

Orchestration Load-governed journey mapping methodology

Mapping what the user thinks, Not just what they click

Version: 1.0

Date: 2026-02-27

Companion to: UX Practice OL Diagnosis v2.0, Design Value Preservation Protocol v1.0, Whitepaper v2.0

Purpose: A practical guide for conducting journey-level design using Orchestration Load principles. This document tells you how to map the cognitive journey — not just the task flow.

Audience: UX designers, design leads, and product teams transitioning from screen-level to journey-level practice.

What this is (and what it replaces)

Traditional journey maps document what users *do*: touchpoints, actions, channels, emotions, pain points. They're typically represented as horizontal swimlane diagrams showing an idealized path through a service. When practiced well, they capture the cross-channel experience and reveal moments of friction and opportunity.

In ritualized practice — under sprint pressure and production overhead — journey maps have often been reduced to flowcharts. Step-by-step click sequences. Logic trees. Product-centric representations that capture the interface but miss the person.

The OL-governed journey map doesn't replace the traditional format. It adds five layers that traditional maps don't capture — layers that become critical when AI enters the workflow:

1. **Cognitive Load Profile** — What's happening in the user's head, not just on their screen
2. **Friction Classification** — Which friction is productive (builds capability) and which is overhead (wastes capacity)
3. **Boundary Map** — What happens at every transition between tools, modes, or contexts
4. **Temporal Trajectory** — How the experience changes with sustained use
5. **Sovereignty Checkpoints** — Where the user must think for themselves, and whether the design supports that

These five layers turn the journey map from a task flow diagram into a cognitive architecture document — one that reveals not just *what* the user does, but *what the experience does to the user*.

The full journey principle

One methodological shift governs everything else: **the journey map must cover the complete cognitive workflow, not just the segment within one tool or one project brief.**

When a team receives a brief — "Improve the analysis dashboard" — the natural response is to map the journey within that dashboard. But the user doesn't live in the dashboard. They move from email → news feed → analysis tool → document editor → presentation tool → meeting. The dashboard is one stop in a journey that crosses six tools and five boundaries.

Designing the dashboard in isolation creates what we call a "pager solution" — optimized within its segment, creating load at every transition. The pager solved messaging beautifully. SMS dissolved the boundary between messaging and the device the user already carried.

OL-governed mapping starts from the user's complete workflow and positions AI's role across that workflow — as a continuity layer, not a feature within one tool.

This doesn't mean every project maps every boundary in full detail. It means every project *acknowledges* the boundaries exist and makes conscious decisions about which to design and which to document for later.

The five OL layers

Layer 1: Cognitive Load profile

What it captures: The distribution of cognitive effort across the six OL components at each stage of the journey.

Component	Question to Ask	What You're Looking For
Cc (Coordination Cost)	How much effort goes to managing the tool rather than doing the work?	File navigation, mode switching, format compliance, specification preparation. High Cc = the tool is in the way.
Cv (Verification Capacity)	Can the user evaluate whether the output is correct? Do they?	Presence of evaluation affordances, evidence of checking behavior, quality of source/basis information. Low Cv = blind trust or learned helplessness.
Cm (Context Maintenance)	How much does the user have to remember, re-find, or reconstruct?	Cross-system information fragmentation, documentation decay, reliance on personal memory for decisions. High Cm

Component	Question to Ask	What You're Looking For
		= the system externalizes poorly.
Cr (Cognitive Reserve)	After overhead, how much executive function remains for actual thinking?	Evidence of strategic reasoning, creative exploration, judgment exercise. Low Cr = all capacity consumed by overhead.
Ct (Temporal Degradation)	Does the experience degrade with sustained use?	Quality drift over sessions, calibration shift, growing complacency, declining verification. Present Ct = silent erosion.
Cx (Cross-boundary Load)	What happens at transitions between tools, modes, or contexts?	Context loss, format translation, cognitive switching cost, calibration contamination. High Cx = boundaries destroy value.

How to map it: At each stage of the journey, estimate each component as High / Medium / Low / Absent. You don't need precise scores for design work — you need to see the *shape* of the load. Where is load concentrated? Where is it absent when it should be present (especially Cv)?

The pattern to watch for: Cc and Cm consuming all capacity, leaving Cr depleted and Cv atrophied. This is the signature pattern of the production-trapped workflow documented in the UX Practice Diagnosis — and it's the same pattern that appears in poorly designed AI tools.

Layer 2: Friction classification

What it captures: For every friction point in the journey, whether that friction is productive (builds capability) or overhead (wastes capacity).

This is the distinctly OL-governed layer. Traditional UX treats all friction as bad — something to eliminate. The OL Framework recognizes that some friction is the product. The question isn't "is there friction?" but "is this the *right* friction?"

Classification	Definition	Design Response	Example
Productive friction	Effort that builds skill, develops judgment, or strengthens evaluation capability. Grounded in "desirable difficulties" research — strategic challenges during acquisition enhance long-term learning.	Preserve and support. Make the friction manageable but don't remove it. Provide scaffolding that helps the user do the thinking, not scaffolding that does the thinking for them.	User must articulate their analysis question before AI engages. Writing the question is effortful. That effort is the product — it activates the generation effect and prevents passive consumption.
Overhead friction	Effort that consumes capacity without producing learning or capability. Coordination cost, format translation, context reconstruction, process	Eliminate or automate. This is where AI compression creates the most value — absorbing the overhead so capacity is freed for productive	User must manually transfer analysis parameters from one tool to another by retyping values. This builds no skill. Automate it.

Classification	Definition	Design Response	Example
	compliance.	effort.	
Ambiguous friction	Effort whose classification depends on the user's development stage. What's productive for a novice may be overhead for an expert.	Design for scaffolding fade. Start with productive friction, reduce it as the user demonstrates mastery. The friction serves a temporary developmental purpose.	Novice user required to explain their reasoning before seeing AI suggestions (productive — builds metacognitive awareness). Expert user who's internalized the reasoning pattern (overhead — the habit is formed, the gate slows them down).

How to map it: For each friction point identified in the journey, apply a simple classification: P (productive), O (overhead), or A (ambiguous). For ambiguous points, note the conditions that would shift the classification in either direction.

The decision tool: When classifying a friction point, ask: "If this friction were removed, would the user lose a capability they need?" If yes → productive. If the user would only lose time → overhead. If it depends on who the user is or where they are in their development → ambiguous.

Layer 3: Boundary map

What it captures: Every transition between tools, modes, contexts, or cognitive states — and what happens to the user's mental model at each crossing.

Boundaries are where traditional journey maps stop and OL-governed maps begin. The traditional map shows the user within a tool. The OL map shows what happens *between* tools — the interstitial space where context dies, calibration shifts, and cognitive load compounds.

Boundary Element	What to Document	Why It Matters
Context transfer	What information travels with the user? What gets left behind? What must be reconstructed from memory?	Context loss at boundaries is the single largest source of invisible cognitive cost. The user compensates by carrying it in working memory — which depletes Cr.
Calibration state	Has the user's quality standard shifted? Has their trust level changed? Are they carrying expectations set by the previous tool?	A user who's been working with a high-quality AI output may accept lower quality from the next tool because their reference standard has shifted. This is "calibration contamination."
Cognitive switching cost	How much does the user need to reorient? New interface patterns, new mental model, new vocabulary?	Switching between tools isn't free. Each switch requires rebuilding the interaction model. Frequent switching depletes executive function.
Load inheritance	Does the user arrive tired, depleted, or	A user who's spent 30 minutes managing

Boundary Element	What to Document	Why It Matters
	overloaded from the previous stage? Does load from one tool compound in the next?	Figma component overrides arrives at the design review with depleted Cr — and now needs Cr for strategic evaluation. The review suffers from a boundary the review didn't create.

How to map it: Draw every boundary crossing as a vertical line in the journey. At each line, note: what transfers, what's lost, what the user must supply, and what state they arrive in. These vertical lines are the most valuable part of the OL journey map — they reveal the costs that no single-tool analysis can see.

The pattern to watch for: "Load inheritance" — where the user arrives at a critical thinking task already depleted by overhead from a previous stage. This is invisible to any analysis that scopes to a single tool or feature.

Layer 4: Temporal trajectory

What it captures: How the journey changes with sustained use — not the first experience, but the 10th, the 50th, the 200th.

Traditional journey maps are snapshots — they capture one pass through the experience. OL-governed maps add a temporal dimension that reveals whether the experience builds capability or erodes it over time.

Temporal Pattern	What It Looks Like	Design Implication
Capability growth	User gets better. Less reliance on scaffolding. More independent judgment. Verification becomes faster but not less frequent.	The ideal trajectory. Design for scaffolding fade — reduce support as mastery develops.
Plateau	User reaches stable performance. Neither improving nor degrading. The tool serves its function without developing the user further.	Acceptable for utility tools. Problematic for tools that claim to develop capability. Check: is the user stuck, or are they done?
Automation complacency	User stops verifying. Trust increases as engagement decreases. "It was right before, so it's probably right now."	The most dangerous temporal pattern. MIT EEG research shows neural disengagement persists even after the AI is removed. Design for periodic re-engagement.
Skill atrophy	User loses capability they previously had. Delegation replaces thinking. When the tool is unavailable, performance collapses.	The sovereignty failure. Research shows unrestricted AI users fail 77% of tasks when AI is removed, vs. 39% for scaffolded users.
Calibration drift	User's quality standard gradually shifts. What was "acceptable" creeps. The 100th output is lower quality than the 1st, but the user doesn't notice because	The temporal degradation pattern. Design quality anchors that resist drift — reference standards, periodic resets, explicit comparison points.

Temporal Pattern	What It Looks Like	Design Implication
	the shift was gradual.	

How to map it: Sketch two journey lines — the "Day 1" experience and the "Day 90" experience. Where do they diverge? Where does scaffolding fade (good) versus where does verification decline (bad)? Where has the user's capability grown versus where has dependency developed?

This doesn't require longitudinal research for design purposes. It requires *temporal reasoning* — thinking through what happens with sustained use based on what you know about the user's learning patterns and the tool's feedback dynamics. The post-launch review (Milestone 5 in the Preservation Protocol) is where this reasoning gets empirically tested.

Layer 5: Sovereignty checkpoints

What it captures: The moments in the journey where the user must exercise independent judgment — and whether the design ensures they have the capacity and support to do so.

Sovereignty checkpoints are the "Articulation Before Amplification" moments — points where the user's own thinking must precede the system's contribution. They're grounded in the generation effect: self-generating information produces superior memory and understanding compared to passive consumption.

Checkpoint Type	What It Looks Like	Design Requirement
Articulation gate	The user must state their position, criteria, or intent before the AI contributes.	The input flow asks "what are you looking for?" before showing suggestions. Never lead with the AI's answer.
Evaluation point	The user must assess AI output against their own judgment before proceeding.	Structural affordance for comparison: user's expectation vs. AI's output. Not just "accept/reject" but "here's what I expected, here's what I got, here's my assessment."
Override opportunity	The user can modify, reject, or redirect the AI's contribution at any point.	Override path accessible within one interaction. Not buried in settings. Not penalized by the interface.
Reflection pause	The user is prompted to consider what they've learned or decided before moving to the next stage.	Brief metacognitive prompt: "What did you notice?" or "How does this compare to your initial thinking?" Not mandatory — but structurally present.

How to map it: Identify every point in the journey where the AI makes a contribution. At each point, ask: "Does the user *think* before the AI *acts*?" If not, this is a sovereignty gap. Then ask:

"Can the user *evaluate* after the AI acts?" If not, this is a verification gap. Both need design attention.

The pattern to watch for: Sovereignty checkpoints that were designed for Day 1 but have been bypassed by Day 90. Users learn to skip reflection pauses. They learn to accept without evaluating. The checkpoint must be designed to remain valuable at the 50th use, not just the first — which connects back to Layer 4 (temporal trajectory).

The method: Step by step

Before You Start

Scope the journey. Apply the Full Journey Principle: start from the user's complete cognitive workflow, not from a feature brief. You'll narrow the design scope later, but you need to see the whole journey first to understand where boundaries fall.

Practical shortcut: Ask the user to describe "what happens before and after" whatever feature you're designing. Map at least one stage upstream and one stage downstream of your project scope.

Identify who you're mapping for. OL-governed maps are role-specific and development-stage-specific. A novice and an expert have different load profiles, different productive friction, and different sovereignty needs. Map for a specific user at a specific stage. You can layer multiple personas later, but start with one.

Gather inputs. The OL journey map draws on several sources:

Input	What It Provides	How to Get It
User observation (contextual inquiry, shadowing)	Real workflow, actual boundaries, genuine friction points	Spend time with users in their real environment. Not a lab. Not a prototype. Their desk, their tools, their workflow.
User interview	Self-reported cognitive experience — what feels hard, what feels automatic, where they compensate	Ask: "Walk me through what you did yesterday." Then probe: "What did you have to remember? Where did you get stuck? What did you skip?"
Task flow analysis	The structural sequence of actions	Traditional task analysis methods work here. The OL layers are added on top of the structural flow.
Tool inventory	The complete set of tools the user touches during the journey	Ask the user to list every tool they opened yesterday. Include email, chat, documents — not just the "product" tools.
Stakeholder context	Organizational constraints, process requirements, success metrics	Understand the three-layer process architecture: what does the design

Input	What It Provides	How to Get It
		process demand, what does the development process demand, what does the organizational process demand?

Phase 1: Map the structural journey (1-2 hours)

This is familiar territory — the traditional journey map stage. But with two adjustments:

Map the full workflow, not just the feature. Include at least one stage upstream and one downstream of your design scope. Show the tools, transitions, and contexts the user moves through.

Mark every boundary. Draw a vertical line at every point where the user transitions between tools, modes, or cognitive contexts. These boundaries are where the OL-specific layers will reveal the most.

Output: A horizontal journey map showing stages, actions, tools, and boundary markers. This is the skeleton that the five OL layers will flesh out.

Phase 2: Apply the cognitive load profile (1-2 hours)

Walk through each stage of the journey and estimate the six OL components.

For each stage, note:

- **Cc:** How much effort goes to managing the tool/process vs. doing the actual work?
- **Cv:** Can the user evaluate output quality? Do they? What supports or hinders evaluation?
- **Cm:** How much must the user remember, re-find, or reconstruct? Where is context stored?
- **Cr:** After overhead, how much executive function remains for actual thinking?
- **Ct:** (Estimate) Will this stage degrade with sustained use? Where might complacency or drift emerge?
- **Cx:** (At boundaries) What's the cognitive cost of each transition?

Practical tip: Use a simple High / Medium / Low / Absent scale. Don't aim for precision — aim for shape. You want to see where load concentrates and where it's absent. The most useful finding is often a stage where Cv is absent — nobody's verifying anything — or where Cr is depleted before a critical thinking task.

Output: The journey map annotated with OL component estimates at each stage. Visualize as a stacked bar chart below the journey, or as color-coded indicators at each stage.

Phase 3: Classify the friction (30-60 minutes)

For every friction point identified in Phase 1 and Phase 2, apply the classification:

- **P (Productive):** Builds capability. Preserve it.
- **O (Overhead):** Wastes capacity. Eliminate or automate.
- **A (Ambiguous):** Depends on user's development stage. Design for adaptive scaffolding.

For each Productive friction point, note *what capability it builds* and *how the design supports the user through it*. Productive friction without support is just frustration. The design must make the difficulty *desirable* — challenging but achievable with the right scaffolding.

For each Overhead friction point, note *what would need to change to eliminate it*. This becomes the AI compression target — the specific overhead that freed capacity comes from.

Output: Friction classification overlay on the journey map. Each friction point marked P, O, or A with a brief rationale.

Phase 4: Map the boundaries (1-2 hours)

This is the phase most absent from traditional practice and most valuable in OL-governed mapping.

For each boundary (vertical line) identified in Phase 1:

1. **Document what transfers:** What information, context, or state moves with the user across this boundary?
2. **Document what's lost:** What gets left behind? What must the user reconstruct from memory or re-find in a different system?
3. **Assess the arrival state:** How does the user arrive at the next stage? Depleted? Reoriented? Carrying incorrect calibration from the previous tool?
4. **Estimate the Cx cost:** Is this boundary High (major context reconstruction), Medium (some re-orientation), or Low (smooth transition)?

The boundary design question: For each High-Cx boundary, ask: "Could we design a bridge here? What would it take to carry context across this transition instead of forcing the user to reconstruct it?"

Not all boundaries can be bridged. Some are organizational (different team, different tool, different process). But knowing *where the expensive boundaries are* transforms design priorities. Sometimes the highest-value design intervention isn't improving a feature — it's bridging a boundary.

Output: Boundary annotations on the journey map. Each boundary marked with transfer/loss notes and Cx estimate.

Phase 5: Sketch the temporal trajectory (30-60 minutes)

This is the most speculative phase but often the most revealing. You're not measuring — you're reasoning about what happens over time.

For the journey as mapped, sketch two versions:

The Day 1 experience. The first time through. Where does the user need the most support? Where are they most engaged? Where is verification most active?

The Day 90 experience. The 50th time through. Where has scaffolding faded (good)? Where has verification declined (bad)? Where has the user developed capability (good)? Where has dependency developed (bad)?

The gap between these two sketches reveals the temporal design challenges:

- Where Day 1 support needs to gracefully fade
- Where Day 90 complacency needs counter-measures
- Where the tool should adapt to the user's growing capability
- Where the user might be growing dependent rather than capable

Output: Two journey sketches (Day 1 / Day 90) with temporal divergence notes. These become inputs to the Post-Launch Review in the Preservation Protocol.

Phase 6: Place sovereignty checkpoints (30-60 minutes)

Identify every point in the journey where:

1. **The AI makes a contribution** — generates, suggests, summarizes, automates, recommends
2. **The user makes a decision** — chooses, evaluates, commits, directs

At each AI contribution point, verify:

- Is there an Articulation gate before it? (User states intent before AI acts)
- Is there an Evaluation point after it? (User assesses output before proceeding)
- Is there an Override path? (User can modify or reject)

At each user decision point, verify:

- Does the user have sufficient Cr to make this decision well?
- Is the decision informed by the user's own reasoning, or is it just ratifying the AI's suggestion?
- Is the decision supported by verification affordances (source, basis, confidence)?

Mark gaps. A gap is anywhere the AI contributes without the user thinking first, or anywhere the user decides without the ability to verify.

Output: Sovereignty checkpoint overlay on the journey map. Gaps marked for design attention.

Synthesis: The complete OL journey map

The six phases produce a journey map with six information layers:

Layer	What It Shows	Primary Value
Structural journey	Stages, actions, tools, boundaries	The skeleton — what happens
Cognitive load profile	OL component distribution at each stage	Where capacity is consumed and where it's available
Friction classification	P / O / A at each friction point	What to preserve, what to eliminate, what to adapt
Boundary map	Transfer, loss, arrival state, Cx cost at each transition	Where the invisible costs live
Temporal trajectory	Day 1 vs. Day 90 divergence	How the experience builds or erodes capability over time
Sovereignty checkpoints	Articulation gates, evaluation points, override paths, reflection pauses	Where the user's independent thinking is protected or exposed

This composite map is the artifact. It informs design decisions at every level — from component behavior to journey architecture to temporal strategy. It's also the reference document for the Design Value Preservation Protocol — each milestone checkpoint draws on what this map reveals.

Worked example: The analysis dashboard

To demonstrate the method, here's a condensed walkthrough using the scenario from the UX Practice Diagnosis — the "Improve the analysis dashboard" brief.

Traditional Approach

The team receives the brief. They map the journey *within the dashboard*:

Login → Select dataset → Configure filters → View results → Export report

They identify friction points: slow load times, confusing filter UI, limited export options. They design improvements. The dashboard gets better. The brief is delivered.

OL-Governed Approach

Step 1: Map the full workflow.

The user doesn't start at the dashboard. They start with a question — triggered by an email from a stakeholder, or a pattern noticed in a morning meeting, or a concern about a KPI trend. The full journey:

Trigger (email/meeting) → Formulate question (internal) → Open dashboard → Configure analysis → Review results → Interpret findings → Draft conclusions (document/slides) → Present to stakeholders (meeting)

The dashboard is stages 3-5 of an 8-stage journey. Designing only stages 3-5 ignores where the question comes from and where the answer goes.

Step 2: Apply the cognitive load profile.

Stage	Cc	Cv	Cm	Cr	Notes
1. Trigger	Low	—	Low	High	User has full cognitive capacity; question is fresh
2. Formulate question	Low	—	Med	High	Critical thinking moment — the user shapes what they're looking for
3. Open dashboard	Med	Low	High	Med	Must remember question while navigating to right dataset and view; Cm spikes
4. Configure analysis	High	Low	High	Low	Filter configuration is Cc-heavy; user must hold question in mind while managing tool
5. Review results	Med	Critical	High	Low	This is where Cv should be highest — but Cr is already depleted from stages 3-4
6. Interpret findings	Low	High	Med	Med	User thinking about meaning — but carrying results in memory from tool to document
7. Draft conclusions	Med	Med	High	Low	Translating analytical results into narrative form; Cm high from holding multi-source context
8. Present	Low	—	Low	Med	Delivery mode — cognitive load shifts to communication

Key finding: The user arrives at the critical verification stage (5) with depleted Cr and high Cm — they've spent their cognitive reserve on tool management and are holding context in working memory. This is the worst possible condition for evaluating analysis results. The dashboard design didn't create this problem. The *journey* created it.

Step 3: Classify the friction.

Friction	Classification	Rationale
Filter configuration complexity	O (Overhead)	Builds no analytical capability. Automate or simplify.
Waiting for data to load	O (Overhead)	Pure waste. Technical improvement target.
Need to formulate the question before analyzing	P (Productive)	This IS the analysis. Articulation before amplification. If AI formulates the question for the user, the user's analytical capability atrophies.
Interpreting results requires domain knowledge	P (Productive)	This effort builds expertise. Support it but don't replace it.
Translating results to stakeholder language	A (Ambiguous)	For a novice: productive (builds communication skills). For an expert: overhead (routine translation). Design for adaptive support.

Step 4: Map the boundaries.

Boundary	What Transfers	What's Lost	Cx Cost
Email → Dashboard	The question (in user's memory)	Stakeholder context, urgency cues, original phrasing	Medium — user must reinterpret their question in dashboard terms
Dashboard → Document	Results (via export or screenshot)	Analytical path (which filters, which comparisons, why), confidence level, caveats	High — the "how I got here" is lost; only the "what I found" transfers
Document → Meeting	Narrative (via slides)	Nuance, uncertainty, analytical caveats that didn't fit the slide format	Medium — the format compresses the findings

Key finding: The Dashboard → Document boundary is the most expensive. The user's entire analytical reasoning — which filters they applied, which comparisons they made, why they drew these conclusions — gets lost. Only the final numbers transfer. This means the user must reconstruct their reasoning when questioned in the meeting, and anyone reviewing the document later has no access to the analytical path. This is a Cm problem that no dashboard redesign can solve — it's a boundary design problem.

Step 5: Sketch the temporal trajectory.

Day 1: User carefully configures analysis, reviews results critically, cross-checks against expectations, documents reasoning.

Day 90: User has memorized their common filter configurations. Configuration is faster (good — scaffolding fade). But they've also stopped questioning whether their standard configuration is still the right one (bad — complacency). They export results more quickly, with less documentation of reasoning (bad — verification decline). Their "standard analysis" has become

a ritual — performed the same way each time without questioning whether the question has changed.

Temporal design implication: The dashboard should periodically prompt: "Your analysis configuration hasn't changed in 3 months. Is this still the right framing for your question?" Not a blocker — a nudge. A sovereignty checkpoint that resists temporal drift.

Step 6: Place sovereignty checkpoints.

Point	Checkpoint Needed	Design Implication
Before analysis begins	Articulation gate	User states their question in natural language before configuring filters. The question is preserved and compared against the results.
When results are displayed	Evaluation point	Results presented alongside the user's original question: "You asked [X]. Here's what the data shows. Does this answer your question?"
Before export	Reflection pause	"Your key findings are [X]. Your original question was [Y]. Anything to add before exporting?" Preserves the analytical reasoning that would otherwise be lost at the boundary.
At configuration reuse	Override + awareness	When loading a saved configuration: "This analysis was designed for [original context]. Your current question is [current question]. Does this configuration still apply?"

What the OL map changes

The traditional approach would have improved filter UX, reduced load times, and enhanced export options. Useful improvements. Pager solutions.

The OL approach reveals that:

1. **The highest-value intervention is at the boundary, not in the tool.** Bridging the Dashboard → Document boundary (preserving analytical reasoning through the transition) is worth more than any dashboard UI improvement.
2. **The critical verification moment (Stage 5) is undermined by the stages before it.** Reducing Cc in Stages 3-4 (through AI-assisted configuration) frees Cr for the evaluation that actually matters.
3. **The question itself is the most valuable cognitive artifact.** Preserving the user's articulation of their question — and comparing it against results — is a sovereignty checkpoint that prevents the analysis from becoming a ritual of running the same filters every week.
4. **The temporal trajectory reveals a complacency pattern** that no Day-1 design review

would catch. The Day-90 user needs different support than the Day-1 user.

None of these insights are visible in a traditional journey map. All of them are visible in an OL-governed map.

Facilitation notes

Running an OL journey mapping session

Duration: Half day (4 hours) for a focused journey. Full day for a complex multi-tool workflow.

Participants: Designer(s), product manager, at least one engineer who understands the technical constraints, and ideally a user or user proxy who can describe the real workflow.

Materials: Large wall or digital canvas. Sticky notes or digital equivalents. The OL component reference (six components with their directions). The friction classification definitions (P / O / A).

Structure:

Block	Duration	Activity	Output
1	45 min	Map the full workflow (Phase 1). Start from the user's trigger, not from the feature. Include at least one stage upstream and downstream. Mark all boundaries.	Structural journey with boundary markers
2	45 min	Apply cognitive load profile (Phase 2). Walk through each stage. Estimate Cc, Cv, Cm, Cr. Note where Cr is depleted before a critical thinking task.	Load profile overlay
3	30 min	Classify friction (Phase 3). P / O / A for each friction point. Debate the ambiguous ones — that's where the insight lives.	Friction classification overlay
4	45 min	Map boundaries (Phase 4). For each boundary: what transfers, what's lost, what the user must supply. Estimate Cx.	Boundary annotations
5	30 min	Sketch temporal trajectory (Phase 5). Day 1 vs. Day 90. Where does capability grow? Where does complacency develop?	Temporal divergence sketch
6	30 min	Place sovereignty checkpoints (Phase 6). Where does the user need to think before AI acts? Where is evaluation supported?	Sovereignty checkpoint overlay
7	15 min	Synthesis. What did we learn that we didn't know? What's the highest-value design intervention? What boundary is the most expensive?	Prioritized design insights

Key facilitation principles:

Protect Phase 2 from becoming a usability review. The load profile is about *cognitive*

experience, not *interface* friction. "The button is hard to find" is a usability finding. "The user arrives at the critical evaluation step with no remaining cognitive capacity" is an OL finding. Keep the conversation at the cognitive level.

The friction classification debate is the most valuable 30 minutes. When the team disagrees about whether a friction point is productive or overhead, that's where the real design conversation happens. Don't rush it. The distinction between "this is hard because it builds skill" and "this is hard because the tool is in the way" is the core OL design judgment.

Include the boundaries. The natural tendency is to spend all the time on the stages and skip the boundaries. Resist this. Budget Phase 4 time firmly. The boundaries are where the highest-value interventions live, precisely because nobody else maps them.

Integrating with existing practice

What Doesn't Change

OL-governed journey mapping doesn't replace your existing UX toolkit. It adds cognitive layers on top of methods you already know:

- **Contextual inquiry** still provides the observational data
- **User interviews** still surface the self-reported experience
- **Task analysis** still maps the structural flow
- **Usability testing** still identifies interaction friction
- **Analytics** still quantify behavioral patterns

What Changes

Traditional Practice	OL Extension
Map the journey within the feature	Map the journey across the full workflow, including upstream/downstream
Identify friction → eliminate it	Identify friction → classify it (P / O / A) → respond accordingly
Pain points as problems to solve	Pain points as either problems (overhead) or investments (productive) — distinguished by their effect on capability
Touchpoints and channels	Touchpoints, channels, <i>and boundaries</i> — with explicit Cx cost at each transition
Snapshot of current experience	Snapshot + temporal trajectory (Day 1 / Day 90)
Emotions as a swimlane	Cognitive load components as a profile — more specific than "frustrated" or "delighted"

Traditional Practice	OL Extension
"How do we make this easier?"	"How do we make this build capability?" — the sovereignty question

Minimum viable OL map

If you can't run the full method, the minimum viable addition to a traditional journey map is:

1. **Mark the boundaries.** Just the vertical lines showing where tools and contexts change. Even without detailed annotation, making boundaries visible transforms the conversation.
2. **Classify one friction point.** Pick the most debatable one. Is it productive or overhead? That single classification introduces the concept and reveals whether the team has shared vocabulary for the distinction.
3. **Ask the temporal question.** For one critical stage: "What happens here at Day 90?" That question alone shifts the design perspective from snapshot to trajectory.

These three additions take 30 minutes. They don't require the full method. They start the shift from screen-level to journey-level thinking.

Connecting to the protocol

The OL Journey Map feeds directly into the Design Value Preservation Protocol:

Protocol Milestone	What the Journey Map Provides
Brief Review	The journey context that the brief should account for — upstream/downstream stages, boundary costs, temporal considerations
Concept Review	The cognitive load profile that the concept must address — where Cr is depleted, where Cv is needed, where sovereignty checkpoints belong
Design Review	The friction classifications that the detailed design must preserve (P) or eliminate (O)
Pre-Handoff	The boundary specifications and sovereignty checkpoints that must be explicit in the development specification
Post-Launch Review	The temporal trajectory predictions that post-launch data should validate or disconfirm

The journey map is created once and referenced at every milestone. It's the evidence base that makes the protocol's questions answerable.

Limitations and honest caveats

The cognitive load profile is estimated, not measured. In practice, you're making informed judgments about load distribution, not conducting psychometric assessment. This is sufficient for design purposes but should be clearly communicated as "design-grade estimation" not "scientific measurement."

The temporal trajectory is speculative at design time. You're reasoning about what *might* happen at Day 90 based on patterns from research and experience. The Post-Launch Review is where this reasoning gets empirically tested. Don't present temporal predictions with more confidence than they warrant.

The Full Journey Principle creates scope tension. Mapping the complete workflow is ideal; project briefs are scoped to features. The practical resolution is to see the full journey but *design* within the project scope — with explicit documentation of boundary effects and upstream/downstream dependencies that live outside your brief.

This method adds time. The five OL layers take 4-8 hours on top of traditional journey mapping. In the production-trapped workflow described in the UX Practice Diagnosis, this time doesn't exist. The method becomes practical when AI compression frees the production capacity that currently blocks strategic work — or when teams make a deliberate decision to invest in journey-level mapping for high-stakes projects.

The friction classification requires judgment. There's no algorithm for distinguishing productive from overhead friction. It's a design judgment call that improves with practice and that benefits from diverse team perspectives. When the team disagrees, that's a signal to investigate further — not a signal that the method is broken.

Document Version: 1.0.0

Created: 2026-02-27

Framework Reference: OL Framework Parts 1-5 (v1.1-v1.5), Whitepaper v2.0

Companion: UX Practice OL Diagnosis v2.0, Design Value Preservation Protocol v1.0