

Kubicle: An AI-powered Model for Learning Design

Date: September 2026

Author: Frank O'REILLY



CONTENTS

Executive Summary	4
Key Research Findings	4
Key Recommendations	5
1.Introduction	6
Overview	6
Terminology Used	6
2.Design Principles	9
Overview	9
The Tell-Show-Do-Apply Learning Design Framework	9
Dual-stage Apply	10
Additional Design Principles	13
3.AI Learning Taxonomy	15
Overview	15
Purpose and Role	15
Objects in the Taxonomy	15
The Pedagogic Limits of Each Tech Medium	16
4.Pedagogic Mapping Framework	18
Overview	18
Framework Inputs	18
Framework Criteria/Rules	19
Framework Output	20
Outstanding Questions	21
5.Orchestration Model	21
Overview	22
The Shaping Layer (or Learning Orchestration in Practice)	23
How the Shaping Layer Works	23
Next Steps to Enhance the Shaping Layer	24
6.Worked Example	26
Overview	26
Step 1: The initial learning design structure	27
Step 2 - Each slot is populated	27
The selection logic in action	28
Step 3 - Orchestration sets the endpoints and assembles the pathway	29
How the three layers show up in one module	30

7.Recommended Next Steps	32
Appendix A – Cycle Shaping Rules	34

IMPORTANT NOTE:

This report was created by Learnovate at Trinity College Dublin for Kubicle.

Executive Summary

Kubicle's Course in a Day (CIAD) project aims to compress eLearning production from a matter of months to under a day, using generative AI and automation, without sacrificing the instructional quality on which Kubicle's reputation rests. Producing courses in that reduced timescale depends on encoding the judgement of an instructional designer - what to teach, how to teach it, and in what order - as logic a system can apply consistently. This report sets out the pedagogical foundation for that logic.

As part of this project, Learnovate was asked to provide three interconnected components. The first, the AI Learning Taxonomy, is a catalogue of every kind of learning object the platform can build, each described precisely enough that its pedagogical fit can be checked rather than assumed. The second, the Pedagogic Mapping Framework, is the selection logic that chooses the right object for a given point in a course and confirms it fits. The third, the Learning Orchestration Model, is the sequencing logic that determines how objects are ordered and how each hands over to the next. Together they turn a learning goal into a coherent sequence of learning activities: the Taxonomy establishes what is possible, the Mapping what is appropriate, and the Orchestration what happens, and in what order.

The central finding of the work is that these three components form a single, coherent system grounded in established learning science rather than a set of independent deliverables - and that the system is already being adopted into Kubicle's own course-build work.

Key Research Findings

- I. An object is best understood as a vehicle paired with an intent. Treating each learning object as a delivery method combined with a Tell/Show/Do/Apply purpose organises the catalogue coherently and explains why one template can serve several learning intents.

2. Each medium has a natural pedagogic ceiling. Every delivery medium can support learning only up to a certain stage - a video cannot require autonomous practice - and recording those ceilings lets the system rule objects out on principle.
3. Object selection can be made on cognitive grounds, not by quota. The medium of a demonstration should follow the cognitive demand of what is being taught - a defensible, case-by-case rule that replaces an arbitrary fixed proportion.
4. Autonomous application should be rehearsed before it is performed live. Splitting Apply into a risk-free on-platform stage and a real-stakes workplace stage gives learning transfer its own designed step, consistent with Gagné's treatment of transfer as a distinct event.
5. The three components form one integrated system. The Taxonomy, Mapping and Orchestration are interdependent layers of a single pipeline, and the quality of the whole rests on the discipline of the Taxonomy beneath it.

Key Recommendations

1. Complete the object declarations so that every object records the cognitive levels it excludes as well as those it suits - the step that makes the Taxonomy able to validate selections.
2. Test the capability verb-tagging against a real capability register, since this mechanism now underpins several selection rules and its reliability needs proving on genuine material.
3. Specify the remaining mapping rules for practice selection, application placement, depth-band effects and feedback, following the pattern of the one already defined.
4. Resolve the Apply gate, agreeing the object that serves it and settling where its rationale and its build respectively sit.

I. Introduction

Overview

Kubicle empowers enterprises with smart training, delivered at the right time and in the right way, to shape tomorrow's skills.

Kubicle's "Course in a Day" R&D project (Project CIAD) aims to radically transform eLearning production processes by leveraging generative AI, multi-agent orchestration, and automation. The core objective is to reduce the end-to-end course development timeline from the current 60–70 days to a target of less than 24 hours. This strategic initiative is driven by the rapid, AI-led transformation of the L&D sector and the need to quickly update content in fast-moving professional domains.

The proposed solution involves modular workflow re-engineering, using AI agents for tasks from scripting to media synthesis, and critically researching alternative instructional formats like interactive simulations to enhance content longevity. Key innovations include developing a proprietary, end-to-end AI-native production engine capable of unprecedented speed-to-market and an embedded Human-in-the-Loop governance model to ensure instructional quality and accuracy.

The project's main technical uncertainties lie in ensuring AI-generated content consistency, accuracy, and the ability to meet the 24-hour benchmark without compromising the professional accreditation standards for which Kubicle is known.

Terminology Used

Term	Description
Course Creation Framework	This determines the formation of each Course Cycle – notably, how many capabilities a cycle carries and how many cycles a course has. It sits above the Pedagogic Mapping Framework and the Orchestration Model.

Pedagogic Mapping Framework	The selection logic that chooses a specific object from the Object Taxonomy for each slot in a course. It sits between Kubicle's macro course structure (choice of TDSA) and the Object Taxonomy.
Course Cycle	A standalone collection of Tell – Show - Do learning objects aligned to one or more Capability at the same Depth Band
TSDA Slot	A single position in the course requiring an object, classified as Tell, Show, Do or Apply. Supplied by Kubicle's macro Course Creation Framework.
Capability	A single, discrete skill or piece of know-how the course is building. Capabilities are used to map objects and are written fresh for each learning project.
Capability Statement	The written form of a capability, phrased as an action (e.g. "distinguish effective from ineffective feedback"). The verb in the statement is what the selection logic reads to determine the capability's cognitive level.
Depth Band	The depth of treatment a course gives a topic - Novice, Competent or Expert. Set at course level, it controls how many capabilities are bundled per cycle and how often an Apply object appears.
Pedagogic Ceiling	The highest learning stage (Tell/Show/Do/Apply) a given Tech Medium can support - e.g. video cannot exceed Show.
Bloom Level	The cognitive level a capability targets, drawn from Bloom's taxonomy (Remember, Understand, Apply, Analyse, Evaluate, Create). Determines which object is a pedagogical fit for the slot.
Tech Medium	The broadest delivery category for an object - video, static, HTML, AI chat or workplace. Sets the outer limit of what an object can pedagogically achieve.
Object Family	A group of related objects sharing a delivery vehicle (e.g. Motion Graphic, Interactive Widget). Sits between Tech Medium and the individual object.
Object / Object Type	An item from the Object Taxonomy, defined as a pairing of a delivery vehicle and a learning intent (for example, Interactive Widget Demonstrate). The output of the Framework.
Dual-stage Apply	The treatment of Apply as two stages - Rehearsed (autonomous, on-platform, risk-free) and Live (real, in the workplace).

Apply Gate	A standalone Apply checkpoint between cycles, where the learner applies a capability and readiness to progress is judged.
-------------------	---

Table 1: Terminology

2. Design Principles

Overview

The design of the learning tool rests on a small set of established pedagogical principles. Before describing the three components that make up the tool - the Taxonomy, the Pedagogic Mapping Framework and the Orchestration Model - it is worth setting out these underlying principles, because every later decision follows from them.

The foundation is the **Tell-Show-Do-Apply framework**, a widely used instructional-design model that sequences learning through four stages, carrying the learner from receiving information to acting on it independently. It is the organising idea that runs through the entire tool: every learning object is defined in part by which of these four stages it serves, the mapping logic selects objects to fill each stage in turn, and the sequencing logic orders them so that a learner progresses coherently from one stage to the next. To this the design adds one significant refinement - a **dual-stage treatment of Apply**, which distinguishes autonomous application rehearsed safely on the platform from live application performed for real in the workplace.

This section sets out both in turn, and then names three further principles that run through the design as a whole: that cognitive demand should drive design decisions, that feedback is intrinsic to practice, and that support should be withdrawn gradually as a learner gains competence. Together, these principles form the pedagogical basis on which the three components of the tool are built.

The Tell-Show-Do-Apply Learning Design Framework

Tell-Show-Do-Apply is a simple, widely used instructional-design framework that maps cleanly onto the problem this module faces. It sorts every learning activity by who is doing the work - the content, or the learner - and sequences four stages that move from receiving information to acting on it independently.

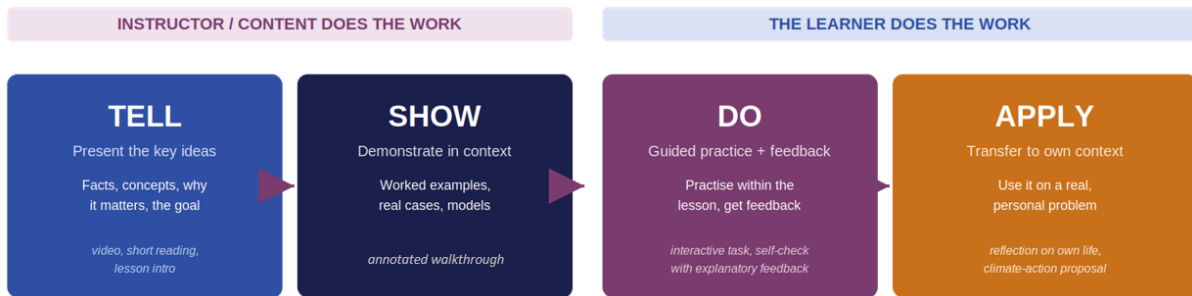


Figure 1: Tell-Show-Do-Apply

The four stages are straightforward. Tell presents the key information - the facts, concepts and, crucially, why it matters. Show demonstrates the idea in context through worked examples and real cases. Do gives the learner guided practice within the lesson, with feedback. Apply asks the learner to transfer the learning to their own real-world environment - their own work context.

Dual-stage Apply

The model treats Apply - the autonomous application of learning to an unfamiliar context - as two distinct stages rather than one. This section sets out the pedagogical reasoning that underpins that structure.

It should be noted that, for the VI release, only the on-platform Rehearsed stage is delivered within the platform; the Live stage takes place in the workplace and sits outside the VI build, though the model accounts for both.

The two distinct stages of Apply are as follows:

The ‘Rehearsed’ Apply (On-platform): The learner independently applies what they have learned to a novel, realistic scenario the consequences of which are shown, but not real. Mistakes cost nothing but provide valuable ‘lessons learnt’.

The ‘Live’ Apply (Off-platform): The learner applies the same skill in the real workplace, with real stakes and real consequences. This is where competency is demonstrated, and ‘Proof of Work’ can be provided for assessment.

Structured practice (Do) takes place on the platform under support; real application (Apply) takes place at work. Between fully supported practice and unsupported real-world performance sits a step that a single-stage model leaves out: autonomous application in a realistic but risk-free setting. Without it, the first occasion on which a learner attempts to transfer their learning independently is in the workplace - the least forgiving place to do so, and the one where the cost of failure is highest.

The rehearsed stage carries the learner from supported to unsupported application while the stakes are still nil; the live stage then carries them from a safe setting to a consequential one, on ground they have already rehearsed. Each transition is taken singly, which is what makes the progression sound.

Failure can be instructive, and the workplace is the costliest place to fail

A risk-free environment lets the learner make a poor decision, see it play out, and adjust – creating a productive-failure cycle on which much of skill acquisition depends.

In the workplace the same error carries real consequences, and the awareness of those stakes can suppress the very experimentation that learning requires. Designing a consequence-free space for autonomous application means the learner can exhaust their instructive mistakes before those mistakes matter. This both protects the organisation from the cost of novice error while protecting the learner’s willingness to take the risks that build judgement.

The Theoretical Foundation for a Dual-Stage Apply

The central theoretical argument for the dual stage approach to Apply rests on Gagné’s Nine Events of Instruction and his treatment of learning transfer as a distinct, intentionally designed event.

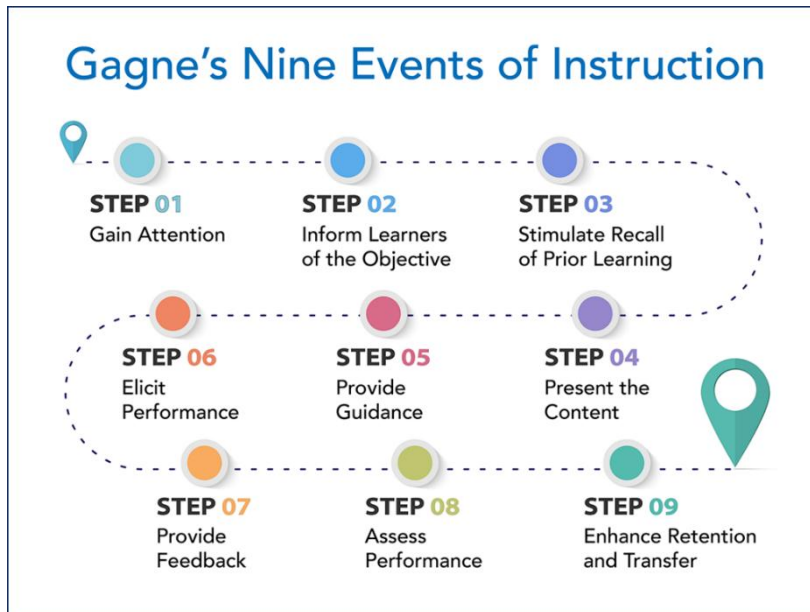


Figure 2: Gagné's 9 Events (Credit: © Penn State is licensed under CC BY-NC-SA 4.0)

In Gagné's model, Assess Performance (Step 8) and Enhance Retention and Transfer (Step 9) are two distinct 'learning activities' where the learning must apply what they have learned. Thus, by adopting a 'dual-stage' approach to Apply, Gagné's Step 8 and Step 9 are accommodated thus adding to the pedagogic strength of the model.

Importantly, the dual stage approach to Apply maps both the 'Rehearsed' Apply and the 'Live' Apply to the Evaluate and Create tiers on Bloom's Taxonomy. In this way, the two Applies occupy the same 'cognitive space' – the only (but important) difference being the context in which they take place and the stakes involved.

The Dual-Stage Apply Model in Practice

Adopting a dual-stage Apply approach simply requires incorporating additional Apply objects that are autonomous, realistic, scenario-based, and on-platform to compliment (and prepare for) the off-platform workplace tasks.

Below are some examples of what these on-platform 'Rehearsed Apply' learning activities might look like (where they sit as objects is discussed elsewhere):

On-platform Apply	Rehearsed application it supports
-------------------	-----------------------------------

Sandbox Assess	Autonomous use of the real tool on an uncontrolled task - in a sandbox, so no real consequences.
Branching Scenario	An unfamiliar, unclear scenario with no single right path. The learner must exercise judgement and see where it leads.
Case Interview	The learner displays their reasoning on an open, realistic case in conversation with an AI chatbot.
Feedback Simulator	Situated, in-role feedback on the learner's output, as if from a real manager or client.

Table 2: Rehearsed Apply Examples

The on-platform / off-platform (the line between 'rehearsed' and 'live') represents an important pedagogic step in the learner journey (and not simply indicative of where the 'Apply' is conducted). In the on-platform environment, the learner applies autonomously at no risk, gains feedback and addresses misunderstandings or gaps. In the off-platform setting the same application happens but this time for real and generates the evidence of competence (Proof of Work).

Summary

Ideally, autonomous application should be rehearsed in a realistic, risk-free, on-platform setting before it is performed live. Doing so is how the system ensures transfer occurs, and how it shortens the path to competence. The rehearsed Apply stage is therefore not an addition to the model so much as the completion of it.

Additional Design Principles

Cognitive demand drives design decisions. How something is taught should align with the cognitive level of what is being taught, not from habit or convenience. This principle underlies the way delivery media are matched to learning stages and the way objects are selected to suit a capability's level.

Feedback is intrinsic to practice. Practice without a response is incomplete and performance without feedback can undermine learning. Therefore, in the proposed design, every object requiring the learner to produce something incorporates feedback, rather than treating it as a separate, optional addition.

Support is withdrawn gradually. A learner should move from supported practice to autonomous application in a safe setting, to application for real - each step removing a little more support than the last. This principle of graduated scaffolding shapes both the progression through a cycle and the dual-stage treatment of Apply described above.

3. AI Learning Taxonomy

Overview

The AI Learning Taxonomy (also referred to simply as the Taxonomy) is the repository of learning objects cataloguing what exists, how it is organised, and the rules governing the appropriate use of each object.

Purpose and Role

The Taxonomy is the catalogue of every kind of learning object the platform can use to build course content. It is the foundation the rest of the system rests on: the Pedagogic Mapping Framework selects from it, and the object-level Orchestration Logic reads the properties it records.

Its purpose is not simply to list what exists, but to describe each object precisely enough that its pedagogical appropriateness can be verified rather than assumed.

Beyond its operational role, the breadth of the Taxonomy carries a wider significance. The range of objects it catalogues demonstrates the variety of learner-intervention techniques that can now be brought to bear within e-learning - a medium that has historically been comparatively limited in how it engages a learner. Cataloguing that range in a structured, selectable form is itself a distinctive contribution of the CIAD approach: it makes the full breadth of available pedagogical techniques visible and usable, rather than leaving them to the ad hoc judgement of individual course builders.

Objects in the Taxonomy

In the Taxonomy, a learning object is defined by the pairing of two independent parts: an Object Family, in essence, the means by which content is delivered (e.g. Motion Graphic, Avatar Guide) and the learning intent it serves expressed through the Tell, Show, Do and Apply framework that underpins the tool's design principles.

This is apparent in the way objects are named. ‘Interactive Widget Discover’ is the object in the ‘Interactive Widget’ family serving a Do (practice) learning intent; while ‘Interactive Widget Concepts’ serves a Tell (presentation) learning intent. Thus, the Object Name is effectively a compound of the two parts (means of delivery and learning intent).

Learning Object Taxonomy - Draft Object List					
Tech Medium	Object Family	Family Description	Object Name	TSDA	Bloom's Level
Video	Motion Graphic	Animated explainer video. Purpose-built visuals, no presenter on screen.	Motion Graphic Concepts	Tell	Remember, Understand
Video	Motion Graphic	Animated explainer video. Purpose-built visuals, no presenter on screen.	Motion Graphic Demonstrate	Show	Understand, Analyse
HTML	Interactive Widget	A bounded, purpose-built interactive element. The learner manipulates it, but within fixed parameters - contrast with Tool Sandbox, which is an open environment.	Interactive Widget Concepts	Tell	Remember, Understand
HTML	Interactive Widget	A bounded, purpose-built interactive element. The learner manipulates it, but within fixed parameters - contrast with Tool Sandbox, which is an open environment.	Interactive Widget Demonstrate	Show	Understand, Analyse

Table 3: Learning Object Taxonomy – Sample Extract

As can be seen from the above table, the catalogue in the Taxonomy is effectively organised on three levels.

At the top is the technology medium (Tech Medium) - how the object is delivered at the broadest level (such as video, file, HTML, or AI chat). Within each medium sit families of related objects (or learning delivery ‘vehicles’), and within each family sit the individual objects carrying its own learning intent.

Thus, the Tech Medium sets the outer limit of what is pedagogically possible; the family groups like with like objects; and the object is the specific, selectable unit or component that can be used to create the learning content.

The Pedagogic Limits of Each Tech Medium

Inherent to the structure of the Taxonomy is that each Tech Medium has a natural ‘pedagogic ceiling’ – that is, the highest learning stage it can support.

Thus, a video can be watched (Tell or Show) but cannot require the learner to practise or produce work of their own (Do) - so it’s ‘pedagogic ceiling’ is Show.

Tech Medium	Content Type	Tell	Show	Do
Video	Passive/consumptive	Y	Y	N
File	Passive/consumptive	Y	Y	N
HTML	Interactive	Y	Y	Y
AI Chat	Conversational	Y	Y	Y

Table 4: Pedagogic Limits of Tech Mediums

These ‘pedagogic ceilings’ are important because they let the system rule an object out of consideration on principle. Thus, a request for an autonomous application activity can never be met by a video, and the Taxonomy makes that a matter of record rather than something a designer must remember.

Currently, however, the catalogue of objects based on the Taxonomy records the cognitive levels each object suits, but not yet, for every object, the levels it excludes.

4. Pedagogic Mapping Framework

Overview

The aim of the Pedagogic Mapping Framework is to determine the selection logic used to choose an option from the Object Taxonomy based on Kubicle's macro course structure logic.

Kubicle's Course Creation Framework (underlying the macro course structure logic) decides that a given slot is a Tell, Show, Do or Apply. The Pedagogic Mapping Framework decides which object fills that particular slot based on a set of criteria and rules outlined below. It sits between the Object Taxonomy (that is the object list) and the macro course framework as the decision layer that turns a pedagogical requirement into an object choice.

At its simplest, the Pedagogic Mapping Framework takes a set of inputs, applies a set of criteria/rules, and defines an output (Object Type).

Framework Inputs

The Framework has the following three inputs:

- **TSDA Slot** - The Tell / Show / Do / Apply Slot chosen for the object to be selected. Supplied by Kubicle's macro framework.
- **Capability 'Bloom' Level** - the cognitive level of the specific capability being taught.
- **Depth Band** - Novice / Competent / Expert. This affects the 'bundling' (or grouping together) of objects at the same depth and the frequency of the Apply objects in the learning journey.

Each capability now carries a cognitive (Bloom) level, derived from how the capability is written. Capabilities statements are authored in a controlled language in which the verb signals the cognitive level - for example, "list" or "recall" indicates a lower level, while "analyse" or "prioritise" indicates a higher one.

The verb is extracted from the capability description and used to determine its Bloom level, so the level is read directly from the wording rather than assigned as a separate step. This supplies the input the Framework depends on: with a cognitive level attached to every capability, the appropriate object can be selected from the Taxonomy.

Framework Criteria/Rules

The Framework applies a small set of rules to turn its inputs into a single object choice. Each rule takes the TSDA slot, the capability's Bloom level and the depth band, and either selects an object or narrows the available options. Every rule carries a pedagogical reason, so the choice can always be explained rather than simply asserted.

Rule 1 - Show medium follows Bloom level

Within a Show slot, the medium follows the cognitive level of the capability being shown. At the Understand level the learner only needs to see a process or example, so a video (Motion Graphic Demonstrate) is sufficient. At the Analyse level the learner needs to grasp how parts relate, which means manipulating something and seeing the result, so an interactive widget (Interactive Widget Demonstrate) is used instead. This replaces the earlier flat '20% of the time' assumption with a rule that can be applied to a single case.

Rule 2 - Do object selection (to be defined)

Where a Do slot has more than one eligible object (for example, Interactive Widget Discover or Guided Construction Practice), the Framework needs a rule to choose between them, likely based on the type and complexity of the capability. Not yet specified.

Rule 3 - Apply object and gate placement (to be defined)

The Framework needs a rule for which Apply object is used at the between-cycle gate, and - where a course has more than one gate - how each offers less support (scaffolding) than the last, so the sequence fades rather than repeating at one level. This applies only where a course has multiple gates (Competent and Expert depth), and is beyond VI scope.

Rule 4 - Depth-band effects (to be defined)

Beyond controlling the bundling (or grouping) and the Apply frequency, the depth band may influence which object is chosen. The Framework needs a rule for how Novice, Competent and Expert treatments change object selection. Not yet specified.

Rule 5 - Feedback characteristics (to be defined)

Feedback is a systematic characteristic of every Do and Apply object. The Framework needs a rule for which feedback approach (how it is delivered, when, and at what level of detail) fits a given capability. Not yet specified.

Only Rule 1 is defined at this stage.

Rules 2 to 5 are identified here so the shape of the Framework is complete and will be specified as the work progresses.

A note on object templates. Beneath the object level, Kubicle's VI work has introduced a further layer of structure that complements these rules. Each object is associated with a selection of HTML widget templates, and each object-and-template pairing is itself aligned to a Bloom level, derived - as elsewhere - from the capability verb. This is best understood not as an additional selection rule but as a structural property of the objects themselves: it extends the same verb-to-Bloom logic that drives object selection down to the level of the specific template used to build the object. Its significance for this Framework is twofold. It confirms, from the implementation side, that the cognitive level can be read reliably from the capability verb; and it means the pedagogical alignment the Framework establishes at the object level is carried through consistently to the templates from which objects are actually built.

Framework Output

The output of the Framework is a single Object Type drawn from the Object Taxonomy - for example, Interactive Widget Demonstrate for an Analyse-level Show slot. Before the object

is returned, it is checked against its own declaration in the Taxonomy: each object records the Bloom levels it is suitable for and the levels for which it must not be used. If the chosen object does not support the capability's Bloom level, the selection is blocked and the Framework falls through to an alternative. In this way the rules propose an object and the Taxonomy confirms it, so the final choice is always a genuine pedagogical fit. In the current VI work this validation is exercised mainly on the Do object, where the selection between practice objects is most active; it applies in principle across all stages as their rules are completed.

This confirmation step depends on the object declarations in the Taxonomy being complete. In the current VI this operates over a deliberately small set of objects, against which the validation approach has been confirmed as sound.

Outstanding Questions

A number of items need to be agreed (or, in some cases, resolved) before the Framework is complete.

Completing the object declarations

Each object needs its suitable and excluded Bloom levels recorded so the Framework's output can be validated. Much of this is already drafted and just needs finishing.

Defining Rules 2 to 5

Rule 1 is written; the remaining rules for Do selection, Apply placement, depth-band effects and feedback characteristics need to be specified.

Confirming how the Apply gates 'fade'

Where a course has several Apply gates, it is recommended that each one offers less support than the last. This is currently beyond the scope of VI.

5. Orchestration Model

Overview

In the original proposal for this project, one of the specified deliverables was the following:

Learning Orchestration Model (Conceptual) – A conceptual model illustrating example sequencing logic showing how learning experiences are constructed.

It is important to clarify the use and meaning of the term orchestration in the report, because it carries a specific meaning elsewhere that is not necessarily the one intended in this context.

In software and AI engineering, orchestration typically refers to the technical coordination of components - the pipeline wiring that governs which service or model runs, in what order, and how data passes between them. However, this is not how the term is used in this context.

What is described here operates one level above that: the reasoning that decides, on pedagogical grounds, which learning objects a course should contain, how they should be ordered so that understanding builds coherently, and how each should hand over to the next so the learner experiences a continuous progression rather than a series of disconnected activities. The concern is the shape and logic of the learning, not the software that executes it.

In that sense, what is presented in this section is best understood as ‘pedagogic orchestration’ rather than ‘technical orchestration’. The distinction is not just a question of terminology. Importantly, technical orchestration is indifferent to what is being sequenced; it would coordinate learning objects no differently from any other components. Pedagogic orchestration is in many ways the opposite: it is concerned precisely with what makes a sequence of learning objects teach well - the order in which capabilities are best introduced, the points at which practice and application should fall, and the transitions that keep a learner oriented as they move through a course.

The rules set out here are therefore grounded in learning science rather than engineering, and the two forms of orchestration are complementary: the pedagogic logic described in this section defines what a well-formed course should look like, and the technical layer, handled separately, is what later gives effect to it.

The Shaping Layer (or Learning Orchestration in Practice)

The purpose of the Shaping Layer is to set out the set of ‘cycle rules’ that will be applied for the selecting the appropriate learning objects and how they are sequenced. In essence, it takes the conceptual model of the orchestration and transforms it into a set of concrete rules executed in the course-build workflow.

While the Learning Orchestration Model (conceptual) discussed above aims to set out what good sequencing looks like and why, Kubicle’s Cycle Shaping Rules (a working draft – see Appendix A) translate this conceptual model into a defined procedure for filling the slots of a cycle - which object opens it, how the Show is delivered, and which practice object the Do takes.

The shaping rules draw their object set directly from the Learning Object Taxonomy, and they build on the mapping logic described earlier rather than replacing it. The rule for choosing a Show medium by cognitive level, for instance, is carried through unchanged and used as the basis for further selection rules of the same kind. In other words, the conceptual model has been taken up as the foundation it was intended to be, with the implementation layer extending it rather than diverging from it.

How the Shaping Layer Works

The shaping rules take a single cycle whose capabilities have already been assigned, each carrying a cognitive ceiling and written as an action and return an ordered set of objects. The rules work slot by slot. The opening object always carries the Tell for every capability in the cycle - a single shared introduction rather than one per capability. The Show is then placed by the same cognitive-level rule established earlier: capabilities at the lower level take their

Show from a video, while those that require the learner to see how variables interact take a separate interactive widget. Where a cycle mixes the two, both routes occur. The result is that the shape of the cycle follows from the capabilities it carries, rather than being chosen in advance.

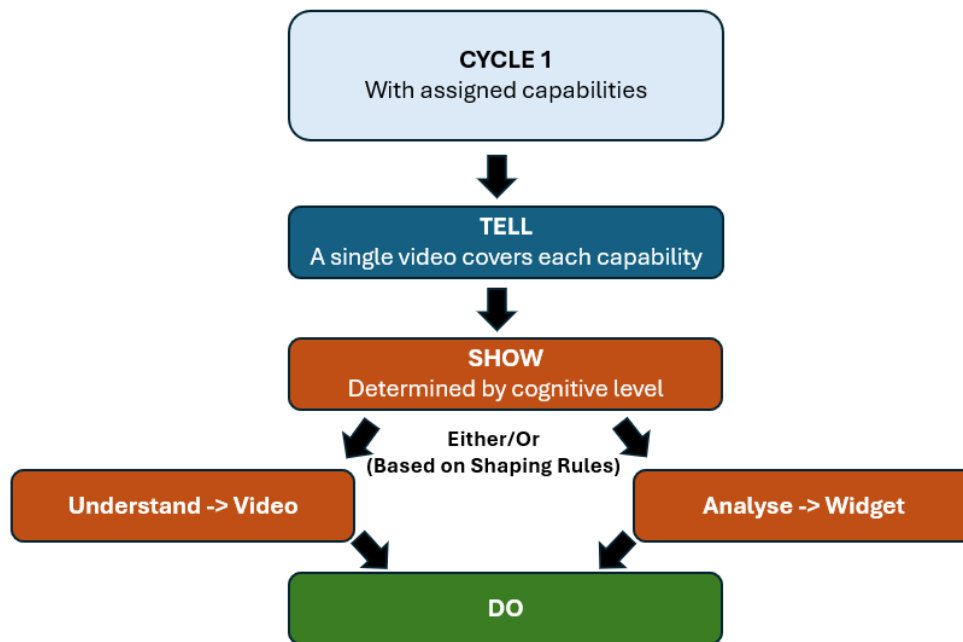


Figure 3: How the Shaping Layer Works

Next Steps to Enhance the Shaping Layer

A logically next step would be to test the shaping rules against a real capability register. Running them against a real set of capabilities is what will show whether the verb-and-structure approach to practice selection holds up in practice, and whether the cognitive-level tagging on which several rules now depend can be applied consistently by those writing the capabilities.

Also worth addressing is the Apply gate. It is the one slot with no object to point at (in VI), and the one whose ownership sits ambiguously between the conceptual model and the shaping layer. Resolving it means two things: agreeing what object serves the gate in VI and confirming that its pedagogical rationale is drawn from this model while its build is handled

in the shaping layer. Until that is settled, the cycle can be shaped but not the checkpoint that follows.

6. Worked Example

Overview

The purpose of this worked example is to demonstrate how the whole system works together – showing how the Taxonomy, Pedagogic Mapping and Orchestration (more specifically, the Shaping Layer) assemble one module (a complete course).

It takes the use case of a management training programme that has a module on **‘Giving Effective Feedback’**. At Novice depth, the module covers a small set of capabilities, and these are pitched here at deliberately different cognitive levels so that each layer of the tool can be seen building the appropriate learning content:

- **Recognise the components of effective feedback** - an Understand-level capability; the learner needs only to grasp what the parts are.
- **Distinguish effective from ineffective feedback in context** - an Analyse-level capability; the learner has to see how the parts interact in a real situation.

These build toward an Apply gate where the learner actually gives feedback in a simulated exchange. The following walks through how the tool assembles the learning objects for the module.

A note on scope for this example: At Novice depth, a single cycle typically carries two or three capabilities together, not one. In that fuller case, all of a cycle's capabilities are introduced in the shared opening Tell; some have their Show carried by that opening video while others, following the Show rule, take a separate HTML widget; and one is practised in the Do slot. To show the selection logic as clearly as possible, the example below deliberately simplifies this — placing one capability in each of two cycles — so that the Show rule can be seen resolving one capability to video and another to a widget without the additional detail of within-cycle bundling. The mechanism illustrated is exactly the same; only the packaging is simplified.

Step 1: The initial learning design structure

Kubicle's macro course framework determines the overall design of the module: a Novice module runs as a series of learning cycles, each following the Tell -> Show -> Do sequence, with a standalone Apply gate between cycles. The tool doesn't decide this structure - it inherits it, then fills each slot with the appropriate learning object.

Module: Giving Effective Feedback (Novice Level)

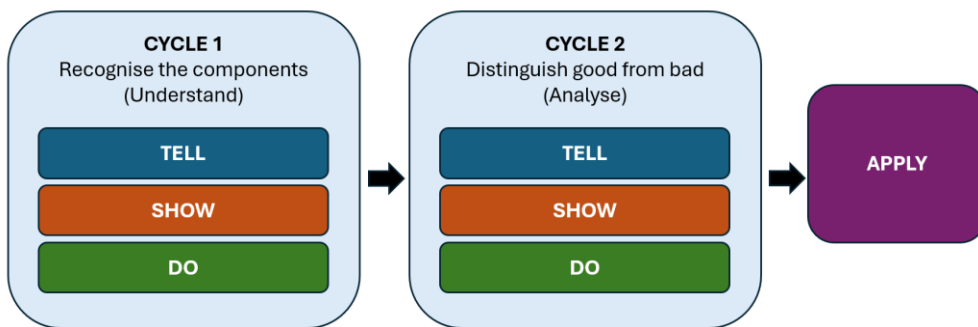


Figure 4: Module Design

The one visible difference between the two cycles above is cognitive level: Cycle 1 delivers an Understand-level capability, Cycle 2 an Analyse-level one. That single difference is what leads the Mapping to choose differently for each - as the next step explains. (As noted above, a live Novice cycle would carry these capabilities together; they are split across two cycles here only to make the contrast legible.)

Step 2 - Each slot is populated

The tool now works slot by slot. For each slot, the Taxonomy proposes the learning objects whose intent matches the stage; the Pedagogic Mapping picks one of these using the capability's cognitive level; and the choice is validated against the declarations determining the appropriate use of that object. It is at the Show slot that the two cycles diverge.

Filling each slot: Object selection logic

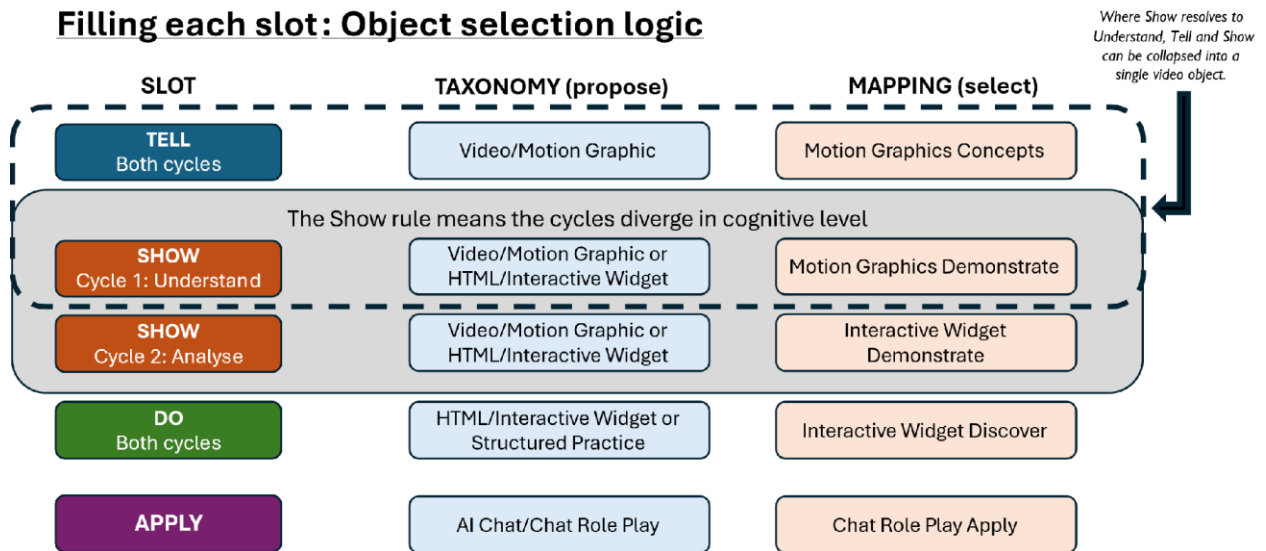


Figure 5: Object Selection

The highlighted Show row is a key element of this worked example. Both cycles face the same choice - video or interactive widget - but the Mapping resolves it differently because the capabilities sit at different cognitive levels. Cycle 1's *recognise the components* only needs the learner to see the parts, so a video suffices. Cycle 2's *distinguish effective from ineffective in context* requires the learner to see how the parts interact when a situation changes - which a video cannot support, so the interactive widget is chosen. In both cycles, the same rule is applied but with a different result determined by the capability.

The selection logic in action

(The rules referred to here are found in the current version of the Cycle Shaping Rules in Appendix A)

Module: Giving Effective Feedback

Cycle 1: What is Effective Feedback

Capability: Define Effective Feedback and identify its key characteristics

Depth Band: Novice

Cognitive Level (Bloom): Understand

Tell is always a video (as per rule T5). Object selected is 'Motion Graphics Concepts'.

Show resolves to a video based on cycle capability and forms a single object with **Tell** (as per rules H1 & H2)

Do resolves to a single object based on cycle capability (as per rule D2). Object selected is Interactive Widget Discover

Cycle 2: Giving Effective Feedback

Capability: Differentiate between good and bad feedback

Depth Band: Novice

Cognitive Level (Bloom): Analyse

Tell is always a video (as per rule T5). Object selected is ‘Motion Graphic Concepts’.

Show resolves to a widget (as per rules H1 & H2). Object selected is ‘Motion Graphic Demonstrate’.

Do resolves to a single object based on cycle capability (as per rule D2). Object selected is Interactive Widget Discover

Step 3 - Orchestration sets the endpoints and assembles the pathway

Once every slot has been filled, the last layer defines what determines whether each object is ‘done’ and joins them into a sequence. In VI this is a fixed on-completion flow (with no branching) so the key element of the process here is deciding the endpoints – including the readiness judgement (‘capability’) at the Apply ‘gate’.

The assembled module : Objects and endpoints

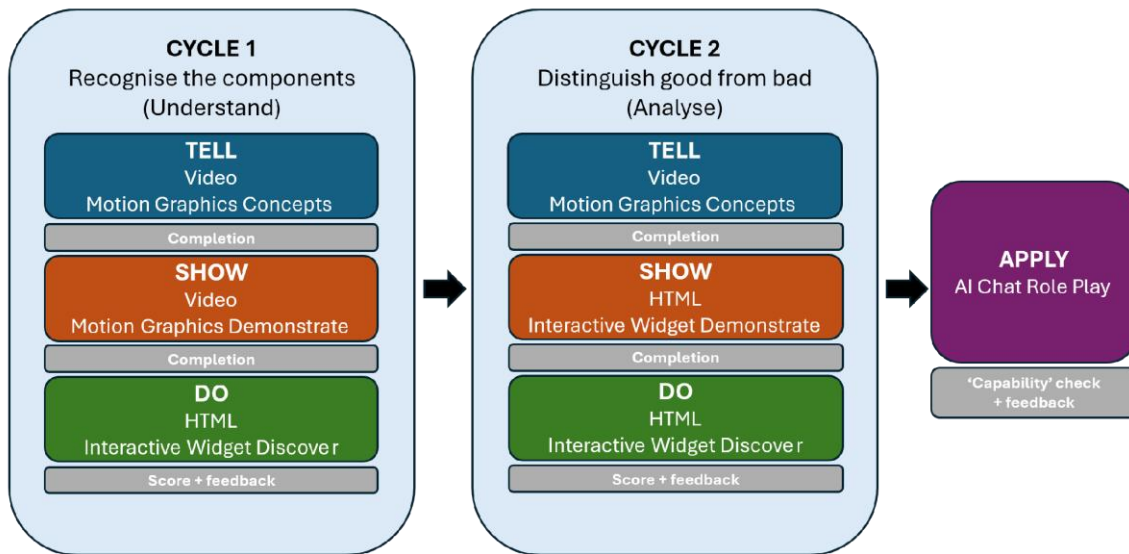


Figure 6: Cycle and Object Sequencing

The finished module in the worked example is composed of seven objects: two teaching cycles feeding an Apply gate. The endpoints reflect the orchestration logic - Tell and Show close on a simple completion signal (nothing is measured, because the learner is receiving), each Do closes on a formative score with feedback (the learner has produced something), and the Apply gate closes on a readiness judgement: the learner gives feedback in a simulated exchange and the AI responds to how they did. In VI every join is a plain 'on completion, advance,' so the sequence is fixed rather than adaptive.

How the three layers show up in one module

To summarise how the different layers come together - each one has a distinct and clear role in building the module content.

The Taxonomy supplied the candidates for every slot - that Tell could be a Motion Graphic, that Show could be either a video or an interactive widget, that the gate could be an AI-assisted role play. It defined the space of what was possible, and - through what each object declares it is good for (the object 'rules') - what was permissible.

The Pedagogic Mapping made the choices, and the feedback module in this worked example shows how it selects an appropriate object rather than being a simple ‘tick-box’: the same Show slot resolved to a video in one cycle and an interactive widget in the other, purely because the capabilities sat at different cognitive levels.

The Orchestration set each object’s endpoint and joined the seven objects into a coherent pathway that builds from recognising the components, to distinguishing good from bad in context, to giving feedback under realistic conditions. It demonstrates what happens, and in what order.

7. Recommended Next Steps

The proposed architecture of the system has been presented in this report and the work that remains is to complete and test the components so the system can run end to end.

The steps below are ordered roughly by priority, and each is drawn from an item already identified in the body of this report.

1. Complete the object declarations

Each object in the Taxonomy currently records the cognitive levels it suits, but not yet, for every object, the levels it must not be used for. Recording those exclusions is what allows the Mapping Framework to validate its choices rather than merely propose them, and it is the single change that turns the Taxonomy from a well-organised catalogue into a self-checking one. Much of this is already drafted; the task is to finish it consistently across all objects.

2. Test capability tagging on a real register

The approach of reading a capability's cognitive level from a controlled set of action verbs now underpins several rules, not only object selection. Testing of this mechanism is already underway on Kubicle's side, against a working set of objects, and the early output has been solid. The remaining task is to extend that testing across a fuller capability register, confirming that the verb-and-structure approach holds up on real course material and that those writing capabilities can apply the controlled language consistently — the point at which the mechanism moves from a validated approach to an established one.

3. Specify the remaining mapping rules

One mapping rule - the selection of a Show medium by cognitive level - is already defined and is used in the worked example above. The remaining rules, for practice-object selection, application placement, depth-band effects and feedback characteristics, are identified, and the two most consequential for VI are already underway on Kubicle's side. Each follows the pattern of the first: read the inputs, propose or narrow an object, and carry a stated pedagogical reason so the choice can be explained.

4. Resolve the Apply gate

Currently the logic around the Apply gate is not settled and how the Apply object will be selected is unclear. As it stands, the checkpoint that follows a learning cycle cannot be resolved.

5. Confirm gate behaviour and fading

Two behavioural questions about the gate remain open: The first - whether it is a soft gate that reports readiness and always advances, or a hard gate that can hold a learner back - is a pedagogical parameter to be determined. The second, the fading of support across successive gates so that each offers less scaffolding than the last, sits beyond the VI scope: it becomes relevant only where a course has several gates, which is to say in the Competent and Expert depth bands rather than at Novice. Both are recorded here so the model is complete, even though only the first falls under the scope of VI.

Appendix A – Cycle Shaping Rules

SCOPE

- S1. These rules apply to lesson objects only. Course objects (course document, glossary, course-long tutor) are out of scope.
- S2. Input: one cycle, with its capabilities already assigned. Each capability carries a Bloom ceiling and is written as an action. The depth band is known.
- S3. Output: an ordered set of two or three objects.
- S4. Cycle formation is upstream and out of scope. How many capabilities a cycle carries, and how many cycles a course has, is decided before these rules run.
- S5. Apply gates sit outside the cycle and are not shaped by these rules.

TELL

- T1. The opening object of every cycle covers EVERY capability in the cycle as a Tell. No exceptions.
- T2. This holds even where a capability's Show is delivered by a later object. A capability mapped to a Show may still require a Tell introduction.
- T3. The Tell is shared across the cycle's capabilities. Do not produce one Tell per capability.
- T4. No object may precede the Tell.
- T5. In V1 the Tell is always a video. A non-video Tell is not reachable by these rules. This is a V1 constraint, not a principle, and is expected to relax.

SHOW

- H1. Determine Show placement by applying the Show Rule to every capability in the cycle:
 - Bloom ceiling at Understand -> Show is carried by video
 - Bloom ceiling at Analyse -> Show requires an interactive widget
- H2. Resolve the cycle from those results:
 - All capabilities resolve to video -> Show sits in the opening video
 - All capabilities resolve to widget -> Show is a separate HTML object

Results are mixed -> both occur

- H3. The mixed outcome requires the cycle to carry more than one capability, so it cannot arise at Expert depth, where a cycle carries one capability. It is most likely at Novice depth, where a cycle carries two or three.
- H4. Where Show sits in the opening video, it covers only the capabilities that resolved to video. Capabilities that resolved to widget take their Show from the widget.

OPENING OBJECT TYPE

01. If no capability in the cycle takes a video Show, the opening object is Motion Graphic Concepts (intent: Tell).
02. Otherwise the opening object is Motion Graphic Exemplify (intent: Tell + Show).
03. Motion Graphic Exemplify declares Bloom for its TELL half. Its Show half is always at Understand, because the Show Rule only ever routes an Understand capability to a video Show.
04. There is one compound object, not several. How much of the Show it carries is a matter for the object's brief, not a separate declared object type.

DO

- D1. The Do always exists as its own demand in every cycle.
- D2. The Do may split across more than one object where the cycle carries multiple capabilities that each benefit from independent practice.
- D3. A Do object never precedes a Tell.

DO SELECTION

- V1. Read the verb in the capability statement. The verb PROPOSES a Do object by pointing at a family of actions, not at a single object. Treat this semantically; do not implement it as a lookup table.
- | | | |
|----------------|--|---------------|
| ordering | (order, rank, prioritise, sequence, stage) | -> Sequencing |
| correspondence | (match, pair, map, associate, attribute) | -> Matching |
| location | (identify, locate, label, spot) | -> Labelling |

```

retrieval      (recall, define, state, name, list)      -> Recall
application    (apply, select, decide, calculate)      -> Discover

```

V2. Read the structure of the knowledge the capability rests on. The structure CONFIRMS or overrides the proposal:

```

a set with one defensible order          -> Sequencing
two sets with correspondences between them -> Matching
one artefact with named parts or positions -> Labelling
discrete facts or terms to be retrieved   -> Recall
none of the above                        -> Discover

```

V3. Where the two signals disagree, STRUCTURE WINS. The reason is buildability, not preference. A Do object needs something to mark against. Where the source carries no single defensible order, a Sequencing object cannot be built even though the capability says "prioritise". The verb can describe an action the content will not support. The content cannot describe knowledge that is not there.

V4. Discover is the residual. Any capability whose knowledge structure resolves to none of the four named structures takes Discover.

V5. Recall is reachable only where the capability's Bloom ceiling is Remember. This is a deliberate Bloom cutoff, and the second in these rules after the Show Rule.

V6. The depth band does not change Do selection. The band has already acted upstream by setting how many capabilities the cycle carries, and the Bloom ceiling already carries cognitive level. The band changes the object's BRIEF (item count, distractor difficulty, scaffolding in the stem), not the object chosen.

SLOT COUNT

- C1. The cycle has a hard maximum of three objects at course level. This cap is never exceeded.
- C2. Three slots is the preferred shape. Prefer three and drop to two only where there is genuinely nothing for a third object to do.
- C3. A two-object cycle is opening video (carrying Tell and Show for everything) plus one Do. It requires every capability to resolve to a video Show, and one Do object to suffice. Expect this to be rare.

C4. The object count is CONFIRMED by the model, never chosen by it. It falls out of the rules above. The model must never deviate from the count these rules produce. Where a deviation appears necessary, that is a human decision.

COMBI OBJECTS

- K1. A Combi is a container: two HTML parts stacked on one screen, where the second part acts on what the first revealed.
- K2. A Combi never contains a video.
- K3. Valid pairings are Show + Do, or Do + Do.
- K4. A Combi is two objects in the build and ONE object at course level. It occupies one slot and counts once against the cap.
- K5. Select a Combi on merit in either of these cases:
 - (a) one capability taken at two rungs, Analyse then Apply
 - (b) the same practice presented in more than one form
- K6. Where the cap squeezes (the shape would otherwise need four slots), fold two demands into a Combi in preference to stretching the opening video or dropping a Do object.
- K7. Construction of the Combi belongs to widget design. Selection of it belongs here.

SLOT RESOLUTION SUMMARY

- P1. Tell slot: resolved by O1 and O2.
- P2. Separate Show slot: Interactive Widget Demonstrate by default. Interactive Widget Explore where the capability requires the learner to change a variable and observe the effect.
- P3. Do slot: resolved by V1 to V6.
- P4. Apply: no V1 object exists. Unresolved, and owned here rather than upstream.

JUDGEMENT

- J1. Where the cap squeezes, resolve it by judgement against these preferences, not by a lookup. The preferences above are checkable after the fact; they are not

a decision table.

J2. Do not harden these preferences into hard rules. Hard rules at this layer are fragile and break at the first nuance.

Note: The Cycle Shaping Rules are included here to record how they were defined at the time this report was prepared since they may (and probably will) evolve in future versions.



Learnovate

 learnovatecentre.org



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

**TECHNOLOGY
CENTRE**
ENTERPRISE IRELAND
IDA IRELAND SUPPORTED 