Decision Guide

Adding a new capability or use case: reuse, configure or build separately?

We are adding a new device or a new clinical scenario to a platform that already runs in production. What do we reuse, and when does it become a separate product?

This is not one decision between "extend" and "rebuild". It is a decision per component, made through three lenses: workflow, data and testing. The costly mistakes come from deciding for the whole platform at once, or from letting the new device dictate the architecture.

The guide has two parts, one for a new capability and one for a new clinical use case, and the matrix puts the decisions on one page.

The decision frame

Reuse as is.	The component serves the new case without a change: a device from the same family on the existing transport, or a report template against the existing data model.
Configure.	The same component takes new settings it was built to take: a device profile, a payload mapping, new thresholds in the rule engine, new field mappings to an EHR. If a setting changes the clinical meaning of a result, decide who approves it and what must be re-verified.
Build separately.	The component needs its own code path. That does not automatically mean a separate product. It may require a distinct service, adapter or release boundary. Choose the smallest boundary that keeps ownership and testing clear.

Three lenses place each component.

Workflow:
same users, same roles, same clinical steps?

Data:
does the new device's output fit the existing model, or does it force new semantics?

Testing:
what has to be re-verified, does the verification and validation (V&V) burden change, and can the component follow the existing release rhythm?

A component can land differently under each lens: the workflow may reuse while the testing does not. That is why the decision belongs to the component, not the platform.

Part 1.

A new capability: what can your platform reuse?

Components do not move together. A new device usually leaves some of the platform untouched, asks for configuration in some places and needs real work in one or two.

Device connectivity:
configure or build, depending on transport.

A device on a transport, protocol and pairing model the platform already handles is a configuration task: a profile, a payload mapping, sampling settings. A device that arrives with its own app, cloud or security model is a separate build.

- [ ] Check how stale or partial data is detected.

Data model:
usually configure.

Map the device's units, identifiers and timestamps onto the existing model. Build new elements only when the device measures something the model cannot hold without ambiguity.

- [ ] Check quality flags and provenance.

Assessment and alert logic:
reuse the engine, configure the rules.

New thresholds and parameter sets are configuration. Different rule semantics, such as trend-based logic on an event-based engine, is new work with its own clinical review.

- [ ] Check who approves rule changes and how many alerts a team can absorb.

Clinician interface and reports:
usually reuse and configure.

New views and report templates on the same role-based shell carry over. A user group that needs a different mental model is a workflow signal, not a styling one.

- [ ] Check role permissions and where the view sits in the clinician's workflow.

Integrations:
reuse, then re-test.

The same connector pattern against another endpoint of a supported system is reuse; a first integration with a new class of system is new work. Re-test both directions, because a working read does not prove the write-back.

- [ ] Check API availability, the authentication model and write-back support.

Security and audit:
usually reuse.

Roles, scopes and audit events carry over unless the new data creates a new access pattern or a different deployment boundary.

- [ ] Check data access rules and audit retention.

Test and deployment:
reuse the pipeline, expect new tests.

Regression suites for unchanged paths carry over. New fixtures and V&V for the new device and its rules do not.

- [ ] Check re-verification scope and release rhythm.

Three cases to read against this list.

In a monitoring platform with configurable alerts and reports, that layer takes new settings instead of new code.

In a cardiac monitoring platform we built, raw ECG goes through event classification and prioritization before it reaches clinicians as tasks. Each step carries rules, so a new device or rule set means re-checking the chain, not only the connection.

For a motion-sensor product we implemented the conversion of raw sensor data into assessment results and validated the calculations. That validation is a cost to plan for with every new assessment.

Part 2.

A new clinical use case: platform extension or a separate product to maintain?

A new use case is a candidate for extending the platform when it shares the existing users, data model, clinical workflow and release needs. If the same clinician reviews the same type of result and takes the same next step, the platform may need only a data mapping, an additional assessment configuration or a report template.

Signs that point to extending the platform:

- [ ] Shared users in the same roles, running the same clinical steps
- [ ] Data that fits one model without ambiguous fields
- [ ] The same verification burden and the same release rhythm

Signs that point to a separate product:

- [ ] Different user groups who need a different view, decision or escalation sequence
- [ ] A data model the platform would have to bend to hold the new data
- [ ] A different V&V burden
- [ ] A release rhythm the roadmap cannot share, such as separate customer commitments

The test is whether sharing an implementation keeps behavior understandable, ownership accountable and the test surface manageable. Mixed signals are normal: shared users with a different test burden may still require a separate boundary, because that test burden returns with every release. A separate boundary does not automatically mean a separate product to maintain.

Data is a key consideration when assessing whether to extend a platform or establish a separate product. An example is a product that pulls fragmented inputs into one patient view and one structured clinical report. There, the data model is the first thing to settle.

Integration work raises the write-back question. In a hospital integration we built, data is read from the hospital information system through a secure API and the report is posted back into it. A working read does not prove the write-back, so test both directions.

Before committing, run a boundary review. Put the current and proposed workflows side by side, list the data objects that cross between them, and list the tests that must pass for each release. If the new scenario needs many conditional branches in core logic, different access rules or a different validation schedule, isolate that component, then decide whether the boundary stays internal or becomes a separately maintained product.

What changes across product stages

The same component can land in a different column at each stage of a product's life.

Prototype.

The question is whether the device, the data path and the clinical workflow work at all, with little yet to reuse. Reuse is justified when it shortens the route to a testable hypothesis without hiding a major unknown.

First release.

The question is repeatability. Error handling, configuration limits, role permissions and regression coverage that a demo could skip now decide whether a component can ship.

Scale.

The question is containment. A shared component helps only if a change for one capability cannot silently break another, so ownership of configuration, backward compatibility and release coordination start to matter.

Reuse, configure or build separately: component matrix

Component	Reuse as is	Configure	Build separately	Dependencies to check
Device connectivity	Same device family on the existing transport, protocol and pairing	New profile, payload mapping or sampling settings on a transport the platform already handles	A new protocol or transport, or a device requiring a distinct connectivity implementation	SDK maturity; pairing flow; signal-quality reporting; stale-data detection
Data model	New data maps onto existing concepts, units and identifiers without new semantics	New fields, units and terminology mappings on existing structures	Entities that do not fit the current episode and patient structure	Terminology mapping; migration of existing data; quality flags and provenance
Assessment and alert logic	Existing rules serve the new case unchanged	New thresholds, parameters or routes on the same engine	New rule semantics, e.g. trend-based logic where the engine is event-based	Clinical owner of rule changes; alert-fatigue budget; regression cases
Clinician interface and reports	Same roles, views and report framework	New views and report templates on the same shell	A user group that needs a different mental model or a different decision	Workflow placement; role permissions; report recipients
Integrations	Same connector pattern against another endpoint of a supported system	New mappings and scopes on the existing integration layer	First integration with a new class of system	API availability; authentication model; write-back support; source of truth
Security and audit	Existing roles and audit events already cover the new data	New scopes, roles and audit events	New access pattern, deployment boundary or data-handling requirement	Data access rules; audit retention; role ownership
Test and deployment	Pipeline and fixtures cover the new component	New test data and fixtures in the existing pipeline	A separate test environment or release path	Re-verification scope; V&V burden; release rhythm; rollback

Six decision questions

Workflow

1. Do the new use case's users work in the same roles and screens as the existing ones? ☐
2. Does the new use case follow the same clinical steps, decisions and escalation sequence? ☐

Data

3. Can the new device's output fit the existing data model without ambiguity? ☐
4. Can the existing assessment and rule logic accommodate the new case without changing the meaning of the results? ☐

Testing

5. What must be re-verified for each affected component, and who approves the results? ☐
6. Does the new use case introduce a different V&V burden or raise a regulatory or certification question that needs to be assessed? Can it follow the existing release rhythm? ☐

Where to start

Take the next device or scenario on your roadmap and score it against the matrix, one row per component. For each row marked build separately, decide whether a distinct boundary inside the existing product is enough or whether it should become a separately maintained product.

Compare your path with us

Empeek builds the layer between devices, data systems and clinical workflows for connected-device and remote-care products. Happy to share more about our approach during the meeting.