, cross-industry study of software delivery performance, graded RECORD in this course's study (Forsgren, Humble and Kim, Accelerate, 2018, and the program's published reports). DORA tracks delivery through four keys, named here under their published names: **deployment frequency, lead time for changes, change**

failure rate, and **time to restore service**. Translated off the software floor, for any operation, the four keys ask:

1. **Deployment frequency.** How often do handovers actually happen, in small, regular landings, or in rare, giant ones? (Module 12 will show why the giant ones hide their failures so well.)
2. **Lead time for changes.** From “this is needed” to “this has landed”: the recipient's experienced wait, not your internal effort.
3. **Change failure rate** (failure share). Of the handovers, how many needed repair, the direct cousin of D itself: failure share is what D is not.
4. **Time to restore service.** When a landing fails, how long until it is right? A fast, honest recovery is a real capability, and it never upgrades the failed landing to clean.

You do not need to instrument all four this week. You need to know they exist, because they are the diagnostic set you will reach for when D moves and you want to know why, and because reading D beside them, honestly, is the next section's subject.

4. Reading D against the world: honestly

DORA publishes performance levels: ranges, per key, that its research associates with different levels of software delivery capability. Read the limit of that apparatus before reaching for it: **the levels are per key, for software delivery, and no published level measures a percentage like D.** There is therefore no band in this module to seat D inside. What an external frame can legitimately do for D is narrower, and still valuable. First, a sanity check on your counting rule: DORA's published failure rates for software delivery run in ranges that should make a counted D near perfection argue with itself. If your D sits far above every comparable published figure, the likeliest explanation is not that your operation is superhuman; it is that your counting rule is kind: your window sloppy, your “afterward” too short, your late class quietly redefined. Second, a diagnostic: compute, from the same register that produced D, your own failure share and your own recovery time, and read them beside D. Those two numbers are yours, counted under your rule, and they are the honest external frame this module offers.

And the claims discipline, stated as this module's objective requires, in writing, every time you make the comparison: **no external comparison entitles you to claim a ranking.** Your sentences are allowed to say: “our failure share this week, counted from the same register as D, is printed beside it, with our recovery times.” They are not allowed to seat D in a band, call your operation elite, certified, or ranked, or borrow a level's name for a number the level does not measure. No table with your name on it exists, and this course does not print league tables that do not exist. The method, the sources, and your own number: that is the whole advertisement, as the doctrine's claims line has said from the first module.

5. The gauge, whole for the first time

With D formalized, assemble the page the way the capstone will seal it in Module 14:

$$\text{RATE} = \text{A} \times \text{D} \times \text{E}$$

Production yield $\times$ first-pass delivery yield $\times$ reporting conformance. Multiplied, never averaged. Until this module the label was production rate; from this week, counted whole, it is the rate.

Read the three components the way Module 1 taught: the rate is the product of the three yields; each component names the stage where the losses live; the claim bars stand beside the rate saying what may be claimed. Your gauge is now complete on paper. Everything after this module (changeovers, batches, plans, reviews, and the capstone itself) is about raising the number and governing it, not about finishing the instrument. You built the instrument. The remaining five modules teach you to make it move.

PART B

The Worked Example

Continuing the teaching example: the billing team's first full week counted under a formal Component D rule (unit and counting rules as in Modules 1-4; delivery practices from Modules 7-8 now standing).

The week, counted

Line	Count	Figure
Handovers in the week	46	
Landed clean the first time	39	
Escapes: late	3	
Escapes: unlanded (surfaced only when asked)	1	
Escapes: error found afterward	3	
Component D = $39 \div 46$		84.8%

The gauge, read whole

Component	Value	From
A: production yield	87.3%	$986 \div 1,129$ countable (Module 1)
D: first-pass delivery yield	84.8%	$39 \div 46$ (this week)
E: reporting conformance	95.0%	$114 \div 120$ (Module 4)
RATE = $\text{A} \times \text{D} \times \text{E}$	70.3%	$0.873 \times 0.848 \times 0.950$

How to read 70.3%. Averaged, the same three components would read about 89.0%, a comfortable number that describes no week this team ever worked. Multiplied, 70.3% is the gauge's composite reading: the product of the three yields, each counted on its own population in its own window (A on the quarter's inventory, D on this week's handovers, E on the month's records). It is not the share of any one cohort that passed all three stages, because no cohort here was tracked through all three, and the page does not claim it is. What the product buys is compounding discipline: a weak stage taxes the whole number and cannot be averaged away. The losses live where the components say they live: delivery, this quarter, is the leakiest stage, and its 7 escapes are classed and owned. For the external frame, the team's page prints D beside the two numbers the same register honestly offers: its failure share, 7 of 46, 15.2%, and its recovery times for the week's escapes; against DORA's published levels it claims nothing, because no published level measures a percentage like D. That restraint, and no stronger sentence, is what the evidence buys. The Module 1 worked example's own house count, D of 94.4% (51 of 54 clean on its census day), is the same instrument on a different operation: same rule, same classes, numbers that mean the same thing. That sameness is the point of a gauge.

PART D

The Field Exercise: Formalize Component D: Count One Week Whole

Your artifact: Component D, formalized, one week of first-pass delivery, counted, logged in the app as your Component D page. **Time:** 30–60 minutes across the week. Module 10 opens, at your own pace, once this page is logged. Logged, not graded.

The page you are building

1. Your **“landed clean” sentence**, one sentence: on time, handed over, no error found afterward within a stated window.
2. The **counting rule**, written before the count: what counts as a handover, what “clean” means, when the count closes, who counts.
3. The **week's count**: every handover, marked clean or escape, with escapes classed: late · unlanded · error found afterward.
4. **Component D**: $\text{clean} \div \text{handed over}$, one decimal.
5. Your **gauge printed whole**: A, D, and E on one page, and $\text{RATE} = A \times D \times E$ computed and labelled as the rate.
6. The **frame sentences**: one sentence reading your failure share and recovery time from the same register beside D, and one sentence on what an external comparison does not prove.

The five steps

1. **Write the rule first (≈15 min).** Definition, window, classes, counter. Before the week's first handover is scored: a rule written after the count is an alibi (Module 1).
2. **Count the week (≈10 min a day).** Every handover, classed as it lands. Do not wait for Friday's memory; escapes are shy and Friday forgets them.
3. **Divide (≈5 min).** D to one decimal, exactly as counted, repairs recorded beside it, never inside it.
4. **Assemble the gauge (≈10 min).** Print A, D, E and the product on one page. Check the label: from this week it reads the rate, not production rate.
5. **Write the frame sentences, date, sign, log (≈10 min).**

Worksheet: the D page

[illegible]

Pitfalls:

- **Repaired counts as clean:** the classic. The repair is honoured on its own line; the first pass stays failed. First-pass means first.
- **The shrinking window:** “no error found afterward” with afterward quietly meaning “by close of business.” State the window in the rule and hold it all week.
- **Handovers that never enter the count:** the informal deliveries, the quick sends. If a recipient got it, it was a handover. Count it or the denominator is a story.
- **Borrowed bands:** seating D inside a published band that does not measure it, or reading a sanity frame as a title. The frame sentence and the not-proven sentence travel together, or neither travels.

Done looks like this: a rule written before its count, a full week honestly classed, and a three-component gauge on one page, reading whole. Log it. Next, Module 10 (The Ten-Minute Turn) starts the Speed stage: the changeovers that eat your week, and how to move their cost out of the stopped time.

Check yourself

Q. Name DORA's four keys under their published names.

A. Deployment frequency, lead time for changes, change failure rate, and time to restore service. This module's plainer renderings of them are translations, labelled as such.

Q. Is D a textbook first-pass yield?

A. Not exactly, and the module prints the difference: textbook first-pass yield counts units completed without defect, scrap, or rework; this book's D also fails a handover for lateness and non-delivery under the doctrine's Article 3, which makes it kin to a perfect-order rate. The name is kept; the difference is on the page.

Q. May you seat your D inside a published DORA performance band?

A. No. DORA publishes levels per key for software delivery; no published band measures a percentage like D. The honest external frame is your own failure share and recovery time from the same register.

THE OPENING: MODULE 9

Artifact logged → Module 10 opens at your own pace. The opening is the logged Component D page: the rule, the week's classed count, D to one decimal, the assembled gauge, and the two frame sentences. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: first-pass delivery yield (the house's delivery yield, defined on its face in the lesson) and DORA's four keys under their published names (deployment frequency, lead time for changes, change failure rate, time to

restore service; DORA/Accelerate research), translated for any operation. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 9 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

The Ten-Minute Turn

Make the changeover cheap, and small runs become possible

This module's artifact: one changeover, mapped, with one step moved out of the stopped window and the new changeover time measured, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** the changeover-cost line, a lever that raises A without working harder. (Capstone arc stage: Speed.)

LESSON PLAN: MODULE 10

The one concept	SMED, single-minute exchange of die: separate the changeover steps that must happen while work is stopped from those that can happen while it runs, and move the second kind out.
Objectives	By the end of this module you can: (1) map one changeover you repeat, step by step, with the stopped time stated; (2) classify each step as must-be-stopped or can-run-alongside, and defend the classification; (3) move one step out of the stopped window and re-measure the changeover; (4) state your changeover cost as a line on your gauge's supporting page.
Source story	Shigeo Shingo's press changeovers RECORD · Southwest Airlines' 1972 ten-minute turn RECORD .
Builds in your gauge	The changeover-cost line, a lever that raises A without working harder. (Capstone arc stage: Speed.)
Worked example	The billing team's intake changeover: the map as drawn, the step moved, and the two honest times.
Field exercise	Map one changeover you repeat; move one step out of the stopped window. Specification in full in Part D.
The opening	Artifact logged → Module 11 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The work between the work

Your gauge is complete on paper (Module 9). The remaining modules raise it, and the first lever is not effort. It is the time your operation spends not doing the work while it gets ready to do different work: switching tasks, tools, cases, customers, contexts. That in-between time is called a **changeover**, and in most operations it has never been mapped, let alone managed. It is simply where the hours went.

Every changeover carries a hidden tax beyond its minutes. While changeovers are expensive, rational people batch: they save up similar work and run it in long stretches to “avoid switching.” Long batches mean slow signal (Module 12), long waits (Module 3), and errors discovered after a whole batch carries them. Make the changeover cheap, and the whole shape of the work changes: small runs become possible, signal arrives early, and Component A rises without anyone working harder. That is why this module opens the Speed stage.

2. Shingo and the press

The method is SMED (single-minute exchange of die), developed by Shigeo Shingo inside Toyota's world, where changing the dies on a stamping press once took hours of stopped production. The “single-minute” in the name is the ambition: a changeover measured in single digits of minutes. Shingo's insight was not “work faster.” It was a distinction, and it has names in his method: **internal setup**, work that can only be done while the machine is stopped, and **external setup**, work that can be done while it runs. SMED's first move is to separate the two honestly; its second is to convert internal work to external wherever conversion is possible. Some steps of a changeover can only happen while the machine is stopped: the die itself comes out and goes in. But many steps were being done during the stop merely by habit: fetching tools, preparing the next materials, filling in the paperwork, adjusting and re-adjusting. Those steps belong outside the stopped window: done before, while the press still runs the old job, or after, while it already runs the new one. Classify honestly, move everything movable, and the stop shrinks to its irreducible core: hours became minutes in the documented cases, not because hands moved faster, but because the stop finally contained only what a stop truly requires.

3. The ten-minute turn

The same arithmetic flies, and its cause is the lesson. In 1972, Southwest Airlines, then a small Texas carrier fighting for survival, was forced to sell one of its four aircraft to make its payments, and chose to fly its four-airplane schedule with three. The only arithmetic that closed was the **ten-minute turn**: the airplane at the gate, emptied, cleaned, fuelled, boarded, and gone in about ten minutes, because an airplane earns nothing sitting at a gate. Everything that could be prepared before the aircraft arrived was prepared. Everything that could wait until after departure waited. The stopped window (the plane on the ground) held only what truly needed the plane. Your operation's “airplane” is whatever stops while you switch: the line, the desk, the case, the clinic room. The question is identical: what is actually in your stopped window, and how much of it never needed to be?

4. The classification: the whole method in two questions

Map the changeover step by step, every step, in order, timed. Then ask of each step two questions:

1. **Must this happen while the work is stopped?** If the honest answer is no, it is a candidate to move: before the stop (prepare it while the previous work still runs) or after it (finish it while the new work already runs).
2. **If it must stay, can it shrink?** The steps that remain inside get the second pass: staged tools, standard positions, the adjustment replaced by a preset. A step that must happen stopped can still stop costing its old time.

Defend each classification in writing on the map. The pressure you will feel, step by step, is the pressure to call things movable that are not (which corrupts the new time) and its mirror: to call things fixed that are merely habitual, which is where the savings live. Habit is the classification to interrogate hardest. “We’ve always done the count during the switch” is a history, not a necessity.

5. The changeover-cost line

The result lands on your gauge's supporting page as the **changeover-cost line**: the changeover's name, its stopped time as first measured, its stopped time after the move, and the date. It is a supporting line, not a component (it does not enter $RATE = A \times D \times E$) because its job is different: it is the lever record. When A rises in a later month, the supporting lines are where you look to see why, and a changeover that fell from 46 stopped minutes to 28 is a why with a date and two honest times on it. Keep both times on the line forever. An operation that records only the “after” time has hidden its own improvement from itself, and improvement you cannot show is improvement you cannot defend, repeat, or teach.

6. One move, honestly measured

This module asks for exactly one step moved, the same one-small-practice doctrine as Article 6. One move, re-measured, teaches the classification better than a redesign programme, and it leaves the operation a working map for the second and third moves it will now make on its own. Measure the new changeover on a real run, not in rehearsal: the time that counts is the time under load, interruptions and all. If the moved step creeps back into the stopped window within a month (it happens), the creep is recorded on the line too. The line is a record, not a trophy.

PART B

The Worked Example

Continuing the teaching example: the billing team's most repeated changeover, switching the intake desk from one customer's batch run to the next.

The map as drawn

Step	Minutes (stopped)	Class
Close out the previous batch: totals, tray count, sign-off	9	Must-be-stopped: the batch is not closed until it is counted
Fetch the next batch's source documents from the file room	11	Can-run-alongside: the documents exist before the switch
Re-set the intake screen, templates, and checklists for the new batch	8	Can-run-alongside, in part: presets can be staged in advance
Verify the new batch's intake count against its manifest	12	Must-be-stopped: the count is of this batch, at this desk
First-piece check and release to processing	6	Must-be-stopped
Total stopped time, as first measured	46	

The move, and the second time

The team moved the document fetch out of the window: the next batch's documents are now pulled and staged at the desk during the previous batch's run. One step, moved, nothing else changed. Measured on the next real changeover: the stopped window came to **35 minutes**: the 46 as first measured, minus the 11 fetch minutes, and nothing else claimed.

The changeover-cost line, as it stands on the team's page: "Intake changeover, stopped time 46 min (first measured, this month) → 35 min (after moving the document fetch out; a fall of 11 min, 23.9%). Next candidate: the screen-and-template reset." Read the line's discipline: both times printed, the move named, the percentage computed on the two printed times ($11 \div 46 = 23.9\%$), and the next candidate already queued. That is a lever being worked, not a victory being framed.

PART D

The Field Exercise: Map One Changeover: Move One Step Out

Your artifact: one changeover, mapped, with one step moved out of the stopped window and the new changeover time measured, logged in the app. **Time:** about 50 minutes plus one live run. Module 11 opens once this is logged, at your own pace. Logged, not graded.

The artifact carries five things

1. The **map**: one repeated changeover, every step in order, the stopped portion timed honestly.
2. The **classification**: each step marked must-be-stopped or can-run-alongside, with the classification defensible in one line each.
3. The **move**: one step of the second kind genuinely relocated: prepared in advance, or finished afterward.
4. The **two times**: before, and after, measured on a real run.
5. The **changeover-cost line** for your gauge's supporting page, carrying both times and the date. Your name closes the page.

The four steps

1. **Map and time (≈ 20 min).** Take the next real changeover and write it as it happens, steps in order, stopped minutes beside each. A map drawn from memory at a desk is a story; draw it at the work.
2. **Classify (≈ 10 min).** The two questions from Part A, step by step. Interrogate habit hardest.
3. **Move one step (≈ 10 min to arrange).** The clearest can-run-alongside step. Arrange its new home (before or after) so the next changeover can actually run the new way.
4. **Re-measure on a real run, date, sign, log (≈ 10 min).** The new stopped time, under load. Write both times on the cost line.

Worksheet: the changeover map

Step (in order)	Stopped minutes	Class (stopped / alongside)	If alongside: new home (before / after)
Total stopped: before ____ min · after ____ min · step moved:			

Pitfalls:

- **The desk-drawn map:** steps as they should happen, times as they should take. Map the real changeover, at the work, watch in hand.
- **Calling the immovable “movable”:** moving a step on paper that in fact still stops the work, so the new time is fiction. If the work waits for it, it is stopped time, wherever it is performed.
- **The rehearsal time:** measuring the new changeover on a quiet practice run. Measure under load, or the line is advertising.
- **Ten moves at once:** redesigning the whole changeover in one week and learning nothing about which move paid. One step, measured. Then the next.

Done looks like this: a step map a colleague could follow, one step genuinely relocated, and two honest times. Log it. Next, Module 11 (One Plan, Four Gates) gives each of your efforts a single plan of record, and four gates at which its progress is ruled, not vibes.

Check yourself

Q. In Shingo's terms, what are the two classes of setup work?

A. Internal setup, which can only be done while the work is stopped, and external setup, which can be done while it runs. SMED separates them honestly, then converts internal work to external wherever conversion is possible.

Q. A step is classified can-run-alongside, but the work still waits for it in practice. What is it?

A. Stopped time, wherever it is performed. If the work waits for it, it belongs inside the window and inside the honest measure.

Q. Why must the changeover-cost line keep both times forever?

A. Because the line is the lever record: a later rise in A is explained from it. An operation that records only the “after” time has hidden its own improvement from itself.

THE OPENING: MODULE 10

Artifact logged → Module 11 opens at your own pace. The opening is the logged artifact: the timed map, the classification, the one moved step, and both measured times on the changeover-cost line. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: SMED changeover method (Shigeo Shingo's published work) and the documented 1972 Southwest Airlines ten-minute turn. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 10 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

One Plan, Four Gates

One plan of record for each effort, reviewed at gates, worked to a standard

This module's artifact: one effort opened under a single plan of record with four gates, the plan and its first gate ruling logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** The plan-of-record and gate discipline that supports the Speed stage. (Capstone arc stage: Speed.)

LESSON PLAN: MODULE 11

The one concept	One build with three named threads: one plan of record per effort · progress ruled at four gates · recurring tasks worked to written standard work. The threads are named here because the module carries all three openly; the build is the plan.
Objectives	By the end of this module you can: (1) open one effort under a single plan of record that names owner, end state, and date (Article 1); (2) define four gates for that effort, each with the evidence that passes it; (3) state the standard work for the effort's recurring tasks: the current best way, written down; (4) hold the first gate conference on the plan and record its ruling.
Source story	ThedaCare Collaborative Care RECORD · Toyota Production System standard work RECORD .
Builds in your gauge	The plan-of-record and gate discipline that supports the Speed stage. (Capstone arc stage: Speed.)
Worked example	The billing team's queue-hygiene effort under one plan: its four gates as defined, and the first gate conference's ruling as recorded.
Field exercise	Open one effort with a single plan of record and four gates. Specification in full in Part D.
The opening	Artifact logged → Module 12 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The effort with five plans

Ask a struggling effort a simple question (show me the plan) and watch what arrives: a slide deck from the kickoff, a spreadsheet someone maintains privately, a thread of decisions half-remembered, and the version in the lead's head, which is the only current one and the only one nobody else can see. The effort does not have five plans. It has none, because a plan that exists in several competing versions cannot be conferred on, cannot be ruled against, and cannot be wrong in any way anyone is obliged to fix. This module installs the alternative, in three parts: **one plan of record, four gates** at which progress is ruled on evidence, and **standard work** for the tasks inside it.

2. The plan of record

A plan of record is the single document an effort answers to. It opens the way Article 1 of the doctrine requires every effort to open: a **named owner**, a **written end state**, a **date**, and it reports in the one uniform form of Article 2: **Status · Owner · Next milestone · Gate · Metric**. Other documents may exist (workings, notes, analyses), but the plan of record is named as the record, and where an older plan disagrees with it, the artifact says what happens to the older plan: superseded, by name, on a date. The point is not paperwork. The point is that from this week, anyone asking where does this effort stand? is pointed to one page, and the page is the same page for everyone, the level test of Article 7, applied to a single effort: the why and the where, in the same words, at every level the effort touches.

3. Four gates, ruled on evidence

A gate is a point in the effort's life where progress is ruled, not reported. Four is the house pattern: enough to catch drift early, few enough that each ruling matters. Name them for your effort's real life; the pattern underneath is constant:

Gate	The question it rules	Passed on evidence of
Gate 1: Opened right	Is this effort real: owner, end state, date, and a first milestone that can actually be checked?	The plan of record itself, complete in the uniform form.
Gate 2: Working as planned	Is the work moving under the plan: commitments kept (Module 2), work in process inside its cap (Module 3)?	The commitment line and flow fields for the effort, as counted.
Gate 3: Landing clean	Is what the effort hands over passing its checks: read-backs, independent verification (Modules 7–8)?	Verification records, with returns standing in them unsoftened.
Gate 4: Closed true	Did it end where the end state said, and do the books and the record agree (Module 4)?	The end state checked line by line; the exception queue empty or its remainder named.

Each gate is passed **only on its stated evidence**: that is what makes it a gate and not a meeting. A gate conference that “feels good about progress” and waves the effort through has held a meeting and called it a ruling; the next gate will inherit the unexamined trouble, with interest. And a gate can rule three ways: **pass**, **pass with named remainder** (the gap stated, owned, and dated), or **hold**: the effort does not proceed past this gate until the evidence exists. A hold is not a punishment. It is the gate doing the job it was built for, and it is recorded with the same respect as a pass.

4. The case conference: the whole team, one plan

The source practice for the conference half of this module is ThedaCare's Collaborative Care model, graded RECORD in this course's study: the patient's whole team (the people whose work touches the case) conferring on one plan for that patient, at the point of care, instead of each discipline carrying its own partial plan in its own notes. Transferred to any operation, the discipline is the same: the gate conference is held **on the plan of record**, with **everyone the effort touches present or represented**, and it rules **on the evidence the gate names**. One plan, all hands, evidence first. Most gate failures are not bad rulings; they are conferences held on three different versions of where the effort stands. The plan of record is what makes one shared version exist to rule on.

5. Standard work: the current best way, written

Inside the effort, its recurring tasks get the third piece: **standard work**: the current best way to do the task, written down by the people who do it, in the order it is done. Toyota's discipline, and widely misunderstood: standard work is not a rule handed down to lock a task in place forever. It is the baseline for improvement: the agreed way, today, against which a better way can be tried, measured, and adopted in the open (Article 6). Without a written current way, every person does the task their own way, variation is invisible, and an improvement discovered by one person on a Tuesday dies with their shift. With it, the task can be taught, the variation can be seen, and the better way (when someone finds one) has something definite to replace. Write it plainly, keep it where the task happens, and treat every proposed change to it the way this series treats formula changes: in the open, dated, with the old way kept on the record.

6. What this builds

On your gauge, this module's build is structural rather than a new number: efforts sized and gated so signal arrives early. A Gate 2 ruling in week two is signal; a surprise at the deadline is archaeology. It is also the module that quietly assembles the cast of the capstone: an operation whose efforts each carry one plan, ruled at gates, worked to written standards, is an operation whose records (Module 14) have something coherent to remember.

PART B

The Worked Example

Continuing the teaching example: the billing team opens its queue-hygiene effort (the work Module 1's count identified ("we're behind on queue hygiene; different problem")) under a single plan of record.

The plan, in the uniform form

Field	As written
Status	Opened this week; intake misroute tray being re-sorted under the new rule.
Owner	Team lead (named on the plan).
End state	Misroutes at intake fall to under 2 per week and stay there for four consecutive weeks; duplicates blocked at intake by the Module 4 validation.
Next milestone	Gate 1 conference this Friday.
Gate	Gate 1: Opened right.
Metric	Misroutes received per week (counted at intake); current week: 9.

The four gates as defined, and the first ruling

The team's gates follow the pattern in Part A, with Gate 2's evidence set as misroutes per week falling two weeks running, with the work-in-process cap (12) held, Gate 3's as the re-sort's output passing independent verification (Module 8) with returns recorded, and Gate 4's as four consecutive weeks under 2 misroutes per week, with the exception queue's hygiene items cleared or named.

Gate 1 ruling, as recorded: "PASS WITH NAMED REMAINDER. Plan is complete in the uniform form; owner, end state, and date stand. Remainder: the duplicate count at intake is not yet being recorded separately from misroutes, so Gate 2's evidence would mix two measures. Owner: team lead. Fix dated: separate intake tallies begin Monday, before any Gate 2 evidence is taken." Read the ruling's shape: the gate neither waved the effort through nor stopped it dead. It passed it with the one gap that would have corrupted the next gate's evidence named, owned, and dated. That middle ruling is the one real operations need most, and the one that only exists when gates rule on stated evidence.

PART D

The Field Exercise: One Effort, One Plan, Four Gates

Your artifact: one effort opened under a single plan of record with four gates, the plan and its first gate ruling logged in the app. **Time:** about 65 minutes, the conference included. Module 12 opens once this is logged, at your own pace. Logged, not graded.

The artifact carries five things

1. The **plan of record**: owner, end state, date, and the uniform form's fields: Status · Owner · Next milestone · Gate · Metric. Where older plans for the same effort exist, the artifact names this one as the record and says what happens to the others.
2. **Four gates** across the effort's life, each passable only on stated evidence.
3. The **standard work** for one recurring task inside the effort, the current best way, written down.
4. The **first gate ruling**, as recorded at a conference held on the plan with everyone the effort touches.
5. Your name and the date.

The five steps

1. **Choose the effort and write its plan (≈20 min)**. One live effort. If it already has plans, this step includes the honest sentence about them: superseded, by name.
2. **Define the four gates (≈15 min)**. Questions and evidence, following Part A's pattern, in your effort's own terms. If you cannot state a gate's evidence, the gate is a meeting. Rewrite it until you can.
3. **Write one piece of standard work (≈10 min)**. One recurring task, the current best way, in order, written with the people who do it.
4. **Confer and rule (≈15 min)**. Everyone the effort touches, one plan on the table, evidence first. Record the ruling (pass, pass with named remainder, or hold) with owners and dates on any remainder.
5. **Date, sign, log (≈5 min)**.

Conference of one. Working alone, the gate conference is a written self-ruling: the plan on the table, the gate's evidence listed, the ruling written in the record's words (pass, pass with named remainder, or hold) with the date. It is weaker than a room, the other eyes are the mechanism, and the ruling is labelled self-ruled so nobody mistakes the form. The gates still rule: a solo hold binds exactly as a room's hold binds.

Worksheet: the plan of record

Field	Yours
Status · Owner · Next milestone · Gate · Metric	
End state · date	
Gate 1: question · evidence	
Gate 2: question · evidence	
Gate 3: question · evidence	
Gate 4: question · evidence	
First ruling · remainder · owner · date	

Pitfalls:

- **Five plans, politely:** declaring a plan of record while the old versions keep circulating and being used. The superseding sentence is part of the artifact; without it, you have six plans.
- **Gates without evidence:** “Gate 2: progress review.” A gate that names no evidence rules on mood, and mood always passes.
- **The ruling nobody wrote down:** a conference that agreed warmly and recorded nothing. The record is the ruling; an unrecorded ruling will be re-litigated at every subsequent gate.
- **Standard work as scripture:** a written way that may never change teaches people to hide their better ways. It is a baseline for improvement, changed in the open, not a monument.

Done looks like this: one plan everyone can point to, four gates with evidence attached, and a first ruling on record. Log it. Next, Module 12 (Small Batches, Fast Signal) splits the work itself: ship less at once, and learn sooner whether it works.

Check yourself

- Q.** Your effort has a kickoff deck, a private spreadsheet, and the lead's memory. How many plans does it have?
- A.** None. A plan in several competing versions cannot be conferred on or ruled against. The plan of record is one named document, and older plans are superseded by name, on a date.
- Q.** A gate conference “feels good about progress.” Has the gate ruled?
- A.** No. A gate passes only on its stated evidence. Feeling good is a meeting; the ruling (pass, pass with named remainder, or hold) is recorded against the evidence the gate names.

Q. Working alone, what form does the gate conference take, and how is it labelled?

A. A written self-ruling on the plan, evidence listed, ruling recorded with its date, labelled self-ruled. The gates still rule: a solo hold binds exactly as a room's hold binds.

THE OPENING: MODULE 11

Artifact logged → Module 12 opens at your own pace. The opening is the logged artifact: the single plan of record, four evidence-bearing gates, one piece of standard work, and the first gate ruling as recorded. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: case-conference practice as published by ThedaCare (Collaborative Care) and standard work as practised in the Toyota Production System, both documented public methods. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 11 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Small Batches, Fast Signal

Ship less at once, and learn sooner whether it works

This module's artifact: one batch, split, with the signal from its first portion read and recorded before the rest is built, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** the batch-size line, the second Speed lever: smaller batches, faster correction, higher A. (Capstone arc stage: Speed.)

LESSON PLAN: MODULE 12

The one concept	The small-batch rule: work in batches small enough that the signal from the first piece arrives before the rest is built.
Objectives	By the end of this module you can: (1) name one batch you currently ship whole and state its size; (2) split it so a first portion can ship and be judged on its own; (3) define, in advance, the signal you will read from the first portion and what each possible reading will make you do; (4) record the signal as read, including when it says stop.
Source story	Zara's design-to-store cycle RECORD .
Builds in your gauge	The batch-size line, the second Speed lever: smaller batches, faster correction, higher A. (Capstone arc stage: Speed.)
Worked example	The billing team's monthly statement run: the split as made, the signal as read, and the stop it honestly produced.
Field exercise	Split one batch you ship; read the signal before building the rest. Specification in full in Part D.
The opening	Artifact logged → Module 13 opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The batch that ships its mistakes wholesale

A batch is any amount of work that travels together: the month's statements in one run, the report written whole before anyone sees a page, the release built for six weeks before a user touches it. Batches feel efficient, one setup, one send, one review. Their cost is invisible until you price the alternative: **everything in the batch shares the batch's fate, including its mistakes.** Find an error in a run of sixty and you have not found one error. You have found an error with sixty chances to have shipped, and no way to know which it took. Find it in the first five of a split batch, and the other fifty-five were never built wrong at all.

2. Zara: the signal is the product

The fashion retailer Zara built its business on the opposite instinct. Where the industry designed whole seasons months ahead and shipped them entire (a giant batch, judged only when the season's receipts were in), Zara's documented model runs design-to-store in short cycles and small quantities: produce a limited run, read what actually sells in the first days, and let that signal decide what gets made in quantity, restocked, or dropped. The lesson generalizes far past clothing. Zara is not fast because its people hurry. It is fast because **its batches are small enough that the truth about them arrives while there is still time to act on it**, and most of a big batch's apparent efficiency is the cost of learning everything too late.

3. The rule, stated plainly

The small-batch rule: work in batches small enough that the signal from the first portion arrives (and is read) before the rest is built.

Three words in the rule do the heavy lifting. Signal: a defined observation, not a vibe, what you will look at, in the first portion, that tells you whether the whole is sound. Arrives: the first portion must actually reach the place the signal lives (the recipient, the check, the real environment), not merely exist. Read: someone named looks, at a time stated in advance, and a decision follows. A first portion shipped and never examined is not a small batch; it is a big batch with a head start.

4. Define the signal before the first piece ships

This is the module's discipline, and it is Module 1's counting-rule discipline wearing work clothes: **write the signal down before the first portion ships. What you will look at, when, and what each reading will make you do: proceed, adjust, or stop.** Written before, because a signal defined after the reading is whatever the reading needed it to be. The three-way decision is written too, because the commonest failure of small batches is not a bad split; it is a signal that said adjust and a team that proceeded anyway, the batch's momentum doing the deciding. Pre-committing the decision is what gives the signal authority. It is the andon cord (Module 5), pulled by a document you wrote while calm.

5. Choosing the first portion

The first portion is the smallest piece that can be judged on its own: representative of the whole, and complete enough to meet the real test. Two failure modes flank it. Too easy: a first portion cherry-picked from the batch's simplest work produces a green signal that certifies nothing; the signal's honesty requires the portion to carry the batch's real difficulties in proportion. Too grand: a first portion that is half the batch has surrendered half the benefit before it ships. The working test: if this portion fails, will we be glad we built only this much? If yes, it is sized right.

Notice what makes any of this affordable: Module 10. Small batches multiply changeovers (every split adds switches) and an operation whose changeovers are expensive will rationally refuse to split. The Speed stage is sequenced for exactly this reason: cheapen the turn first, then split the batch, and the two levers together produce the flow that neither produces alone.

6. The batch-size line

On your gauge's supporting page, beside the changeover-cost line, add the **batch-size line**: the batch's name, its old size, its new first-portion size, and the signal's reading as recorded. Like the changeover line, it is a lever record, not a component: it does not enter the formula. Its job is to let a future rise in Component A be explained: smaller first portions, errors caught after 15 pieces instead of 60, corrections that cost an afternoon instead of a recall. And when a signal says stop, the stop is printed on the line with the same pride of place as a proceed. In this system, a faithful stop is a success the gauge knows how to honour. It is several dozen unshipped errors, caught at the price of fifteen pieces.

PART B

The Worked Example

Continuing the teaching example: the billing team's monthly statement run (60 statements, historically produced whole and sent in one batch at month's end) split for the first time.

The split and the pre-written signal

The team split the run into a first portion of **15 statements** and a remainder of 45, with the signal written before the first portion was produced: "Read the first portion's verification results (Module 8 standard) and its first-pass count at the mid-month check. Proceed if the portion is clean or its errors are one-offs with named causes. Adjust if the same error appears twice. Stop and fix the source before building the remainder if the same error appears three times or any privacy-class error appears at all."

The signal as read

Line	Count	Figure
First portion produced and verified	15	
Errors found in the first portion	2	
First-portion error share = $2 \div 15$		13.3%
The two errors	Same class: statements matched by name-similarity at one manual step, the trap from Module 6's register	

The decision, as recorded: “Signal reads ADJUST, by the rule written before the portion shipped: same error twice. The name-matching step is removed and the account-number match (Module 4 validation) applied to the remainder before it is built. Remainder of 45 produced under the corrected step; verified clean, 0 errors of the class.” Now price the alternative honestly: under the old whole-batch run, the same error rate would have put about 8 statements wrong in a single send, discovered, if at all, after they had landed. The split did not make the team more careful. It made the error arrive when it was still small, in front of a rule that had already decided what to do about it.

One line of institutional memory belongs on this page. The name-matching trap was first caught by the register in Module 6, and the fix was applied then to the statement send. It has reappeared here, in the monthly run, because the fix was applied to one procedure and never carried to its sibling: name that plainly, a **relapse**. The register caught the Module 6 fix failing in practice; the fix is now re-applied at the shared step, and the relapse is logged on the batch-size line, so the pattern (a fix applied narrowly) has a name the next review can use. Registers remember; people relapse; the register's job is to make the relapse visible the week it happens, not the quarter after.

The batch-size line as it stands on the team's page: “Monthly statements: batch 60 whole → first portion 15 + remainder 45; first signal: adjust (2 same-class errors, corrected in the remainder; remainder clean).”

PART D

The Field Exercise: Split One Batch: Read the Signal First

Your artifact: one batch, split, with the signal from its first portion read and recorded before the rest is built, logged in the app. **Time:** about 55 minutes plus the run itself. Module 13 opens once this is logged, at your own pace. Logged, not graded.

The artifact carries five things

1. The **batch**, named, with its current size stated.
2. The **split**: the smallest first portion that can stand judgement on its own, and the remainder it precedes.
3. The **signal**, written before the first portion ships: what you will look at, when, and what each reading decides: proceed, adjust, or stop.
4. The **reading**: the signal as actually read at the stated time, recorded exactly.
5. The **decision** the reading produced, including a stop, faithfully recorded, which earns this artifact in full.

The four steps

1. **Split (~15 min)**. Choose the first portion: representative, not cherry-picked; small, not half the batch.
2. **Write the signal (~15 min)**. Before anything ships. Observation, time, and the three-way decision, in ink.
3. **Ship the first portion and read at the stated time (~15 min plus the run)**. The portion goes to its real test: recipient, check, real environment. Read what is there, not what was hoped.
4. **Record, date, sign, log (~10 min)**. The reading and the decision as they happened. If the decision overruled the pre-written rule, that is recorded too, with its reason. The rule can be argued with in the open; it cannot be quietly disobeyed.

Worksheet: the split and the signal

Field	Yours
Batch · old size	
First portion · remainder	
Signal: look at · when · proceed if · adjust if · stop if	
Reading as read	
Decision that followed	

Pitfalls:

- **The cherry-picked portion:** the batch's easiest work sent first to guarantee a green reading. A signal that cannot fail is not a signal.
- **The signal written afterward:** the reading arrives, then the “rule” is composed to bless it. Written before, or it does not count.
- **Proceeding through an “adjust”:** the batch's momentum overriding its own pre-commitment. If the rule is wrong, change it in the open for the next batch: after honouring or honestly recording the overruling of this one.
- **Splitting without Module 10:** a split that multiplies expensive changeovers until the team re-merges the batch in self-defence. Cheapen the turn, then split.

Done looks like this: a batch that no longer travels whole, a signal defined before it could be wished for, and a decision that followed the reading. Log it. Next, Module 13 (The Weekly Review) gives your gauge its steering rhythm: review by exception, and one controllable input to pull for each component.

Check yourself

Q. What three words in the small-batch rule do the heavy lifting?

A. Signal (a defined observation), arrives (the first portion reaches the real test), and read (a named person looks at a stated time, and a decision follows).

Q. The signal reads “adjust” and the team proceeds anyway, quietly. What has happened, and what must the record show?

A. The batch's momentum overruled its own pre-commitment. An overruling may be argued in the open, but it is recorded with its reason; the rule cannot be quietly disobeyed.

Q. The Module 6 register's fix was applied to the statement send, and the same trap fires in the monthly run. What is that called, and where is it logged?

A. A relapse: a fix applied narrowly, failing in practice at its sibling step. It is logged on the batch-size line, so the pattern has a name the next review can use.

THE OPENING: MODULE 12

Artifact logged → Module 13 opens at your own pace. The opening is the logged artifact: the split, the pre-written signal, the reading as read, and the decision that followed. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: small-batch working and fast signal as documented in Zara's published design-to-store model, translated for any operation. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 12 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

The Weekly Review

Review by exception, and steer by the inputs you control

This module's artifact: your input-metric page, one controllable input per component, with one weekly review run and its rulings recorded, logged in the app. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** one controllable input metric named per component · the weekly review rhythm. (Capstone arc stage: Governance.)

LESSON PLAN: MODULE 13

The one concept	Exception-driven review; input metrics: review what departed from standard, and steer by one controllable input per component, not by the outcome alone.
Objectives	By the end of this module you can: (1) name one controllable input metric for each gauge component (A, D, and E) that your operation can move this week; (2) distinguish each input from the outcome it steers, in writing; (3) run one weekly review by exception: the departures from standard, in order, with rulings recorded; (4) set the review's standing rhythm, day, hour, attendees, and the page it reads from.
Source story	Amazon's Weekly Business Review, as described in Working Backwards (Bryar & Carr, 2021) INFERENCE : a participant account, carried at that strength.
Builds in your gauge	One controllable input metric named per component · the weekly review rhythm. (Capstone arc stage: Governance.)
Worked example	The billing team's input-metric page, and its first exception review: the exceptions as listed, the rulings as recorded.
Field exercise	Name one controllable input per gauge component; run one review. Specification in full in Part D.
The opening	Artifact logged → Module 14, the capstone, opens at your own pace. Logged, not graded.

PART A

The Lesson

1. The meeting that reads everything and rules on nothing

Most operations hold a weekly meeting. It reads the whole dashboard aloud, in order, at length. Green items get their minutes like red ones. By the time the genuinely broken thing arrives on page nine, the room is tired, the hour is nearly gone, and the ruling (if any) is “keep an eye on it.” Nothing about that meeting is a review. It is a recital. This module replaces it with the discipline your whole gauge has been building toward: **review by exception**: the meeting reads only what departed from standard, in the order the gauge page lists it, and it rules.

2. The source: a review built on inputs

The standing example is Amazon's Weekly Business Review, carried in this course's study at the grade it deserves: INFERENCE, a participant account, described from inside by people who ran it (Bryar and Carr, Working Backwards), read at that strength and not as audited fact. The practice is real; the account is a participant's, and the grade keeps those two sentences separate. Its two disciplines are the ones this module adopts. First, the review is driven by a standard metrics package, read in a fixed order, week after week: the same page, so departures are visible against memory. Second, and more important: the review steers by **input metrics** (the numbers the operation directly controls) rather than arguing with output numbers it can only wish at. A sales figure, an outcome, can be lamented but not pulled. The inputs underneath it (what was offered, how fast, to whom) are levers with handles. Your gauge needs the same separation, and it needs it in writing.

3. Inputs steer; outcomes report

Every gauge component is an outcome: A, D, and E report what your operation did. You cannot move an outcome directly this week: you can only move the things that produce it, and read the outcome next week. An **input metric** is a number your operation controls this week, chosen because it steers one component. The discipline is one input per component, stated beside the outcome it steers:

Component (outcome)	The shape of a good input	What makes it an input, not a hope
A: production yield	Something counted before finishing happens: open work held at or under the cap; commitments stated binary, on time.	The team controls it today: starting less and stating promises well are decisions, not results.
D: first-pass delivery	Something at the delivery point: read-backs run on every irreversible send; verifications completed before claim.	Running the check is this week's act; the clean landing is next week's consequence.
E: reporting conformance	Something in the books' upkeep: entries reconciled on the day they occur; exception queue reviewed and worked weekly.	Reconciling daily is a rhythm a person can keep or break by Friday, and can be counted by Friday.

Write the distinction down for each of yours, in the exercise's words: this input, we control; that outcome, it steers. If a proposed input cannot be moved by anything the team does this week (if it is a market condition, another department's mood, a hope wearing a number), it is not an input. Strike it and find the lever with a handle.

4. Review by exception: the order of business

The review itself runs on a fixed, short liturgy:

1. **The page.** One page: your gauge page as assembled in Module 9, with its supporting lines (Modules 10 and 12) and its candor lines (Modules 5 and 6). The review reads from the page, never from memory or from a fresh deck built for the meeting.
2. **Exceptions only, in the page's order.** A departure from standard: a component moved the wrong way, a gate held (Module 11), a queue ageing (Module 4), a catch or near-miss line gone silent (Modules 5–6), an input missed. What conformed is not discussed. It has already said everything it has to say by conforming.
3. **Every exception gets a ruling.** Not a discussion (a ruling: what will be done, by whom, by when) or the explicit ruling that it is accepted, with the reason stated. “Noted” is not a ruling; it is how exceptions learn to come back bigger.
4. **The rulings are recorded on the page.** Next week's review opens by reading last week's rulings before it takes new exceptions. A ruling that was not kept is the first exception of the new week: the review's own PPC, in effect, and it is counted with the same honesty as every other line in this series.

5. The standing rhythm

The review exists only if its rhythm is fixed in advance: **the same day, the same hour, the same attendees, the same page**, written down as part of this module's artifact. A review scheduled “when things calm down” is scheduled for never, and an operation's things do not calm down; they are calmed, weekly, by exactly this meeting. Keep it short by keeping it exceptional: a review that rules on five departures in thirty minutes serves the operation better than a two-hour recital that rules on none. And protect the page's candor the way every earlier module protected its own line: the review that punishes the bearer of a bad number will, within a month, be reviewing numbers that have learned to behave, while the operation underneath them has not.

PART B

The Worked Example

Continuing the teaching example: the billing team's input-metric page, and the rulings from its first review by exception.

The input-metric page

Component	Input (controlled this week)	Outcome it steers
A	Open efforts at or under the cap of 12, counted every Friday	Production yield: work finishes faster when less is open (Modules 3, 1)
D	Read-backs run on 100% of statement sends; no send proceeds without one	First-pass delivery yield (Modules 7, 9)
E	Entries reconciled the day they occur; exception queue worked every Friday	Reporting conformance (Module 4)

The first review, as recorded

Exceptions taken in the gauge page's order, **6 in total**: the work-in-process count at 14 against the cap of 12 (A's input missed) · one send run in a rush with its read-back skipped and caught after the fact (D's input missed; the send itself verified clean on re-check, and the skip, not the outcome, is the exception) · three exception-queue items aged past two weeks (E, counted item by item: 3) · the near-miss register silent for the week (Module 6's silence finding). **Rulings recorded: 6 of 6**: each exception ruled: an action with an owner and a date, or accepted with its reason stated.

Two rulings worth reading closely. On the skipped read-back, the review ruled on the input, not the lucky outcome: "The send verified clean. The exception stands anyway: the input was missed, and inputs are what we steer. Ruling: sends are held at the tray until the read-back is logged, the tray, not memory, enforces it." And on the silent near-miss week: "Ruled as a reporting gap, not a clean week: the lead files the week's first almost personally at Friday's count, in front of the team." That is exception review working as designed: the page said what departed, the room ruled, and the rulings were written where next week's room will have to answer for them.

PART D

The Field Exercise: Name the Inputs: Run the Review

Your artifact: your input-metric page, one controllable input per component, with one weekly review run and its rulings recorded, logged in the app. **Time:** about 60 minutes, the review included. Module 14, the capstone, opens once this is logged, at your own pace. Logged, not graded.

The artifact carries four things

1. The **input-metric page**: one controllable input for each of A, D, and E, each written beside the outcome it steers, each distinguished from that outcome in writing.

2. The **review record**: one weekly review run by exception: the departures from standard, in the gauge page's order, with rulings recorded.
3. The **standing rhythm**, fixed in writing: day, hour, attendees, and the page the review reads from.
4. Your name and the date.

The four steps

1. **Name the inputs (≈20 min)**. One per component. Apply the handle test to each: can this team move this number this week, by its own acts? If not, it is not an input: keep looking until you find the lever that is.
2. **Convene the review (≈30 min)**. The gauge page on the table. Exceptions only, in the page's order. Rule on each: action, owner, date, or accepted, with the reason.
3. **Record the rulings (≈5 min)**. On the page, where next week's review will open by reading them.
4. **Fix the rhythm, date, sign, log (≈5 min)**. Day, hour, attendees, page, in writing, standing, until changed in the open.

Working alone. The review of one is still a review: the same page, the same order, the rulings written, the rhythm kept. What a room gives you is challenge; the solo substitute is time: write the exceptions, then rule on them after a night's sleep, in the record's words. Label the record self-reviewed; the label keeps the weaker form honest, and the rulings bind exactly as a room's do.

Worksheet: inputs and rulings

Field	Yours
Input for A · outcome it steers	
Input for D · outcome it steers	
Input for E · outcome it steers	
Review: exceptions in page order · ruling on each	
Standing rhythm: day · hour · attendees · page	

Pitfalls:

- **Reviewing the whole page:** reciting conforming items until the exceptions get the tired end of the hour. Exception-only is not rudeness; it is how the broken thing gets the room's best thirty minutes.
- **Inputs that are outcomes in disguise:** "our input is a higher D." That is the outcome asking to steer itself. Inputs are acts you can count by Friday.
- **Rulings of "noted":** an exception acknowledged and unowned will return, larger, with seniority. Rule, or accept with a reason on the record.
- **A rhythm of "when we can":** the review that floats is the review that sinks. Fix the hour; let other things move around it.

Done looks like this: three inputs you can actually pull, one review that ruled on exceptions only, and a rhythm that will still be there next week. Log it. Next, the capstone (Module 14, Registers That Remember) assembles the whole gauge, places its formula under change control, and reads your delta against the baseline you set in Module 1.

Check yourself

Q. "Our input this week is a higher D." What is wrong with the sentence?

A. D is the outcome; it cannot steer itself. An input is an act the team controls this week and can count by Friday, like read-backs run on 100% of sends.

Q. At what grade does this book carry Amazon's Weekly Business Review?

A. INFERENCE: a participant account, described from inside by people who ran it (Bryar and Carr, Working Backwards), read at that strength and not as audited fact.

Q. The first review finds six exceptions. How many rulings must be recorded?

A. Six: each exception ruled, an action with an owner and a date, or accepted with its reason stated. "Noted" is not a ruling.

THE OPENING: MODULE 13

Artifact logged → Module 14, the capstone, opens at your own pace. The opening is the logged artifact: the input-metric page, one recorded exception review, and the standing rhythm in writing. Logged, not graded. Modules already opened stay open.

What this module teaches, shows, and never shows

Teaches: exception-driven review and input metrics, as described in published participant accounts of Amazon's Weekly Business Review (Bryar & Carr, Working Backwards), graded INFERENCE as a participant account, on the plan's face. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 13 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Registers That Remember

Capstone: place the gauge under change control, assemble the page, read the delta

This module's artifact: your sealed gauge page, the assembled page, formula printed, delta read, logged in the app. This page is the course's completion instrument. **Time:** lesson about 20 minutes of study reading · audio about 19–24 minutes · field exercise as stated in Part D (design figures from the course format specification; the series has not yet run, and these are design figures, not measured results). **Gauge piece built:** the capstone itself: the full gauge page assembled · the formula placed under change control · the delta read against the M1 baseline · the gauge page issued. (Capstone arc stage: Capstone.)

LESSON PLAN: MODULE 14

The one concept	Change control on records of record: the register remembers; changes are amendments, printed and dated, never edits that rewrite the past.
Objectives	By the end of this module you can: (1) state the change-control rule for your gauge: who may amend the formula or a counting rule, how, and with what printed on the page; (2) assemble your full gauge page from the components and lines built across Modules 1–13, with nothing recomputed under a new rule; (3) read your delta against your Module 1 baseline and state it exactly as counted; (4) issue your gauge page as your completion instrument, dated, with its claim bars standing beside its rate.
Source story	Configuration management, the defense discipline of baselines and amendments (ANSI/EIA-649) RECORD .
Builds in your gauge	The capstone itself: the full gauge page assembled · the formula placed under change control · the delta read against the M1 baseline · the gauge page issued. (Capstone arc stage: Capstone.)
Worked example	The billing team's assembled gauge page at the capstone: the final count, the full formula read, and the delta against its Module 1 baseline, claim bars beside the rate.
Field exercise: the capstone	Seal the gauge: assemble the page, print the formula, read the delta. Specification in full in Part D.
The opening	Artifact logged → the course is complete. The logged capstone artifact is the assembled gauge page itself.

The Lesson

1. The register that forgot

Every long-running record faces the same quiet temptation: the past, as written, becomes inconvenient. A counting rule turns out to have been unfair. A quarter's numbers would read better under the definition the team uses now. And the record (a spreadsheet, a register, a living document) is editable. So someone edits it: the old figures are recomputed, the old rule is replaced in place, and the register now says the past was always as the present prefers. Nothing was lied about, exactly. Everything was forgotten, precisely. This capstone module is the discipline that forbids that forgetting: **change control**: the register remembers; changes arrive as amendments, printed and dated; the past is never rewritten in place.

2. Configuration management: the source discipline

The defense world (which cannot afford a register that forgets what a system was, because people maintain, certify, and stake safety on those records) institutionalized this as **configuration management**: every item of record has an identified baseline; every change to that baseline is proposed, ruled on, recorded with its reason and its date, and applied forward; and the history of baselines is itself part of the record, retrievable forever. Graded RECORD in this course's study, and deliberately unglamorous: it is not a method for making better changes. It is a method for making changes that cannot pretend they were always so. Your gauge (fourteen modules of counts under one formula) has become exactly the kind of record this discipline exists to protect. Its value is its history. Change control is how the history survives contact with the future.

3. Your change-control rule, stated

The rule for your gauge has the same shape as every rule in this series, written, plain, and complete before it is needed:

1. **Who may amend.** A named role (the gauge's owner) and no one else. Amendments proposed by anyone are welcomed; amendments made by anyone but the owner do not exist.
2. **How.** In the open: the new formula or counting rule, the reason for the change, and the date, printed on the gauge's face. The amendment rides beside the number it will govern, where every future reader will meet it.
3. **What is kept.** The old series, entire. The page carries the break: readings before the amendment under the old rule, readings after under the new, and for one full cycle both numbers are reported, the old-rule figure standing beside the new until the new series has earned its own history (Module 1's promise, now given its machinery).
4. **Never backward.** No amendment is ever applied to a number already reported. A reported number is a fact of the record. It can be annotated, superseded going forward,

even formally retracted with its reason printed. It cannot be silently recomputed into a better past.

Notice the family resemblance, because it is the series' whole argument arriving at its last form. The counting rule written before the count (Module 1). The signal written before the batch ships (Module 12). The standard written before the verification (Module 8). In every case, the same move: **decide in the calm, in writing, and let the record (not the moment's comfort) govern what happens in the storm.** Change control is that move applied to the gauge itself.

4. Assembling the page

The capstone's assembly is deliberately undramatic, because drama here would mean invention. The page is assembled **from parts already built**, and nothing on it is recomputed under a new rule for the occasion:

On the page	Built in	Carried as
Component A	M1: raised by M10-M12	The current count, under the M1 rule
The commitment line	M2	Kept ÷ committed, with reason codes
The flow fields	M3	Work in process · aging · completions
Component E + exception queue	M4	Conformance under the stated rule; queue standing open beside it
The catch line · near-miss & alert counts	M5-M6	Leading indicators, beside the escapes they precede
Delivery practices + Component D	M7-M9	Read-back and verification standing; D counted first-pass
Changeover-cost & batch-size lines	M10 · M12	The levers, with their before-and-after figures
The plan of record · gates · standard work	M11	The effort's single plan, its four gates and their rulings, the standard work as written
Input metrics + review rhythm	M13	One controllable input per component; the standing review
The formula	M1, sealed here	$RATE = A \times D \times E$, with its adoption date and its change-control rule
The position line (E')	M1 (house gauge)	Printed on the house's page in Module 1 and carried where the operation counts a position; reported separately, never blended into E
The claim bars	M1, carried throughout	Beside the rate. Always beside the rate.

5. Reading the delta

The delta is a subtraction, honestly come by: **today's rate, on the assembled page, minus your Module 1 baseline**: both counted under the same formula and the same rules, because the formula was never redefined along the way. Where your baseline was a production rate (A alone, as Module 1 required it to be labelled), the page says so, and reads the delta component by component as well as whole: A then and A now, beside the first full rate your gauge could honestly print. And the page dates its components: a rate assembled from counts taken in different weeks is an assembly, labelled with each component's count and window, not a photograph of today. State the delta exactly as counted. A small delta, honestly come by, is a complete success of this course: it means the instrument is real, and a real instrument that reads a small improvement this quarter reads a large one the quarter it happens. A large delta produced by a moved formula is worth nothing, and everyone who has followed this series (starting with you) knows exactly why.

6. The page is the instrument: the course's last word on credentials

This course issues no belts and no certificates of attendance, and this module issues the one thing it promised instead: **your gauge page** (assembled, sealed under change control, dated, its claim bars standing beside its rate) is your completion instrument. It does not say you sat through fourteen modules. It says an operation measured itself honestly across fourteen modules, built an instrument piece by piece, never once moved the formula to flatter itself, and its number moved by this much. Show it to anyone. Every figure on it can be recomputed from the record behind it, and the record remembers. That is more than a certificate has ever been able to hand over, and it is, in the end, the whole method: one honest number, built to be believed, in an operation that can say why at every level. Keep counting. The register remembers: now make sure what it remembers is worth the keeping.

PART B

The Worked Example

Continuing the teaching example to its close (the constructed teaching example, as labelled in Module 1): the billing team's assembled capstone page. Baseline from Module 1: Component A of 87.3% (986 verified of a countable inventory of 1,129), labelled at the time as a production rate, A alone, under the formula adopted that week and never since amended.

The final count

Component	Count	Value
A: production yield, recounted under the M1 rule	936 verified of 1,040 countable	90.0%
D: first-pass delivery yield	39 clean of 46 in the counted week (M9)	84.8%
E: reporting conformance	114 reconciled of 120 (M4); exception queue standing beside it	95.0%
RATE = A × D × E	0.900 × 0.848 × 0.950	72.5%
Claim bars, beside the rate	At the full standard, complete and cleared	No claim bar was ever counted for the team; no unit on its page stands counted at the full standard, and the page prints the absence rather than a borrowed figure

The provenance of this page, printed because the capstone is where provenance matters most. The components were counted in different weeks and the page dates them: A recounted this week under the Module 1 rule, on the inventory as it stands today (1,040 countable); D from the Module 9 week (39 clean of 46); E from the Module 4 month (114 of 120). The rate is an assembly of those counts, not a single day's photograph. A's denominator moved between the two censuses: 1,129 countable in Module 1, 1,040 now. The inventory is a live queue and it turned over in between; the printed record does not decompose the 89-unit difference, and the A delta must not be read as pure improvement. The page also carries no position line (E'): no position was ever counted for the team in any module, and the absence is printed here as a fact, not filled with the house gauge's line.

The delta, read exactly

Against the Module 9 first full reading (A 87.3% · D 84.8% · E 95.0% · rate 70.3%): the rate stands at **72.5%**: a delta of **+2.2 points**, arrived entirely through Component A, which moved 87.3% → 90.0% (**+2.7 points**) under its original counting rule, on the strength of the Speed stage's levers: the intake changeover down from 46 to 35 stopped minutes (M10) and the statement run split so its errors are caught in the first 15 (M12). D and E stand where they were first counted, printed unchanged, because unchanged is what they are. The formula's face carries its adoption date from Module 1 and, from today, its change-control rule. No figure on this page was recomputed under a new rule for the occasion. One limit rides with the headline, printed here rather than footnoted: A's denominator moved between the baseline census and this one (1,129 to 1,040) and the record does not decompose the movement, so the +2.7 is read as a ratio's movement with that fact attached, not as pure like-for-like gain.

Read the delta the way the course has taught you to read everything: 2.2 points is not a transformation story, and the page does not dress it as one. It is a small, real movement, in the component the work actually touched, under a formula that never moved, which is precisely the kind of number that can be built on, planned against, and shown to another person without a footnote of apologies. The team's gauge page, issued this week, is its completion instrument. It will still be counting next week.

PART D

The Field Exercise: The Capstone: Assemble, Seal, Read the Delta

Your artifact: your sealed gauge page, the assembled page, formula printed, delta read, logged in the app. This page is the course's completion instrument. **Time:** 30–60 minutes of assembly, on records you have been keeping for thirteen weeks. The course completes when this page is logged.

The capstone carries five acts

1. **Assemble the page** from parts already built: Component A (M1), the commitment line (M2), the flow fields (M3), Component E and its exception queue (M4), the catch, near-miss, and alert lines (M5–M6), the delivery practices and Component D (M7–M9), the changeover and batch-size lines (M10–M12), and the input metrics and review rhythm (M13). Nothing on the page is recomputed under a new rule for the occasion.
2. **Print the formula** on the page's face ($\text{RATE} = A \times D \times E$) with the date it was adopted in Module 1.

3. **Place the formula and every counting rule under change control**, in writing: amendment only in the open (new rule, reason, and date printed), the old series kept, no amendment ever applied backward.
4. **Read the delta**: today's rate against your Module 1 baseline, stated exactly, with the claim bars standing beside the rate, and the baseline's own label (production rate, A alone) stated beside it where that is what the baseline was.
5. **Date and sign the page; log it**. The logged page is issued as your completion instrument.

Worksheet: the assembled page, in outline

Line	Yours
A · D · E · RATE (as counted, rules as adopted)	
Supporting lines: commitment · flow · catch · near-miss/alerts · changeover · batch · inputs	
Claim bars, beside the rate	
Formula · adoption date · change-control rule	
Baseline (M1, as labelled then) · today's rate · delta, exactly	

Pitfalls (the capstone's own):

- **The occasion recount**: recounting a component “a little more carefully” for the final page, under what amounts to a new rule. The page assembles the record; it does not improve it. If a recount is owed, it is owed as next cycle's work, under an amendment printed in the open.
- **The delta dressed up**: comparing today's full rate against the kindest available baseline, or averaging where the formula multiplies. The delta reads against the M1 baseline as labelled then, component by component where the labels require it.
- **Change control with a back door**: “corrections” exempted from the amendment rule. A correction that changes a reported number is an amendment, or it is a retraction with its reason printed. There is no third door.
- **The page as a trophy**: framed, finished, and never counted again. The page is an instrument, issued at the capstone and read at every weekly review thereafter. Completion is the start of the gauge's working life, not its retirement.

Done looks like this: one page that says what your operation's number is, how it is computed, what may not yet be claimed, and how far the number has travelled since the week you first counted it, under a formula that never moved. Log it. The course is complete; the counting is not. That was always the arrangement.

Check yourself

Q. Your capstone rate reads higher than your Module 1 baseline, and A's denominator moved between the two censuses. What must the page print?

A. The movement itself: both denominators, the dates of both counts, and the limit that the record does not decompose the difference, so the delta is not read as pure improvement.

Q. Who may amend the formula, and how?

A. Only in the open: the amendment, the reason, and the date printed on the gauge's face, the old series kept, and no amendment ever applied backward to raise a number already reported.

Q. The team in the worked example carries no claim bar and no position line. Is its page incomplete?

A. No, provided the page says so: no claim was counted at the full standard for the team, and no position was counted for it, and the page prints the absence as a fact rather than borrowing the house gauge's lines.

THE OPENING: MODULE 14

Artifact logged → the course is complete. The logged capstone artifact is the assembled gauge page itself: components and lines as built, the formula printed with its adoption date and placed under change control, the delta read exactly against the Module 1 baseline, and the claim bars standing beside the rate. That page is the completion instrument. The course issues no other. Logged, not graded.

What this module teaches, shows, and never shows

Teaches: change control and configuration management as documented in defense practice, applied to the subscriber's own gauge. **Shows:** worked examples at the WHAT level: what was produced, and what it counted. **Never:** unpublished detail, any principal's private information, or the internal machinery behind the examples.

Cost of this module: external cash spent to build this taught edition, **\$0.00** · compute **UNMEASURED** (metered in aggregate only, so no per-module figure exists, and none is claimed). Printed because a course that teaches honest counting counts itself.

Raise the Rate · Module 14 of 14 · Taught Edition · October 2026 · Team Tess Playbook. Sources for this module are cited in the Sources and Bibliography at the back of Raise the Rate: The Complete Lesson Plan Book (Final Edition, October 2026). Evidence grades: RECORD, a named, checkable source · INFERENCE, a reasoned transfer, labelled as such · FOLKLORE, widely repeated, no primary source, labelled on its face.

Your Gauge, Assembled

The whole instrument on one page, what fourteen modules build

The three components

Comp.	Question it answers	Built in	Definition
A	PRODUCE: of everything committed, how much stands finished and verified?	M1; raised by M10-M12	Verified units ÷ countable inventory units (inventory minus items excluded under the rule, exclusions printed), counted under a rule written before the count
D	DELIVER: of everything handed over, how much landed clean the first time?	Formalized in M9; defended by M7-M8	Clean first-pass handovers ÷ all handovers in the window
E	REPORT: do the books foot, and do they tell the truth about the position?	M4; defended by its exception queue	Reporting conformance under the operation's stated rule (Module 4's continuous close: records reconciled as work happens, exceptions standing named in the queue); position reported separately, never blended in

Supporting lines stand on the same page: the commitment line (kept ÷ committed, with reason codes), the flow fields (work in process, aging, completions), the in-process catch line, the near-miss and alert counts, the changeover-cost and batch-size lines, and one controllable input metric per component, reviewed weekly by exception.

$$\text{RATE} = \text{A} \times \text{D} \times \text{E}$$

Production yield × first-pass delivery yield × reporting conformance. The stages multiply; they never average. A defect in any stage taxes the whole number, in the open, and a zero anywhere is a zero everywhere.

Claim bars beside the rate, always

The rate says how you are doing; the claim bars say what you may claim. A claim bar counts units that have crossed a whole standard (complete and cleared) out of all the units there are. The bars print beside the rate, never blended into it. A bar at zero beside a climbing rate is not a contradiction; it is momentum and distance, both true, neither lying about the other.

The delta

Your completion reading is a subtraction, honestly come by: **today's rate, on the assembled page, minus your Module 1 baseline**: both counted under the same formula and the same rules, because the formula was never redefined along the way. That is why the delta can be believed: not because it is large, but because nothing between the two readings was allowed to move except the work.

What Happens Next

The map is in your hands; the taught editions stand ready, at your pace

The taught editions. Modules 2 to 14 each have a taught edition (the full lesson with its audio), included whole with the Team Tess Playbook subscription. They are members' material, worked at your own pace: a module's edition opens when you log the field-exercise artifact of the module before it, and opened modules stay open. **Module 1 has no taught edition:** Module 1 is printed complete in this book, lesson, worked example, and exercise, and there is nothing further to open. There is nothing further to buy to finish this course.

This book stays free. It is the whole map, and maps should be free: anyone considering the course can read every module, judge Module 1 complete, and know exactly what the fourteen modules ask and build, before you begin.

What may be said about this course: the claims discipline, in one line: the method, the sources, and your own number, nothing more is claimed, and nothing more needs to be.

Cost of this book. Printed here because a course that teaches honest counting counts itself.

External cash spent to build this book: **\$0.00**.

Compute: **UNMEASURED**: the house meters its compute in aggregate only, so no per-book compute figure exists, and none is claimed.

Raise the Rate: The Complete Lesson Plan Book · Final Edition · October 2026 · Team Tess Playbook. Sources: the published works cited in the Sources and Bibliography at the back of this book; every graded claim traces to a source named there or on its module's plan. The house's own figures are its own counts, printed as counted where they stand. Times for Modules 2–14 are design figures from the format specification and are labelled as such wherever they appear.

Sources and Bibliography

The published sources the graded stories rest on

Every RECORD grade in this book rests on a source named here. INFERENCE grades mark the house's reasoned transfers from these sources and are labelled where they stand. No source is cited for a claim it does not carry, and no citation in this list is decorative: each entry is a work the text actually relies on.

Module 1 • Your Number

Ballard, Glenn. The Last Planner System of Production Control. PhD dissertation, University of Birmingham, 2000. · Pyzdek, Thomas, and Paul A. Keller. The Six Sigma Handbook. McGraw-Hill, 4th edition, 2014 (rolled throughput yield and first-pass yield as terms of art, defined and distinguished in Module 1, section 3).

Module 2 • Commitments You Can Count

Ballard (2000), as above: percent plan complete, the reliability ranges, and the plateau.

Module 3 • Less Started, More Finished

Little, John D. C. "A Proof for the Queuing Formula: $L = \lambda W$." Operations Research 9, no. 3 (1961). · Liker, Jeffrey K. The Toyota Way. McGraw-Hill, 2004 (kanban and work-in-process limits).

Module 4 • Make the Error Impossible

The Joint Commission. National Patient Safety Goals (medication reconciliation), 2005 onward. · Payer first-pass yield is taught as a claims discipline; no published band for it is cited, and none is claimed anywhere in this book.

Module 5 • Stop the Line

Adler, Paul S. "Time-and-Motion Regained." Harvard Business Review 71, no. 1 (1993) (NUMMI). · Liker (2004), as above (andon).

Module 6 • Report the Almost

NASA. Aviation Safety Reporting System: program documentation and published reports, NASA Ames Research Center (asrs.arc.nasa.gov). · Kenney, Charles. Transforming Health Care: Virginia Mason Medical Center's Pursuit of the Perfect Patient Experience. Productivity Press, 2010 (the Patient Safety Alert system).

Module 7 • Read It Back

Gawande, Atul. The Checklist Manifesto. Metropolitan Books, 2009 (the Boeing Model 299 and the checklist's origin; crew resource management as the later development it is).

Module 8 • Verified by Another

NASA. Independent Verification and Validation Program: program documentation, NASA IV&V Facility, Fairmont, West Virginia.

Module 9 • First Pass, Every Pass

Forsgren, Nicole, Jez Humble, and Gene Kim. Accelerate: The Science of Lean Software and DevOps. IT Revolution Press, 2018; and the DORA research program's published reports (the four keys under their published names).

Module 10 • The Ten-Minute Turn

Shingo, Shigeo. A Revolution in Manufacturing: The SMED System. Productivity Press, 1985. · Gittell, Jody Hoffer. The Southwest Airlines Way. McGraw-Hill, 2003 (the ten-minute turn era).

Module 11 • One Plan, Four Gates

Toussaint, John S., and Roger A. Gerard. On the Mend: Revolutionizing Healthcare to Save Lives and Transform Your Organization. Lean Enterprise Institute, 2010 (ThedaCare's Collaborative Care). · Liker (2004), as above (standard work).

Module 12 • Small Batches, Fast Signal

Ferdows, Kasra, Michael A. Lewis, and Jose A. D. Machuca. "Rapid-Fire Fulfillment." Harvard Business Review 82, no. 11 (2004) (Zara).

Module 13 • The Weekly Review

Bryar, Colin, and Bill Carr. Working Backwards. St. Martin's Press, 2021 (the Weekly Business Review: a participant account, graded INFERENCE in this book).

Module 14 • Registers That Remember

ANSI/EIA-649. Configuration Management Standard. SAE International (the defense discipline of baselines and amendments).