PRODUCT DEVELOPMENT / ONE-PAGE WORKING CHECKLIST

Seven practical rules for modular product family design

Use these rules while drawing, testing and reviewing module boundaries.

REMEMBER

A module boundary is meaningful only when the module can change without forcing neighbouring parts to be redesigned.

PURPOSE	CONTENTS	SOURCE PUBLICATION
Draw, test and review module boundaries	Quick checklist and detailed guidance	Methodical Development of Modular Product Families

VFAB INTERNAL STANDARD

VFAB 2.0 — Quick Reference Standard

FOR REVIEW

DOCUMENT ID	REVISION	STATUS
VFAB-STD-001	P01	For review
ISSUE DATE	**DOCUMENT OWNER**	**APPROVED BY**
31/08/2026	VFAB Design Team	Not assigned

1. CORE PRINCIPLE

Every VFAB component must have a defined identity, boundary, interface, permitted variation, invariants and verification method.

- **Prefabrication** defines where a component is made.
- **Modularity** defines how interaction between components is governed.
- No declared interface means the component is not approved as a module.

2. SYSTEM HIERARCHY

Structural Module	Container-scale structural frame.
Functional Primitive	An atomic component with no standalone programmatic meaning.
Micro Functional Module	One complete usable room or functional space.
Meso Functional Module	A cluster of rooms spanning at least two Structural Modules.
Macro Functional Module	A complete dwelling composed of multiple Meso Functional Modules, arranged over no more than two storeys.

3. DIMENSIONAL RULES

- Base unit: **75 mm**.
- Construction step: **150 mm**.
- Standard wall bay: approximately **2440 mm nominal**.
- Each standard wall bay contains **eight logical slots**.
- External wall zone: approximately **170 mm**.
- Fabrication coordinates must come from the approved template geometry.
- Do not assume that each slot is 305 mm.

4. PANEL CODE STRUCTURE

Every panel code contains exactly 10 characters:

`[2-character template]` + `[8-position field]`

D	Large door	S	Small door
L	Sliding door	W	Large window
V	Small window	u	Service or utility box
C	Column	x	Ordinary wall
-	Continuation of a multi-slot component		

`01CS--V--x`

P1 = Column
P2–P4 = Small door
P5–P7 = Small window
P8 = Ordinary wall

Prohibited symbols: d, w, A and c

5. INTERFACE CHECKLIST

1. What connects?
2. Where does it connect?
3. What are the dimensions and tolerances?
4. What passes through the interface?
5. What may vary?
6. What must remain unchanged?
7. Who is responsible?
8. How is it inspected and accepted?
9. How is it disconnected, repaired or replaced?
10. What is the Interface ID, revision and compatibility status?

No declared interface = not approved as a module

6. INFORMATION WORKFLOW

Configuration → **Panel code** → **Revit type and instance** → **BOQ** → **Shop drawing** → **Work order** → **QA record** → **Shipping pack** → **Installation location**

Every physical component must be traceable through this complete information chain.

7. MANDATORY QA RULES

- Use only approved document revisions.
- Confirm that Revit, BOQ and shop drawings use matching codes.
- Check panel orientation and handing before fabrication.
- Record every change affecting an interface.
- Apply the required verification trigger after every interface change.
- Fabrication must not begin from WIP or FOR REVIEW information.

8. PROHIBITED ACTIONS

- No unapproved site drilling, cutting or welding.
- No undocumented changes to dimensions, codes or connections.
- Do not assume adjacent RHS beams act compositely without structural design.
- Do not treat mirrored physical panels as identical fabrication items.
- Do not use speculative BOQ quantities as confirmed values.
- Do not manually rename or overwrite published panel codes.

COMPLIANCE

VFAB 2.0 is an internal project system and does not replace statutory requirements.

- The applicable NCC edition and jurisdictional requirements.
- Relevant Australian Standards.
- Approved structural, electrical, fire, waterproofing and accessibility documentation.
- Project-specific engineering and certification requirements.

IF THIS SHEET CONFLICTS WITH AN APPROVED DRAWING, SPECIFICATION OR STATUTORY REQUIREMENT, STOP WORK AND REQUEST CLARIFICATION.

PRODUCT DEVELOPMENT / ONE-PAGE WORKING CHECKLIST

Seven practical rules for modular product family design

Use these rules while drawing, testing and reviewing module boundaries.

01 List what will vary

Define the technical requirements that vary before drawing module boundaries.

DETAIL Build the architecture from recurring technical differences, not existing part names.
WHAT A variation map: what varies and what stays common.
WHY Prevents arbitrary boundaries and hidden variants.
WHEN Concept design, before architecture freeze.
WHERE Requirements list and family matrix.
HOW Use two columns, then link each variation to the function it changes.
EXAMPLE Torque varies; frame mounting points stay common.

02 Put parts that change together in one module

Parts repeatedly changed by the same technical driver are one module candidate.

DETAIL Recurring co-change shows real coupling; one accidental case does not.
WHAT A cluster of components that change together.
WHY Keeps coordinated redesign inside one boundary.
WHEN After comparing variants and change scenarios.
WHERE Function map, bill of materials and layout.
HOW Change one part on paper; mark every forced neighbour. Group recurring clusters.
EXAMPLE Higher-power motor, matching gearbox and cooling unit.

03 Keep parts that change separately in separate modules

Parts driven by different requirements should have separate boundaries.

DETAIL Independent change drivers need independent modules unless safety or physics requires coupling.
WHAT Separate modules for independently changing parts.
WHY Stops one choice multiplying another.
WHEN Parts vary for different requirements or replacement cycles.
WHERE Product structure, assembly definition and option matrix.
HOW Ask: Can A change while B stays fixed? If yes, separate them.
EXAMPLE Battery capacity and cover colour vary independently.

04 Keep one variation in one module

Keep each variable technical requirement inside one bounded module.

DETAIL Localise the effect across parts, software, drawings and tests.
WHAT One variable requirement assigned to one module.
WHY Stops small differences spreading through unrelated systems.
WHEN During variant allocation and module scoping.
WHERE BOM, drawings, software configuration and tests.
HOW Trace every affected item. Move the boundary or standardise the connection if the change spreads.
EXAMPLE A sensor option changes the sensor module and setting only.

05 Use the same module for the same job

Reuse a module only when function, load and environment remain within verified limits.

DETAIL Common appearance is insufficient; verified technical fit controls reuse.
WHAT One unchanged module used in several products.
WHY Reduces parts, drawings, tests, tools and spares.
WHEN A new model needs a function already provided.
WHERE Across the family and suitable adjacent families.
HOW Check function, load, environment and interface against module limits.
EXAMPLE One controller serves several machine sizes within its ratings.

06 Standardise the interface before reusing the module

A reusable module needs one stable interface across all intended uses.

DETAIL Different neighbour connections destroy interchangeability.
WHAT A controlled specification for every boundary connection.
WHY Lets the module change without neighbour redesign.
WHEN Before approving common use or interchange.
WHERE Mechanical, electrical, fluid, software and data boundaries.
HOW Specify fit, tolerance, load, power, data, safety and tests; verify every use.
EXAMPLE Controllers share one mount, connector and communication protocol.

07 Prove that the module can change alone

A useful module can be replaced, resized or upgraded without redesigning its neighbours.

DETAIL Independent change is the practical test of modularity.
WHAT A paper-based change test for each boundary.
WHY Exposes hidden dependencies before release.
WHEN Architecture review and before change approval.
WHERE Module, neighbours, drawings, software, tools and tests.
HOW Make the change on paper and list all outside changes. Redraw if the list is long.
EXAMPLE A larger battery keeps the same space, mount, connector, cooling and control interface.

Reasoning sequence

Move from variation requirements to proof of independent change.

STEP 01

Identify variation

What must change and what must remain common?

→

STEP 02

Set boundaries

Group dependent elements; separate independently changing elements.

→

STEP 03

Control the interface

Control geometry, load, power, data, safety and verification.

→

STEP 04

Test the change

Replace, upgrade or resize without affecting neighbours.

Seven practical rules

Each rule should change how we group, separate, reuse or standardise components.

01

VARIATION

RULE 01

List what will vary

Define the technical requirements that will vary across the product family before drawing or dividing modules. Boundaries must come from recurring technical differences, not existing part names or habits inherited from an earlier project.

WHAT
A short variation map that clearly separates “must vary” from “must stay common”.

WHY
Prevents arbitrary module boundaries and exposes hidden variants early.

WHEN
During concept design, before the product architecture is frozen.

WHERE
The product-family requirements list and comparison matrix.

HOW
Create two columns—“must vary” and “must stay common”—then link each variation to the function it affects.

Example Output torque may vary while the frame mounting points must remain common.

02

GROUP COUPLING

RULE 02

Put parts that change together in one module

Components that repeatedly change together because of the same technical driver are candidates for one module. A single coincidental case is not enough evidence.

WHAT
A module candidate containing components with a recurring co-change relationship.

WHY
Keeps redesign effects inside one boundary instead of allowing them to spread across the product.

WHEN
After comparing variants or simulating likely change scenarios.

WHERE
The function–component map, BOQ/BOM and physical layout.

HOW
Change one component on paper and mark every component forced to change with it; group them if the relationship keeps recurring.

Example If a higher-power motor always requires a matching gearbox and cooling unit, the three may form one drive module.

03

SEPARATE CHANGE

RULE 03

Keep parts that change separately in separate modules

Components that change for different reasons or at different times should not be locked into one module unless a strong physical or safety constraint requires coupling.

WHAT
Separate boundaries for components with independent change drivers.

WHY
Stops one choice from generating unnecessary combinations for another choice.

WHEN
When two components vary by different requirements, users or replacement cycles.

WHERE
The product structure, assembly definition and option matrix.

HOW
Ask: "Can A change while B stays fixed?" If yes, separate them unless a documented technical constraint requires them to remain together.

Example Battery capacity and cover colour vary independently, so the battery and cover should not become one assembly variant.

04

LOCALISE

RULE 04

Keep one variation in one module

Each variable technical requirement should affect only one small, clearly bounded area of the product structure wherever practical.

WHAT
One variable requirement allocated to one clearly identified module.

WHY
Stops a small difference from spreading into unrelated systems.

WHEN
During variant allocation and when defining the scope of each module.

WHERE
The BOQ/BOM, drawings, software configuration and verification plan.

HOW
Trace the variation through every affected item; if it crosses several modules, move the boundary or standardise the connection.

Example A sensor option should change only the sensor module and its setting, not the enclosure, controller and complete wiring system.

05

REUSE

RULE 05

Use the same module for the same job

Reuse a module only when the required function, loads and operating conditions remain within verified limits. Similar appearance is not sufficient technical justification for reuse.

WHAT
One verified module reused unchanged across several products.

WHY
Reduces the parts, drawings, tests, tools and spares that must be managed.

WHEN
When a new model or size needs a function already provided by an existing module.

WHERE
Across models in the same family and, where suitable, adjacent product families.

HOW
Compare the required function, load, environment and interface with the module limits; reuse it only when every condition is satisfied.

Example Use one controller across several machine sizes when its I/O capacity, safety rating and environmental rating remain suitable.

06

INTERFACE

RULE 06

Standardise the interface before reusing the module

A common module must have consistent mechanical, electrical, software and information interfaces across every intended use. If each neighbouring system connects differently, the module is no longer interchangeable.

WHAT
A controlled specification for every connection that crosses the module boundary.

WHY
Allows the module to change without redesigning the surrounding product.

WHEN
Before approving the module for common use or interchange between products.

WHERE
At every mechanical, electrical, fluid, software and information boundary.

HOW
Specify fit, tolerance, load, power, data, safety and verification conditions, then confirm that the same specification suits every intended use.

Example Several controllers can share one mounting pattern, connector and communication protocol without changing the surrounding machine.

07

VERIFICATION

RULE 07

Prove that the module can change alone

A boundary is useful only when the module can be replaced, resized or upgraded without redesigning neighbouring modules. A line on an architecture diagram is not enough to prove modularity.

WHAT
A paper-based change test for every proposed module boundary.

WHY
Exposes hidden dependencies before detailed release, tooling manufacture or verification planning.

WHEN
During architecture review and before design release or approval of a major change.

WHERE
The candidate module, neighbouring modules, drawings, software, tools and related tests.

HOW
Simulate replacing, upgrading or resizing the module and list every external change. If the list is long, review the boundary.

Example A battery module passes when a higher-capacity battery keeps the same space, mounts, connector, cooling and control interface.

CHECK BEFORE APPROVAL

Is the module truly independent?

Tick each statement during the review. Your progress is stored in this browser so you can return and continue later.

- [ ] What must vary and what must stay common have been clearly separated.
- [ ] Components that change together have been considered for grouping.
- [ ] Independently changing components are not locked into the same module.
- [ ] Each variation is localised within the smallest practical module.
- [ ] Reuse has been checked against function, load, environment and interface.
- [ ] Every boundary interface has a specification and verification method.
- [ ] The module has passed an independent replacement, resize or upgrade test.