SCHOOL OF SYSTEMS ENGINEERING

The Interface Control Checklist

SOSE

Nine questions that tell you whether your project is going to integrate.

Most projects don't fail because something breaks. They fail because nobody thought about how everything integrates together.

The gaps between teams go unowned until the day the parts are supposed to meet, and by then a problem that would have cost £1 to fix at the requirements stage costs £100.

> **How to use it.** Work through the nine questions on your current project. If you can answer all nine in writing, your interfaces are under control. Every question you can't answer is a gap, and gaps are where projects fail. Be honest with yourself: a "sort of" is a no.

THE NINE QUESTIONS

☐ 01

Are your requirements SMART?

Specific, Measurable, Achievable, Relevant, Time-bound. If a requirement can't be tested, it isn't a requirement, it's a hope. "The system shall be user friendly" cannot be verified. "The system shall complete a transaction in under two seconds" can.

Gap if: you have requirements nobody could write a test for.

☐ 02

Does each requirement have a named owner?

Not a team. Not a department. A person. Requirements owned by "engineering" are owned by nobody, and the ones nobody owns are the ones that quietly go missing between the spec and the build.

Gap if: you can't put a name next to every line.

☐ 03

Is there a physical architecture with named subsystems?

An actual diagram showing the real parts of the thing you are building, with names. Not a slide of boxes labelled "Platform" and "Services." If you can't draw the system, you can't say where its boundaries are, and the boundaries are the whole game.

Gap if: the architecture only exists in people's heads, or across three different diagrams that disagree.

04

Does every subsystem have a named owner?

Same rule as requirements, and for the same reason. Every box on that diagram needs a person accountable for it. An unowned subsystem is a subsystem where problems accumulate until integration.

Gap if: any box on the diagram has no name against it.

05

Is there a named interface between each pair of subsystems?

Every place two subsystems touch is an interface, and every interface needs an identity of its own. If it doesn't have a name, it doesn't have an owner, a spec, or a test. It just has two teams each assuming the other one has it covered.

Gap if: interfaces are implied by the diagram rather than listed.

06

How do the subsystems connect?

Physically and logically. Which protocol, which bus, which cable, which API, which message format. Written down and agreed by both sides before either side starts building.

Gap if: two teams would give you different answers.

07

What are the connectors?

The specifics of the joint itself. Connector type, pin allocation, voltage, endpoint, schema version, whatever applies to your domain. This is the level of detail where integration actually succeeds or fails, and it's the level most projects skip until it's too late.

Gap if: it's specified to "standard" rather than to a part number or a schema.

08

What is transferred across each interface?

The data, power, signal, fluid or force that crosses the boundary. In what units, at what rate, in what range, in what format. Both sides need the same answer. Mismatched units and mismatched assumptions about range are two of the most common and most expensive integration failures there are.

Gap if: you know what connects but not what flows.

09

How do you prove each subsystem is doing what it was meant to?

For every requirement and every interface, there should be a defined method of verification and the evidence to go with it. Test, demonstration, inspection or analysis. If proof only happens at the end, you will find out about the problem at the point where it costs the most.

Gap if: verification is a phase at the end rather than something running throughout.

NOW COUNT YOUR GAPS

0 to 1	Your interfaces are under control. Keep running this check every time something changes.
2 to 4	You have integration risk that isn't on anyone's risk register yet. Close the gaps now, while it's still a conversation rather than a rework programme.
5 or more	Integration will be the thing that costs you the schedule. This is the normal state for most projects, which is exactly why most projects overrun.

Run it more than once

The single most useful thing you can do with this checklist isn't filling it in. It's filling it in again.

Use it iteratively and recursively, as frequently as your project schedule will allow. Not once at kickoff and not once before delivery. Every time a requirement changes, a subsystem gets redesigned or a supplier changes a part, run it again. That is what flags a problem while it still costs £1 instead of £100.