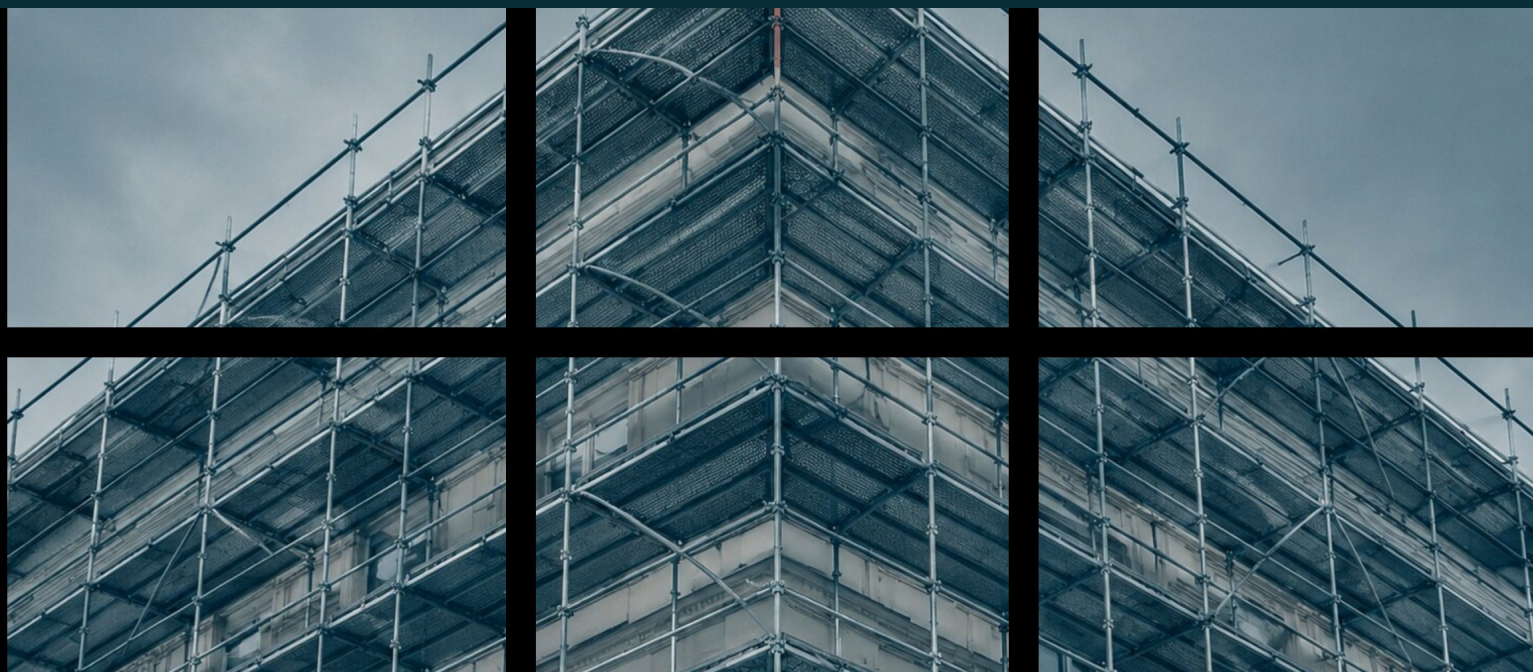


BEYOND WRITTEN REQUIREMENTS

The Offshore Team Delivered Exactly What We Specified — and That Was the Problem

Why distance exposes the business context that requirements documents strip away from software delivery



Giovanni Leonardi

February 2005

The Offshore Team Delivered Exactly What We Specified – and That Was the Problem

Transformation · No. 46

Originally written February 2005 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

“By the time the work reaches the person writing the code, the ambiguity has been removed from the document but not from reality.”

1. The release that matched the specification

The offshore team delivered the release on the agreed Friday.

The code compiled cleanly. The supplier's tests passed. The defect count was within tolerance and the change log showed that every approved requirement had been implemented. By the measures in the contract, this was a good delivery.

During the first business trial, a customer-service representative tried to amend an order that had already been split between two depots. The new screen allowed the address to change, but applied it only to the undelivered line held at the first depot. The second depot retained the old address. Both records were valid. Together, they sent one order to two places.

The development team had followed the rule it had been given: an address change updates open order lines. Nobody had explained that one customer order could be represented in two warehouse systems, or that the service representative was accountable for the whole order rather than the screen in front of her.

The team had delivered code. It had not been allowed to develop understanding.

That distinction sits beneath many of the difficulties now attributed to offshore development. We speak of language, culture and distance, all of which can matter. But the deeper problem is often created by the delivery model itself. It treats understanding as information that can be packed into a requirements document and shipped with the work.

Understanding does not travel that way. It grows through exposure to consequences, exceptions, questions and decisions.

2. The long distance inside the organisation

Geography creates visible distance: different working hours, telephone lines that make interruption awkward, and fewer informal conversations. Yet some of the longest distances in an offshore programme are organisational.

A business user explains a need to an internal analyst. The analyst writes a requirement for the buyer's project office. A supplier coordinator interprets it for an offshore team lead. The team lead divides it among designers and programmers. Questions travel back through the same chain.

Each hand-off makes the message cleaner and poorer.

The business user's original account contains uncertainty: "Most orders work this way, except when stock is split, when credit is held, or when a customer changes the delivery point after dispatch." The requirement turns that uncertainty into a sentence that can be priced and accepted. The programme values the sentence because it creates control. The delivery team receives the control but not the conditions that made it necessary.

This is not merely a communications failure. It is a compression mechanism.

- Business language is translated into system behaviour.
- System behaviour is translated into contractual scope.
- Contractual scope is translated into tasks.
- Tasks are completed without direct sight of the business consequence.

By the time the work reaches the person writing the code, the ambiguity has been removed from the document but not from reality.

A precise requirement can still be precisely incomplete. Clarity of wording is not the same as completeness of understanding.

3. Why the team learns to stop asking

Offshore teams are often told to ask questions, but the surrounding system discourages them.

A fixed-price arrangement rewards the supplier for controlling interpretation. A question that exposes a missing rule may become a change request, which immediately gives both parties a commercial interest in the answer. The buyer may hear opportunism where the team intended clarification. The supplier may hear scope expansion where the business intended explanation.

Status reporting adds another pressure. A team with many open questions can appear less capable than a team progressing through assigned tasks. Analysts therefore make reasonable assumptions to preserve momentum. The assumptions are recorded, but their operational consequence may not be tested until much later.

Hierarchy can deepen the effect. Junior staff may hesitate to challenge a senior buyer or state that a requirement makes little business sense. That hesitation is sometimes described lazily as a national cultural trait. In practice, buyers help to create it when they treat challenge as delay, send questions through several levels and praise teams for quiet compliance.

Time difference then turns a ten-minute conversation into a two-day cycle. A question is raised near the end of one working day, clarified during the next, and answered after another hand-off. Under schedule pressure, the team chooses between waiting and assuming.

The predictable behaviour follows:

- Ask only when the document is internally contradictory.
- Interpret silence as acceptance.
- Deliver the narrowest testable meaning.

- Treat discovered context as a change.
- Protect the milestone even when understanding remains thin.

The team has not failed to communicate. It has adapted rationally to the incentives around communication.

4. A composite delivery with ninety-three per cent acceptance

Consider a programme moving a regional order-management process onto a common packaged system. An offshore supplier configures and extends the application. The buyer retains process design, acceptance and supplier management.

The requirements pack contains 280 numbered items. After six months, 260 pass formal acceptance tests: 93 per cent completion. The remaining twenty concern reports, minor field rules and one unresolved interface. The programme board agrees that the first operational trial can proceed.

The trial uses twelve customer-service staff and representative orders from three depots. Within two days, 41 business queries are raised. Only nine are software defects against the written requirements. The others expose missing understanding:

- Orders split across depots cannot be amended consistently.
- Credit holds are released at account level, while operations manages them by order.
- Substitute products inherit the price but not the promised delivery date.
- Returned goods create a second customer balance that staff cannot see on the main screen.
- The daily warehouse cut-off differs by depot, but the application uses one time.

None of these rules is obscure to the operating staff. They have managed them for years through local screens, telephone calls and paper notes. But the process workshops concentrated on the standard path, and the supplier's team never observed a busy afternoon in the service centre.

The programme initially labels the 32 non-defect queries as new scope. Commercially, that position is defensible. Operationally, it is useless. The system cannot support the process the business actually performs.

A recovery review changes the approach. Two experienced service representatives work directly with the supplier's senior analyst under an agreed protocol. They take one complex order at a time and trace the business consequence through application, warehouse and account records. The offshore team demonstrates the screen while the users explain not only what is wrong but why it matters.

In three weeks, the parties identify eleven genuine changes, eight training issues, six master-data problems and seven local practices that should be eliminated rather than automated. The distinction is important. Without shared understanding, every surprise looked like a software defect or a commercial dispute. With understanding, the programme could choose the correct remedy.

The code did not suddenly improve because conversation replaced discipline. It improved because disciplined conversation connected system behaviour to operational consequence.

5. The strongest case for specification

There is a serious objection to this argument.

Offshore delivery becomes uneconomic if every programmer must acquire the buyer's full business context. Specialisation works because complex work is divided. A well-designed requirement should allow a competent team to deliver without sitting beside every user. Too much direct contact can produce informal scope, conflicting instructions and solutions shaped by the loudest stakeholder. The buyer's failure to define its process should not become the supplier's unlimited obligation to discover it.

All of this is true.

Documentation, modularity and controlled communication are not bureaucratic obstacles. They are necessary conditions for delivery across distance. A supplier cannot price an intention, and a programme cannot govern hundreds of conversations as if they were instructions.

But the conclusion is not that context is unnecessary. It is that context must be designed into the delivery system with boundaries of its own.

Not every developer needs to understand the entire enterprise. The team needs enough business understanding at the points where interpretation affects behaviour. It needs access to authoritative users, representative exceptions and rapid decisions. It needs a clear route for turning a discovered ambiguity into a requirement, a training response, a data correction or an explicit exclusion.

The alternative is not efficient specialisation. It is deferred discovery.

6. Build a team that can interpret, not merely receive

The most effective offshore arrangements create a strong interpretive layer between business intent and code. This is more than a coordinator who forwards questions.

The layer needs three capabilities:

- Business consequence: knowing what the user is trying to protect or achieve.
- System consequence: knowing how a rule travels across applications, data and interfaces.
- Commercial consequence: knowing when clarification changes the contracted work.

One person rarely holds all three. A practical arrangement pairs a senior supplier analyst with a buyer business owner and a commercial lead. Questions can then be answered without allowing every conversation to become an uncontrolled instruction.

Five disciplines make the arrangement real.

1. Show the work
 - a. Let the supplier observe real transactions, including busy periods and exceptions.
 - b. Use representative records with sensitive details removed.
2. Explain the why
 - a. Add the business consequence to important requirements.
 - b. State what failure would look like in operation.
3. Test the exception

- a. Build acceptance cases from the awkward ten per cent, not only the normal ninety.
 - b. Include cross-system and end-of-day conditions.
- 4. Shorten the question path
 - a. Give named analysts direct access to authoritative users within agreed limits.
 - b. Set a response time before silence turns into assumption.
- 5. Reward intelligent challenge
 - a. Treat a well-founded question as risk removed, not progress lost.
 - b. Record assumptions and test their consequence before build completion.

These practices do not require constant travel or unrestricted access. They require deliberate exposure at the moments where understanding changes the work.

7. Measure understanding through the quality of decisions

Traditional transition measures count documents, requirements and defects. They should be supplemented with measures that reveal interpretation.

Track how many assumptions remain untested, how many questions pass through more than two hand-offs, how many acceptance failures arise from missing business rules and how often the supplier can explain the consequence of a design decision without asking the buyer to restate it.

One useful test is simple: present the team with a new exception that is not described in the requirements and ask it to reason aloud. A team with understanding will identify the affected process, seek the right evidence and frame the decision. A team with memorised rules will search for the nearest sentence and apply it literally.

“Understanding is visible when the team encounters something the document did not predict and still asks the right question.”

The objective is not cultural uniformity. Different teams will communicate differently, and those differences can be useful. A team that challenges assumptions from another operating environment may reveal weaknesses the local organisation no longer sees.

The objective is a delivery model in which difference does not become distance from consequence.

8. Code is the residue of understanding

The offshore team is often the last visible actor in a much longer chain of simplification. When the result disappoints, it is convenient to say that the team did not understand the business. Frequently, the programme did not create the conditions in which understanding was possible.

It gave the team requirements without observation, questions without authority, tests without representative exceptions and milestones that rewarded literal completion. The

resulting code reflected the system perfectly.

The lesson from the delivery side is not to abandon offshore development, nor to replace specifications with conversation. It is to recognise that specifications carry decisions better than they create them.

Business understanding must be produced through a managed exchange: users expose consequence, analysts interpret it, developers challenge the rule, and governance resolves what changes. Distance makes this exchange more deliberate, but it does not make it optional.

The supplier should be accountable for asking intelligent questions and making assumptions visible. The buyer should be accountable for access, context and timely decisions. Both should be judged by whether the delivered behaviour works in the operating world, not merely whether it corresponds to the words exchanged months earlier.

The team that delivers code without understanding has not necessarily failed to learn. It may have learned exactly what the programme taught it: that compliance with the document matters more than comprehension of the outcome.