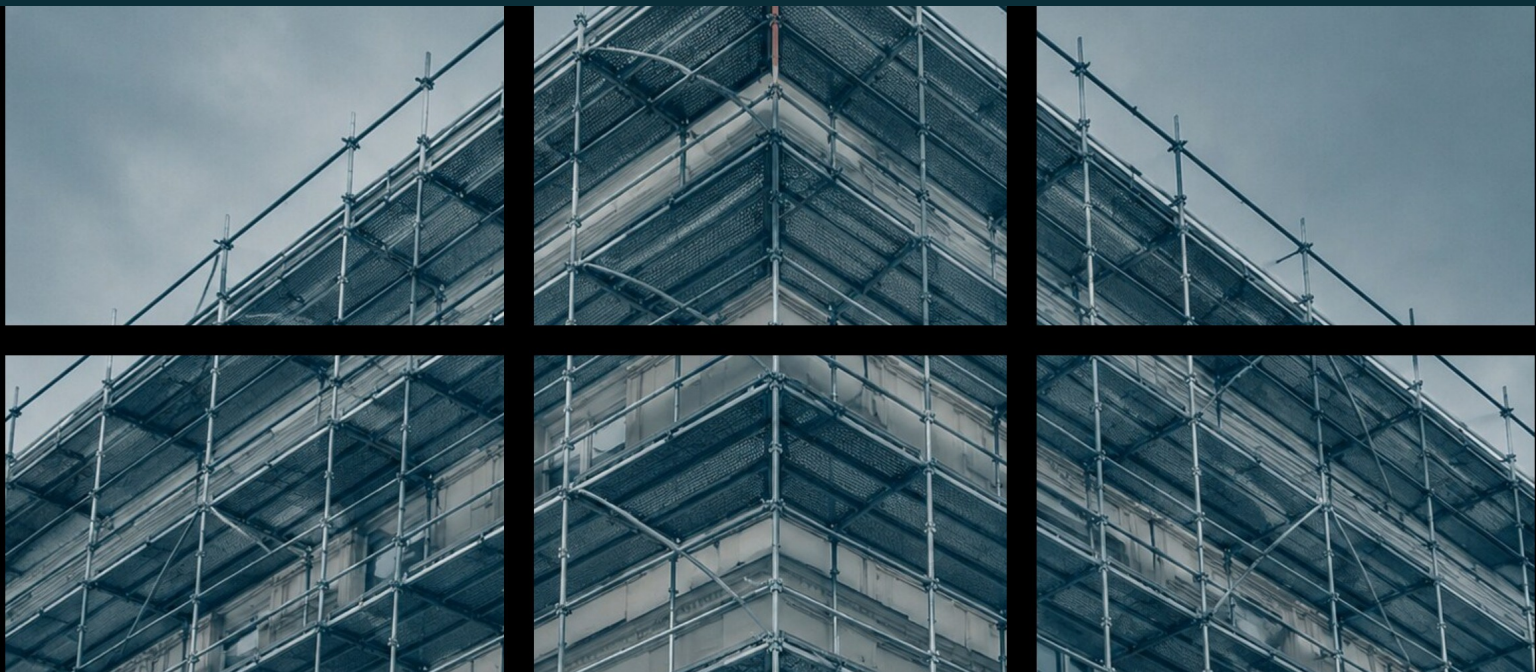


BEYOND WRITTEN REQUIREMENTS

The Offshore Team Delivered the Code but the Client Lost the Understanding

Why written requirements cannot replace shared business context



The Offshore Team Delivered the Code but the Client Lost the Understanding

Transformation · No. 69

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

“Code can cross an ocean as a deliverable; understanding crosses only through repeated working contact.”

Correct Code, Wrong Behaviour

The release contains 86 approved requirements. All 86 are marked complete. The offshore development team has passed unit testing, the systems integrator has accepted the build and only eleven defects remain open — none classified as critical.

During business testing, a branch supervisor enters a request to increase a customer's credit limit. The new screen accepts the request exactly as specified and routes it to the account team. What it does not do is distinguish between a temporary increase for a pending payment and a permanent increase requiring a different approval. That distinction appears nowhere in the requirements pack because local staff have always handled it through judgement.

The software works. The business process does not.

This kind of failure is being described too readily as a cultural or communication problem. Time zones, accents and differing working habits certainly add friction. But the deeper issue is structural: organisations are asking offshore teams to deliver business change while giving them only written instructions about the business.

Code can cross an ocean as a deliverable; understanding crosses only through repeated working contact.

Documentation Is Necessary and Incomplete

The present enthusiasm for offshore development has encouraged a particular operating model. Analysts close to the business define requirements. Documents pass to a remote development centre. Questions return through a coordinator. Completed code passes back for testing.

The model appears efficient because it separates expensive business analysis from lower-cost construction. It also assumes that understanding can be made complete before development begins.

That assumption rarely survives contact with real work. Important knowledge lives in:

- exceptions users no longer think to mention;
- terminology whose meaning changes between departments;

- reasons behind controls that appear unnecessarily cumbersome;
- priorities revealed only when two requirements conflict;
- practical consequences that are obvious to an operator but invisible in a specification.

A document records what somebody knew to write down. It cannot record the questions nobody knew would matter.

In the composite credit-limit release, the offshore developer saw one field, one validation rule and one routing instruction. The branch supervisor saw a decision that could expose the organisation to loss. Both interpreted the same requirement rationally from the context available to them.

The Handover Model Protects the Wrong Efficiency

The case for formal handover is not frivolous. Direct conversation across every boundary can become chaotic. Requirements drift, decisions go unrecorded and suppliers are exposed to endless informal change. Written specifications provide traceability and commercial clarity. They are especially important when teams work different hours and several organisations share delivery.

The mistake is treating documentation as an alternative to working contact rather than the record of it.

The handover model saves analyst and developer time during construction, then spends far more during testing:

1. A requirement is interpreted without business context.
2. The code passes technical tests because it matches the stated rule.
3. A user discovers the missing distinction late.
4. The issue is disputed as defect or change.
5. Analysts rewrite the requirement, developers alter the design and testers repeat the sequence.

The apparent efficiency exists only because rework is counted somewhere else.

Build Understanding Into Delivery

The remedy is not to bring every developer onshore or require every user to join every design discussion. It is to create deliberate routes through which context can accumulate.

Three practices matter now:

- Named business counterparts. Each important process area needs a user or analyst who answers questions, explains exceptions and owns decisions through the release.
- Direct design conversations. Coordinators should organise contact, not filter every exchange. Developers need to hear why a rule exists and probe what happens at its edges.
- Examples before prose. Realistic cases, sample files, exception scenarios and expected outcomes expose ambiguity faster than another page of general description.

For the credit-limit function, five worked cases would have revealed the missing logic before construction: permanent increase, temporary increase, overdue account, exceptional executive approval and declined request. The specification would then record decisions already tested against business reality.

This is not a rejection of disciplined requirements. It is disciplined requirements produced through a richer mechanism.

Retain the Ability to Explain the Business

Offshore teams are often blamed for failing to understand context that the client itself cannot articulate. That is convenient and wrong. If local analysts cannot explain which decisions matter, if business users appear only at acceptance testing and if every question must cross three management layers, distance merely makes the client's weakness visible.

The client must retain more than contract ownership. It must retain people who understand the operating process, can make timely decisions and remain accountable for whether the delivered system serves the business. The supplier can bring technical skill and delivery scale. It cannot manufacture business meaning from an approved template.

The lesson in the current outsourcing wave is therefore not that remote teams cannot understand the business. It is that understanding is not a parcel included in a transition plan. It grows through questions, examples, corrections and repeated contact with the people who live with the consequences.

When that contact is absent, the offshore team may deliver every line of code requested. The client then discovers, late and expensively, that it outsourced construction without creating comprehension.