[Engineering Question]

The Operator's Dictionary · https://kb.jeffreysummers.com/docs/engineering-question/

Definition

[Engineering Question] is the first question in the operation: was this area engineered to operate this way, under these circumstances, congruent with the overall design of the business? If it wasn't, every fix is [Repair Work] and the problem stays unsolved.

It is one question with two forms. At design, it asks whether the area is being engineered to operate this way, under these circumstances. At the break, it asks whether the area was engineered to operate this way, under these circumstances. The first form decides what gets built. The second form decides what gets done about what broke.

A design is fully realized only when it carries three things: the area, the standard the area runs to, and the means of imposing that standard. When an area breaks, the break came from one of two places. Either the design wasn't fully realized, or the standard that maintains the integrity of the design wasn't imposed. The break form finds which one by reverse engineering the break against the design.

It runs at every level of the operation: the station, the role, the shift, the line on the P&L, each of the Five Fundamentals, and the business as a whole. The question stays the same and only the level it's asked at changes.

Mechanism

It goes first. The question comes before any fix and before any design. Every other move the operator makes depends on its answer, so the answer has to exist before the move gets made. Asked second, it becomes an audit of money already spent.

The design form. Before an area exists, the operator asks whether it's being engineered to operate the way he needs it to, under the circumstances it will actually face, and whether it fits everything else he's building. This is where the box gets sized. If you can't afford a 2,500 square foot restaurant, you go smaller. If you can't afford fifteen servers on the stage, you need fewer

sections, which means a smaller footprint across the board. The design form ties every area to what the operation can actually carry before the lease is signed and the schedule is written.

The break form. After an area exists and something breaks, the operator asks whether the area was ever engineered to operate this way, under these circumstances. A break doesn't answer that question by itself. It comes from one of two places: the design, or the standard that holds the design together. Either the area was never fully engineered for what broke it, or it was and the standard wasn't imposed.

A fully realized design. Engineering an area isn't finished when the area is drawn. A fully realized design carries the area, the standard the area runs to, and the means of imposing that standard: who imposes it, the authority they hold to do it, the time in the schedule to do it, and the read that shows whether it's holding. A design that calls for a standard and leaves out the means to impose it isn't a fully realized design. The standard can't hold, and when it slips, that's a design break, not an integrity break.

The break form is a locator. The break form isn't a yes-or-no test. Its job is to find where the break came from, at what level, and on which branch. The operator finds it by reverse engineering the break. He works back from the break through the design, step by step, comparing what the design called for with what actually happened. The source is the first point where they don't match. A labor cost that breaks the business gets traced back through the labor model until the trace reaches the point where the model was never engineered to work through the inputs it now has to contend with, such as higher rates. The locator lands on the labor model, in Profit, and tells the operator what he's actually facing before he spends a dollar on it.

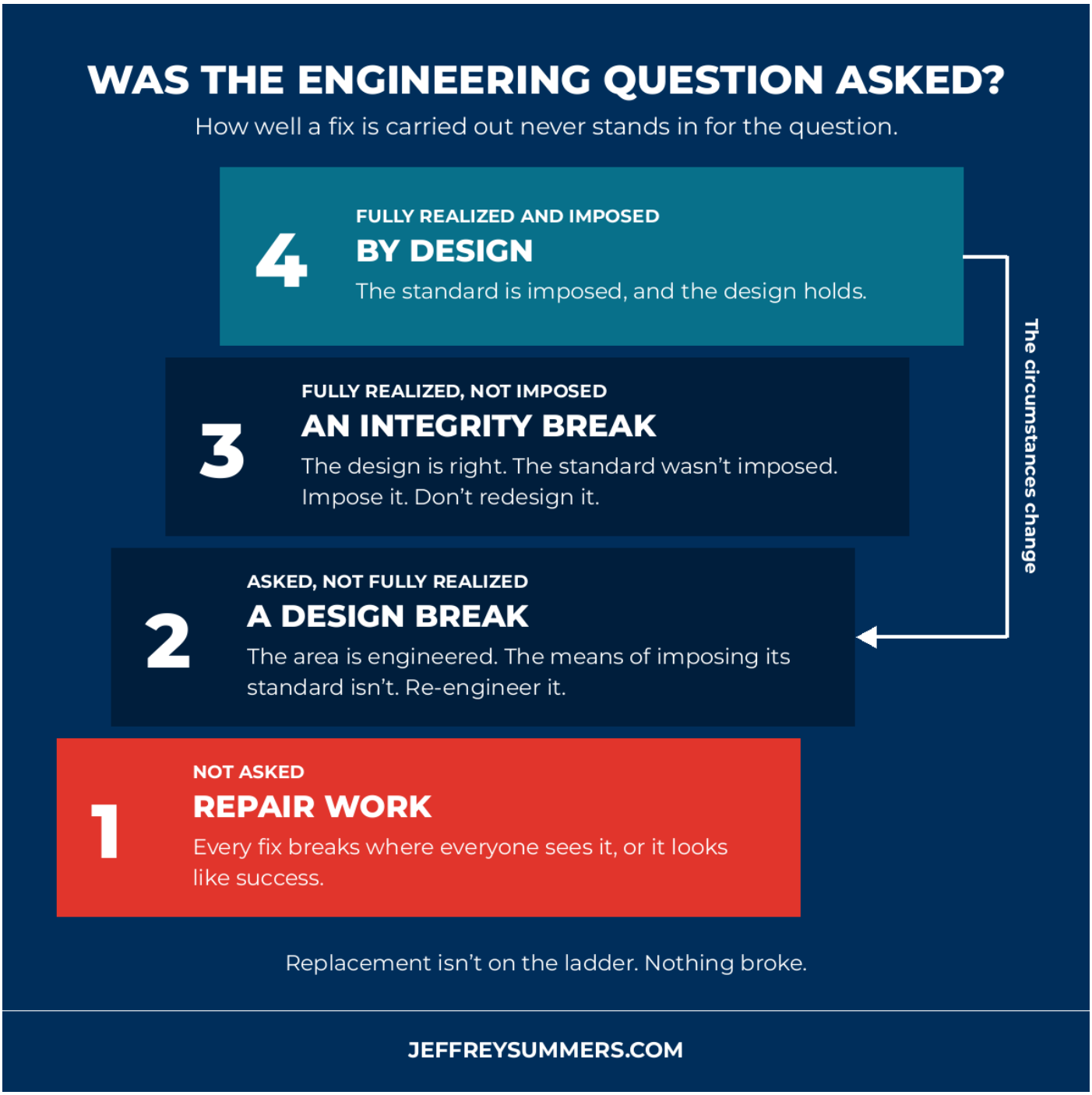
Trace the design, not the event. Reverse engineering the break only works if the trace runs back through the design. Run it back through what happened, and the trail ends at the last person who touched the break, and the operator is back to asking who failed. Run it back through the design, and the trace stops at the design every time, either where the design never called for what the area needed or where it called for it and the standard wasn't imposed.

Two branches. The trace lands on one of two branches. A design break is where the design never called for what the area needed at that step, or called for a standard with no means to impose it. The area wasn't fully engineered, and it gets re-engineered. An integrity break is where the design called for it, the means to impose the standard was there, and the standard wasn't imposed. Nothing gets re-engineered. The standard gets imposed. The tell between the two is whether the means of imposing the standard was actually used. If it was used as designed and the standard still slipped, the means doesn't hold under these circumstances, and it's a design break. If it

wasn't used, it's an integrity break.

What failed comes first, then who. After a break, the question is always what failed before it's ever who. On a design break the answer is a what: the design. On an integrity break the answer becomes a who, and the who is never the cast member standing where it broke. A fully realized design names who imposes the standard and gives them the means. The standard was supposed to be imposed on that cast member, not left up to them, so the trace runs past the cast member to whoever held the means and didn't use it. Both branches end with the operator. A design break is the operator who didn't fully realize the design. An integrity break is the operator, or the role he gave the means to, who didn't impose the standard. Restaurants don't fail. Operators do.

By Design doesn't stay By Design. A fully realized design was engineered for a set of circumstances: the rates the operation pays, the volume the book takes, the menu it runs, the cast it has. When those circumstances change, the design is no longer realized for the area it's running. Nothing about the design changed. What it has to contend with did. The area drops back onto the design branch, and the design form runs again before anything breaks, not after. That's [Constraint Inheritance] when the operator misses it, an area still engineered for circumstances that are gone. It's [No Static Achievement] in the design itself: an area that was By Design last year has to be engineered again for this year's circumstances to stay By Design.



Engineered to some degree is not engineered properly. Every area in a restaurant was engineered to some degree. Somebody drew the station, wrote the schedule, and set the price. The question isn't whether anyone engineered the area. It's whether the area was engineered properly, for this way of operating, under these circumstances. That's why the answer to a no is to re-engineer the area, not to engineer it from nothing.

Congruence is part of the question. No part of the operation can be engineered in isolation. Each Fundamental can't be designed to operate independently of the others, and each area inside a Fundamental can't be designed independently of the areas around it. An area that works on its own terms but doesn't fit the overall design of the business wasn't engineered properly, and it will

break where it meets the parts it was never designed to fit.

Every level has its own first cause. There's a first cause for labor breakage and a first cause for food cost breakage, and each one sits at the level where that break happened. The context of the whole business is a different level of the same question. At that level, the first cause is the operator mistaking the transaction for the thing that produces the outcome. The levels don't compete. Each one asks the question about the thing in front of it.

A fix in isolation is Repair Work. When the question was never asked, whatever gets done to the broken area is done to that area alone. The labor line gets reconciled against a percentage. The food cost line gets a tool. The fix touches the piece that broke and nothing it connects to, so the operation returns to the state it was already in, and the break comes back.

Replacement is not a break. Swapping a part the design expects to wear out is replacement. A bulb burns out, a gasket wears through, a pan warps. The design already accounted for it, so nothing broke and the question doesn't fire. Replacement and repair are two different things. Repair answers a break, and a break means the design didn't hold.

What comes after the trace. On a design break, the area is known to need redesign. Then [Question Dumb Shit] runs on everything the operation currently does in that area: does this activity help or hinder the redesign? If it hinders, stop doing it. Then the area gets re-engineered to fit the overall design, under the circumstances it actually faces. On an integrity break, nothing gets redesigned and nothing gets bought. The standard gets imposed by whoever owns imposing it, with the means the design already gave them.

The specimen. A labor-productivity model that works from the labor percentage instead of from what each shift needs to accomplish. It's attempting to reconcile labor costs without first asking whether the question was ever asked. Reverse engineer a schedule from the percentage and you have to ask where you cut. Which task doesn't get done because you won't schedule the hours for it: equipment maintenance, deep cleaning, four-walls marketing? The work the business needs done to grow can't be scheduled, so the operation shrinks. The higher labor cost is the [Lost Opportunity Tax] for not having engineered the restaurant properly from the beginning, and productivity management applied to that labor model is Repair Work on a box that was never sized to the crew it can afford.

Load-Bearing Distinction

Not [Question Dumb Shit]. The Engineering Question comes first. It finds out whether the area was ever engineered to operate this way, under these circumstances. Question Dumb Shit comes after, once the area is known to need redesign: does what we do now help or hinder the redesign? If it hinders, stop. The Engineering Question decides whether redesign is required. Question Dumb Shit clears the path for it.

Not [Repair Work]. Repair Work is what every fix becomes when the answer is no. It's the fix in isolation, made when neither the Engineering Question nor Question Dumb Shit was asked. The Engineering Question is the test that tells you which one you're doing before you spend the money.

Not [Hacksterism]. Hacksterism sells you one piece as the fix. A vendor sells a labor tool, another sells a food cost tool, and so on. The Engineering Question asks whether that piece was ever designed to fit the rest of the business. A piece engineered in isolation fails no matter who sold it.

Not fault-finding. Fault-finding starts at the who and stops at the first person it can name. The Engineering Question starts at the design. It gets to a who only on the integrity branch, only after the break has been reverse engineered, and only to the role that owned imposing the standard. The cast member where the break happened is where the trace starts, not where it ends.

Not replacement. Replacement swaps a part the design expects to wear out. Nothing broke, so the question doesn't fire. The operator who files a break under replacement keeps swapping parts in an area that was never engineered for what it faces. The operator who files replacement under breaks redesigns areas that were working.

The term is load-bearing because the industry works every cost, every role, and every complaint on its own, as if the piece that broke were the whole problem. Without the question named and asked first, the operator spends his money fixing parts of a design nobody checked, and calls the recurrence bad luck, bad people, or a bad market.

Diagnostic Tests

Test One — The First Question Test. Take the last thing that broke in the operation and write down the first question anyone asked about it. If the first question was who did it, what it costs, or what will fix it, the Engineering Question was never asked. If the first question was whether the area was engineered to operate this way, under these circumstances, it was.

Test Two — The Trace Test. For the same break, reverse engineer it back through the design, step by step. At each step, write down what the design called for and what actually happened. The source is the first step where they don't match. Name the area and the level where it sits: station, role, shift, line, Fundamental, or the business as a whole. If the trace ended at a person instead of a step in the design, it ran back through the event, not the design, and it has to be run again.

Test Three — The Full Realization Test. Take the area that broke and ask whether its design names the standard, who imposes it, the authority they hold to impose it, the time in the schedule to do it, and the read that shows whether it's holding. If any one of those is missing, the design wasn't fully realized, and the break is a design break.

Test Four — The Imposition Test. When the trace lands on a standard that slipped, ask whether the means of imposing it was used. If it was used as designed and the standard still slipped, the means doesn't hold under these circumstances, and it's a design break. If it wasn't used, it's an integrity break, and the who is the role that held the means, not the cast member where it broke.

Test Five — The Isolation Test. Look at the fix currently being applied to the break. List everything in the operation the fix touches besides the broken area. If the list is empty, the fix is being worked in isolation, and it's Repair Work regardless of how well it's executed.

Test Six — The Percentage Test. Ask how the target for the broken line was set. If the answer is an industry average, a historical number, or a percentage handed down from somewhere else, the line was set by reference, not engineered against what this operation is built to produce. Reconciling to it is Repair Work.

Test Seven — The Where Do I Cut Test. List the work the business needs done to grow, then mark which of it gets cut when the line hits its target. Equipment maintenance, deep cleaning, and four-walls marketing are the usual first casualties. Every task on that list that can't be scheduled is evidence the area was never engineered for the business it's supposed to grow.

Test Eight — The Congruence Test. Pick a Fundamental and ask what its design depends on in the other four. Does the schedule fund the standard? Does the review form measure what the standard says matters? Does the cost structure carry the GX/CX, or does the GX/CX get cut to carry the cost structure? Any answer that runs one Fundamental against another means that Fundamental was never designed to be congruent with the overall design.

Test Nine — The Replacement Test. Before calling something a break, ask whether the design expected the part to wear out and get swapped. If it did, it's replacement and the question doesn't

fire. If the design never expected the part to fail, it's a break, and the trace runs.

Test Ten — The Vendor Test. When a tool is pitched to fix a broken line, ask the seller whether the tool asks if the area was engineered to operate this way, under these circumstances, and whether it fits the rest of the business. If the tool works its line alone, it's selling one piece as the fix.

Test Eleven — The Design Form Test. Before building or changing any area, ask whether it's being engineered to operate this way, under the circumstances it will actually face, whether it fits the overall design, and whether it carries the means of imposing its own standard. If you can't afford the staffing the footprint requires, the footprint fails the test before the lease is signed.

Family Position

Top-level. Parent-level term that sits first in the sequence of operating the restaurant: the first thing to do before designing the restaurant, and the first thing to do when something breaks. It runs at every level, and at each level the answer sends the operator into design, into the redesign that [Question Dumb Shit] clears the path for, or, on an integrity break, into imposing the standard the design already carries. Cross-Fundamental. At the Fundamental level, the question asks whether each Fundamental was ever designed to be congruent with the overall design of the business, because no Fundamental can be designed to operate independently of the others.

Perspective read. When something breaks, the operator stops asking who failed and asks first whether the way he reads the operation was ever designed to be congruent with the overall design of the business, before he attempts a fix. The default read goes straight to fault. A break arrives, and the first question in the room is who did it, so the answer is always a person and the fix is always aimed at a person. The read itself is usually inherited. It came from a percentage, an average, or a habit of reading one line at a time, and it was never built against what this operation is designed to produce. The tell is a read that sees every break as a separate event. The response is to put the question ahead of the who, every time, so the read finds the design first. A who only enters on the integrity branch, after the break has been traced, and it's the role that owned imposing the standard.

Product read. When the GX/CX breaks, the operator asks whether the experience was ever designed to be congruent with the overall design of the business, before he attempts a fix. A broken experience gets read as an execution failure, so the response is retraining, a new script, or a tool. If the experience was designed beyond what the rest of the business was built to deliver, every one of those responses is Repair Work on the Product. Technology shows the same split.

Tech that supports the GX/CX is good. When it supplants the GX/CX, it's bad, and it usually gets installed as a fix for a break somewhere else, such as labor, while being described as support. The question asks whether the experience and everything delivering it were designed as one thing.

People read. When a role breaks, the operator asks whether the role was ever designed to be congruent with the overall design of the business, before he attempts a fix. A role breaks when the cast member in it can't hold it: the standard slips, the turnover repeats, the same position keeps getting rehired. The default response is to coach harder or replace the person, which treats the role as finished and the person as the variable. The question comes first. It asks whether the role was ever designed to carry what it's being asked to carry, with the authority the standard requires, inside the schedule and the cost structure the business actually runs. If it wasn't, rehiring the position to the same design returns the operation to the same break. If it was, and the standard the role runs to wasn't imposed, the trace doesn't stop at the cast member in the role. It stops at whoever owned imposing that standard on the role.

Performance read. When execution breaks on the stage, the operator asks whether the way the work gets produced was ever designed to be congruent with the overall design of the business, before he attempts a fix. Late tickets, a standard that doesn't hold on a Friday, a shift that falls behind and never recovers. The default fix is more bodies or more pressure, and every body you throw at a broken shift makes it that much more broken. The question asks whether production was engineered for the volume the book takes, the menu it runs, and the cast it has, and whether it fits the standard, the schedule, and the room it runs in. A production design that never fit those things breaks at the same point every time, whoever is on the line.

Profit read. When the money breaks, the operator asks whether the cost structure was ever designed to be congruent with the overall design of the business, before he attempts a fix. The default response reconciles the broken line against its percentage, so labor gets cut to hit the number and food cost gets a tool to hit its number, each one worked alone. A cost structure built from historical averages wasn't engineered to work through the inputs it now faces, such as higher rates, and it wasn't engineered to fund the work the business needs done to grow. The higher cost is the Lost Opportunity Tax, the excess over a properly engineered box that never gets counted. The response is to re-engineer the cost structure, and where necessary the box itself, so the operation can grow inside the cost constraints it actually holds.

Cross-References To Locked IP

Parent:

- [Five Fundamentals] — the structure the question runs across, each Fundamental asked whether it was designed to fit the other four

Related:

- [Question Dumb Shit] — the question that runs after the no, clearing every activity that hinders the redesign
- [Designed Architecture] — the architecture an operation has when its areas pass the question at design
- [Constraint Architecture] — finds the cap on output; the Engineering Question tells you whether you're repairing or redesigning around it
- [Constraint Inheritance] — a step built for a limitation the operation no longer runs, an area engineered for circumstances that are gone
- [No Static Achievement] — a fully realized design doesn't stay realized; when the circumstances change, the area has to be engineered again to stay By Design
- [Lost Opportunity Tax] — the cost of an area that was never engineered properly, paid in excess nobody counts
- [By Design Or By Default] — the choice-layer verdict; a part can be chosen on purpose and still fail the Engineering Question

Opposing patterns:

- [Repair Work] — the fix in isolation, what every fix becomes when the question was never asked
- [Repairman Syndrome] — the operator running an endless sequence of repairs with no pass on why the breaks keep coming
- [Hacksterism] — one piece sold as the fix, engineered in isolation from the rest of the business
- [Vendor Capture] — the pitch arriving before the diagnosis, so the tool defines the break instead of the question locating it
- [Default Architecture] — parts ruled from outside the building and never composed against each other, which the question finds

Why This Matters

The industry sells pieces. A vendor sells a labor tool. Another sells a food cost tool, another sells a scheduling platform, another sells a training program, and so on. Each one works its line alone,

and each one gets bought to fix a break in that line. The main reason none of them work is that the overall design of the business was never taken into consideration. You can't engineer each piece in isolation, and every tool built to fix one piece is built on the assumption that you can.

The operator who buys the piece is doing Repair Work, and he usually doesn't know it. The labor line comes down for a few periods, the work that grows the business stops getting scheduled, and the break comes back under a different name. He reads the recurrence as bad people, a bad market, or a bad tool, and goes looking for the next fix. The question that would have located the problem in the first place never gets asked, because nothing in the sale asks it.

That's why the question has a name. Named, it applies to every issue in the restaurant, at every level, with no exceptions. It doesn't replace any other idea or any other part of the thinking. It puts every idea in its place in the sequence of operating a restaurant. The question goes first, Question Dumb Shit clears the path, and the design gets built or rebuilt to fit the whole. Every other tool in the framework works better once this one has been asked, because every other tool assumes the area it's working on was engineered to be worked on.

Operating Consequence

Ask it first. Before any fix, any purchase, any retraining, any cut, the operator asks whether the area was engineered to operate this way, under these circumstances, congruent with the overall design of the business. Nothing else gets decided until it's answered.

Read every break as evidence. A break isn't an event to patch. It's proof that either the design or the standard didn't hold. The operator stops asking whether the area was built for this and starts reverse engineering the break against the design to find where it came from, at what level, and on which branch.

Ask what failed before who. The first question after a break changes. "Who did this" becomes "what was this area engineered to handle, and what is it handling now." A who only comes into it on the integrity branch, and it's the role that owned imposing the standard, never the cast member where it broke.

Design the imposition. Every design names its standard, who imposes it, the authority they hold, the time in the schedule to do it, and the read that shows whether it's holding. A design without the means to impose its own standard isn't finished, and it doesn't get built.

Impose, don't redesign, on an integrity break. When the trace lands on a standard the design already gave an owner and the means to impose, nothing gets re-engineered and nothing

gets bought. The owner imposes the standard. Redesigning an area that was fully realized throws away a design that works to cover a standard nobody held.

Stop reconciling to a percentage. A line set by an average or a historical number gets re-engineered against what the shift needs to accomplish and what the business needs done to grow. The operator asks what tasks need to be done to grow the business, then asks what gets cut when he hits the percentage.

Refuse the piece until the question is answered. No tool, program, or platform gets bought to fix a broken line until the question has located the break and the redesign has been named. The tool gets evaluated against the redesign, not against the line.

Separate replacement from repair. Parts the design expects to wear out get replaced without ceremony. Breaks get the question. The operator stops filing breaks as wear.

Ask it at design. Every new area, footprint, role, schedule, and cost structure gets the design form before it's built. The box gets sized to the crew the business can afford, and every area gets checked against everything else being built.

Name the level. Every break has its own first cause at its own level. The operator names the level before he names the cause, so a labor break gets located in the labor model and a whole-business break gets located in how the business was conceived.

What Changes Tomorrow

Take the line that broke most recently, and if nothing is obvious, take labor. Pull last period's schedule and the target it was written to. Write down how the target was set. If it came from a percentage, an average, or last year's number, the schedule was written to reference, not engineered to the business.

Then list the work the business needs done to grow: equipment maintenance, deep cleaning, four-walls marketing, training, and anything else that moves the operation forward. Mark every item that didn't get scheduled because the hours weren't there. Each mark is a place the labor model wasn't engineered for the business it's supposed to grow.

Now ask the Engineering Question of the labor model: was it engineered to operate this way, under the circumstances it faces today, including the rates you pay now, and congruent with the overall design of the business? If the list has marks on it, the answer is no, and the trace has already found the area. Don't reconcile the line. Run Question Dumb Shit on what the schedule

currently does, stop whatever hinders the redesign, and re-engineer the model, and the box if it has to come down to fit the crew you can afford.

If the trace instead lands on a standard the schedule already gave hours and an owner to impose, and it wasn't imposed, that's an integrity break. Leave the model alone and have the owner impose the standard.

Ask the question first, about everything. The fix you make without it is the fix you'll be making again.