

Nine questions to answer before you migrate

Most failed upgrades do not fail on the technology. They fail because something nobody documented turned out to be load-bearing. Work through these before you scope the project, and you will know whether what you are planning is an upgrade or a rescue.

01

Which of your customizations does the new version already do natively?

Every major release absorbs things that used to need custom code. Porting a module that the platform now ships is paying twice for the same behaviour, and it is the single most common reason upgrades run long.

A GOOD ANSWER LOOKS LIKE

A line-by-line list of custom modules, each marked port, replace with standard, or retire.

02

Which of them is anyone actually still using?

Customizations accumulate across partners and years. Some were built for a process that no longer exists, or for a person who has left. Nobody removes them because nobody is certain they are dead.

A GOOD ANSWER LOOKS LIKE

Usage data, not opinion. Last-accessed dates, record counts, and a named owner willing to say a module can go.

03

Which integrations break loudly, and which break silently?

A connector that throws an error gets fixed the same day. A connector that quietly stops syncing a field gets discovered at month-end, after the numbers have already been reported.

A GOOD ANSWER LOOKS LIKE

Every integration listed with what happens on failure, and which ones have no alerting at all.

04

What gets validated before cutover rather than after?

Data drift that exists in the old system moves into the new one unless it is caught first. Migrating bad stock figures faithfully is still migrating bad stock figures.

A GOOD ANSWER LOOKS LIKE

A reconciliation pass scheduled before go-live, with a documented tolerance for what counts as acceptable variance.

05

What happens if you have to stop halfway?

Upgrades get paused. A supplier deadline lands, a key person is unavailable, testing turns up something nobody expected. The question is whether pausing is survivable or catastrophic.

A GOOD ANSWER LOOKS LIKE

A parallel-run plan where the current system stays live and usable until the new one is proven, not a single-weekend cutover with no way back.

06

Who decides when a customization cannot be ported cleanly?

This decision gets made mid-project, under time pressure, usually by whoever is closest to the keyboard. That is how a business rule quietly changes without anyone signing off.

A GOOD ANSWER LOOKS LIKE

A named person on your side, not the implementation partner, with authority to accept a process change.

07

What does your reporting depend on that nobody has documented?

Reports are often built on custom fields, manual adjustments, or a spreadsheet step outside the system entirely. These are invisible in a technical audit and only surface when a number comes out wrong.

A GOOD ANSWER LOOKS LIKE

The finance team walking through how each critical report is actually produced, including the steps that happen outside the ERP.

08

Which version are you moving to, and how long is it supported?

Moving to whatever is newest is not automatically right, and moving to whatever is safest can leave you upgrading again sooner than you planned.

A GOOD ANSWER LOOKS LIKE

A target version chosen against your own next three years, with its support window written down.

09

What does another year on your current version cost?

The upgrade has a number attached. Standing still has one too, but it arrives as workarounds, manual exports, and key-person risk, so it never gets compared.

A GOOD ANSWER LOOKS LIKE

A rough annual figure for the manual hours, the deferred capability, and the people risk. It does not need to be precise. It needs to exist.

IF YOU TAKE ONE THING FROM THIS LIST

The triage in question one usually removes more work than any other decision in the project. Teams that carry every customization forward on principle are the teams whose upgrades run long.

