

Infrastructural UX (UX3): Designing Safety and Usability Into Web3 Systems

Abstract

Web3 systems expose users to irreversible actions, economic risk, and adversarial conditions that interface-first UX practices are poorly equipped to mediate. Despite continued investment in interface design, onboarding, and education, adoption beyond speculative use remains limited. This paper argues the primary failure is not user misunderstanding, but a misplacement of UX responsibility: critical safety and recoverability decisions are encoded upstream of the UI.

We introduce Infrastructural UX (UX3) as a framework for relocating UX concerns into system behaviour, protocol affordances, state-transition design, and risk mediation mechanisms. In decentralised environments, many user harms originate at the execution layer—transaction finality, permission models, and ownership semantics—where UI improvements cannot retrofit meaningful safeguards. When protocols surface these constraints directly to users without mediation, systems export cognitive load and amplify failure consequences, undermining trust and long-term adoption.

We further argue that machine learning (ML) is a necessary—but bounded—component of UX3. Onchain transactional data provides high-fidelity behavioural signals with temporal structure and economic consequence. Static rules and manual safeguards struggle to adapt to behavioural diversity, shifting threat models, and evolving adversarial tactics. Properly constrained ML pipelines can support behavioural baselining, anomaly detection, and proportional intervention, reducing harm without overriding user agency or decentralisation requirements.

Rather than proposing new interfaces or formalising speculative model categories, this work focuses on system-level primitives: protected states, revocation and deletion semantics, anomaly-aware transitions, and explicit intervention boundaries. It also examines the ecosystem's reliance on human intermediaries to reduce complexity—and why this produces systemic fragility and trust collapse. The paper is written for builders, founders, engineers, architects, and UX professionals designing decentralised systems.

TL;DR

Problem

Web3 UX fails because harm originates at the execution and protocol layer (irreversible transactions, uniform state transitions, unmediated risk), not the interface layer. Better UI alone cannot correct upstream failures.

Solution framework

- UX3 (Infrastructural UX): Relocate UX responsibility into state transitions, protocol affordances, and risk mediation—before any interface is rendered.
- Core primitives: Protected states (anomaly-triggered safety modes), deletion taxonomies (hide/revoke/burn), proportional friction, anomaly-aware transitions, explicit intervention boundaries.
- ML necessity: Onchain data is behavioural signal with economic consequence. Static rules cannot adapt to behavioural diversity or adversarial evolution. ML enables baselining and proportional intervention within UX3 boundaries.

Current failure mode

Web3 frequently outsources complexity reduction to humans (influencers/KOLs/educators/community). When they fail (compromise, misalignment, absence), trust collapses systemically and scales through negative network effects.

For builders

1. UX responsibility is system-level: design state transitions, not just screens.
2. ML pipelines must meet interactive latency constraints, preserve user override, and remain interpretable.
3. Design for failure: false positives, drift, adversarial evasion, degraded modes.
4. Privacy + decentralisation are requirements: prioritise local computation, federated approaches, and cryptographic privacy where appropriate.

Bottom line

Either build infrastructure that absorbs complexity and bounds risk systemically, or continue losing ease-seeking users to trust collapse and ecosystem fragility.

1. The Infrastructural UX Problem

1.1 Web3's adoption barrier is structural, not educational

The dominant narrative frames Web3 UX failures as knowledge deficits: users “don’t understand” wallets, gas, custody, or transaction finality. The prescribed solution is more education—tutorials, influencer content, simplified messaging, gamified onboarding.

This diagnosis is incomplete. The core problem is structural: many Web3 systems export cognitive and operational load to users because the infrastructure does not mediate complexity or risk. Users are asked to reason about:

- Irreversible state transitions (limited recourse, limited undo).
- Adversarial environments (scam contracts, phishing, social engineering).
- Custody responsibility (key management, approval models, multi-chain coordination).
- Economic consequence (actions frequently carry financial risk, often unrecoverable).

These are not primarily interface issues. They are execution semantics and system design issues that manifest as UX failures.

1.2 UX responsibility has moved, but design practice has not

In many Web2 systems, UX feels interface-level because the underlying stack absorbs complexity: error recovery, fraud detection, customer support loops, identity verification, and dispute resolution operate invisibly.

In Web3, execution is often final, state is global, and custody is user-held. This shifts UX responsibility:

- from interface to execution semantics and constraints
- from feedback to affordances (what the system permits before damage occurs)
- from recovery to risk mediation (how the system bounds harm under uncertainty)

We call this Infrastructural UX (UX3): the discipline of designing user experience into system behaviour, protocol affordances, state-transition logic, and risk mediation mechanisms—before any interface is rendered.

1.3 Scope and audience

This paper argues that:

1. Interface-level UX improvements cannot fully solve Web3 adoption because many harms originate upstream: execution semantics, permissions, and recovery paths are infrastructural.
2. UX3 requires new primitives: protected states, deletion taxonomies, proportional friction, anomaly-aware transitions, and explicit intervention boundaries.
3. ML pipelines are often necessary as behavioural mediation layers to scale UX3 interventions, but must remain bounded by UX3 primitives and decentralisation principles.
4. Reliance on human intermediaries (influencers, educators, community moderators) for risk translation introduces systemic fragility and trust collapse.

This is written for builders, founders, protocol designers, engineers, architects, and UX professionals working on decentralised systems. It is a practical reference for designing systems that absorb complexity rather than exporting it to users.

Builder takeaway

- UX responsibility in Web3 frequently sits at the execution and protocol layer, not only UI.
- Education can complement UX, but cannot substitute for missing infrastructure.
- UX3 is an engineering discipline focused on constraints, recovery, and risk mediation.

2. Why Interface-Level UX Fails in Web3

2.1 UIs represent; protocols execute

Once a transaction is signed, the interface no longer controls outcomes. A polished UI can streamline flows and reduce confusion, but cannot on its own prevent:

- irreversible transfers to incorrect addresses
- draining approvals granted to malicious contracts
- permanent loss from mismanaged keys
- exploit execution during time-sensitive operations

The UI is a projection layer; the protocol or execution environment is the outcome layer. UX failures persist when the outcome layer lacks safeguards the UI cannot retrofit.

2.2 False confidence from smoother flows

Reducing friction and improving onboarding can create a misleading sense of safety. Users carry Web2 expectations—undo buttons, password recovery, customer support, fraud protection—into an environment where those affordances may not exist.

This is not a user failure. It is a design mismatch: the system often cannot distinguish between confident expert actions and first-time exploratory behaviour. State transitions are frequently uniform regardless of user context, intent, or competence.

2.3 The delegation trap

Because protocols do not adapt, Web3 stacks often delegate safety to external layers:

- wallets add heuristics (simulation, warnings, risk labels)
- frontends curate options but cannot enforce safety across all access paths
- educators and influencers interpret risk and expected behaviour

This creates:

- fragmentation: mitigations differ across wallets/frontends
- coverage gaps: users can bypass mitigations via alternative clients or direct calls
- trust collapse: if trusted humans or tooling fail, the entire stack looks unsafe

- negative network effects: each failure reinforces “Web3 is unsafe by design”

2.4 Irreversibility is upstream

Consider a user approving unlimited token spending to a malicious contract. A UI can warn, simulate, or require confirmations. But without system-level primitives it cannot:

- constrain the approval by default
- enforce time delays for high-risk transitions
- route suspicious transitions into a protected path
- implement revocation semantics that reduce harm under compromise

These interventions require infrastructural affordances.

Builder takeaway

- UI improvements cannot fully address execution semantics like irreversibility and permission scope.
- Delegating safety to wallets/frontends/education produces fragmentation and bypass paths.
- Harm often originates upstream of the interface—in protocol rules and transition logic.

3. Infrastructural UX (UX3): Definition and Primitives

3.1 What is UX3?

Infrastructural UX (UX3) is the discipline of designing user experience into system behaviour, protocol affordances, state-transition logic, and risk mediation mechanisms—before any interface is built.

UX3 treats execution constraints as the primary UX surface. It asks:

- Which state transitions are permissible, and under what conditions?
- How does the system respond to anomalous or risky behaviour?
- What recovery paths exist when users deviate from expected flows?
- How does the system distinguish exploratory from confident actions?
- Where should irreversibility be bounded, and where should it be absolute?

UX3 is not “hiding complexity”. It is absorbing complexity so interfaces present meaningful choices rather than forcing users to become risk analysts.

3.2 Core primitives

3.2.1 State transition constraints

Not all transitions should be uniformly accessible. UX3 requires defining:

- guarded transitions: confirmations, delays, multisig/secondary checks for high-risk actions
- protected states: escrow-like or restricted modes triggered by anomalies, preventing irreversible loss during review
- reversible windows (where feasible): time-bound cancellation or escalation paths prior to finalisation

Example: A protocol or account system introduces a delay on unusually large withdrawals for newly active accounts. This does not block the action; it bounds irreversibility and buys time for detection and user response.

3.2.2 Deletion semantics taxonomy

“Delete” is ambiguous in Web3. UX3 separates deletion into explicit operations:

1. Local visibility removal: hide from a UI/index without altering global state

2. Revocation / suspension: move into a locked, timelocked, or escrow state; potentially reversible
3. Terminal destruction: irreversible burn/self-destruct/permanent state change; requires deliberate confirmation

Example: “Delete my NFT” must be clarified as:

- hide in gallery only (1)
- move to recovery vault (2)
- burn permanently (3)

Conflating these semantics creates expectation mismatch and avoidable irreversible loss.

3.2.3 Anomaly-aware transitions

UX3 systems detect behavioural deviations and route anomalous actions through protected paths rather than treating all actions equally.

Components:

- baselines: user/cohort patterns (frequency, amounts, counterparties, contract types, temporal rhythm)
- deviation detection: first-time contract use, unusual value relative to history, burst behaviour after dormancy, new counterparty clusters
- proportional friction: additional confirmation, simulation overlays, caps, delays, or review only when risk is elevated

3.2.4 Explicit intervention boundaries

UX3 interventions must preserve sovereignty:

- transparency: users understand why friction appears
- overrideability: informed users can proceed
- proportionality: severity matches risk

- audit trails: decisions and overrides are recordable for review

Red line: silently blocking transactions without user recourse is not UX3; it is control infrastructure.

3.3 Where UX3 should live (coverage framing)

Wallets and frontends can implement UX3 principles, but coverage is incomplete without support at the account/protocol/standard layer: users can bypass UI safeguards via other clients or direct calls.

UX3 primitives are most robust when supported by:

- account-level validation logic (e.g., spending limits, delays, secondary confirmations)
- token/contract standards (revocation semantics, safe defaults)
- execution environments that enable constrained transitions across all interfaces

Builder takeaway

- UX3 = designing experience into state transitions and system affordances, not only UI polish.
- Primitives: guarded transitions, deletion taxonomy, anomaly-aware routing, explicit boundaries.
- Wallet/front-end mitigations help, but consistent coverage improves when system-level affordances exist.

4. Why Machine Learning Is Necessary (Not Optional)

4.1 The scaling problem

UX3 requires behavioural context and adaptive risk mediation. Hard-coded rules struggle with:

- behavioural diversity: novices vs power users, varying patterns and intents
- adversarial evolution: scams and exploit tactics adapt rapidly
- state-space complexity: cross-domain interactions (trading, NFTs, lending, governance, bridging)
- temporal dynamics: both user competence and protocol risk change over time

Static thresholds (“flag transfers > X”) generate unacceptable false positives and false negatives. ML is often required to baseline behaviour per user/cohort and detect meaningful deviations.

4.2 ML as a behavioural mediation layer (bounded role)

ML in UX3 is not “predicting the future” or replacing user judgment. It is a mediation layer that translates behavioural signals into bounded interventions.

A UX3-aligned ML pipeline supports:

- observe: ingest transaction intent + context
- encode: construct behaviour features and baselines
- infer: anomaly/risk estimation relative to baseline and external context
- act: trigger UX3 primitives (caps, delays, protected states, confirmations) proportional to risk

Constraints:

- ML outputs must be interpretable (“why was this flagged?”).
- Users retain override with informed consent.
- ML augments—not replaces—cryptographic safeguards.
- Systems must degrade gracefully if ML is unavailable.

Builder takeaway

- ML is often necessary to scale behavioural mediation across diverse users and evolving threats.
 - ML must be bounded: interpretability, overrideability, proportionality, graceful degradation.
 - UX3 defines permissible interventions; ML helps decide when to apply them.
-

5. Transactional Data as Behavioural Signal

5.1 Why onchain data is different

Web3 produces behavioural signals unlike Web2 analytics:

- explicit intent: actions require signing and typically carry cost
- economic consequence: mistakes and attacks have direct financial impact
- temporal structure: sequences reveal learning curves and patterns
- relational structure: counterparties/contracts form interaction graphs
- global visibility (subject to privacy tooling): actions are broadly observable

This makes onchain data well-suited for behavioural modelling—while raising strong privacy requirements.

5.2 Feature construction (practical modelling inputs)

5.2.1 Temporal features

- transaction frequency, burst detection, dormancy
- recency and time-of-day/weekday patterns

- common action sequences (approve→swap→stake)

5.2.2 Relational features

- first-time contract and counterparty interactions
- contract age and reputation signals (where available)
- interaction breadth: number of unique protocols/assets
- shared counterparties with known risky clusters (where available)

5.2.3 Value-based features

- transaction size relative to wallet history
- approval scope (unlimited vs bounded)
- cumulative drains (many small transfers vs one large)

5.2.4 Expertise proxies (use cautiously)

- wallet age and interaction depth
- repeated safe usage patterns
- evidence of prior risk management behaviour (revocations, spending limits)

5.3 Privacy and minimisation (default posture)

Behaviour modelling does not require centralised data lakes. Approaches include:

- local models: run baselines and inference client-side
- federated learning: aggregate model updates without raw data sharing
- cryptographic privacy: zero-knowledge proofs for specific claims (where feasible)
- differential privacy: noise for aggregates (with careful calibration)

Builder posture: treat privacy as a UX3 primitive. Default to local computation and explicit consent for aggregation.

Builder takeaway

- Onchain transactional data is sparse but intent-rich and economically consequential.
 - Feature construction should encode temporal, relational, and value patterns.
 - Privacy-preserving approaches exist; centralisation is a design choice, not a requirement.
-

6. ML Pipelines in Practice

6.1 Pipeline architecture: observe → encode → infer → act

Observe

- capture transaction metadata: function intent, parameters, value, counterparty, timing
- retrieve contextual signals: contract age, known exploit patterns, network conditions (where available)

Encode

- build features from Section 5
- maintain cached user baselines and thresholds
- encode intent type: approval, transfer, swap, stake, vote, bridge

Infer

- anomaly detection relative to baseline
- risk scoring combining anomaly severity and context

- intent confidence: exploratory vs routine

Act (UX3 primitives)

- low risk: proceed without added friction
- medium risk: simulate, explain, require explicit confirmation
- high risk: protected state, caps, delays, secondary confirmation pathways
- critical matches (where confident): temporary hold with explicit recourse, not silent failure

6.2 Performance constraints (without fragile numbers)

Signing flows are interactive; users will abandon sluggish pipelines. Practical systems typically require:

- fast inference (sub-second experience)
- precomputed baselines and cached reputational context
- lightweight models for realtime screening with slower post-hoc improvement loops
- edge/client execution where possible to reduce latency and centralisation

6.3 Failure modes and how to design for them

6.3.1 False positives (over-friction)

Mitigate via:

- protected paths rather than hard blocks
- override with informed consent
- adaptive thresholds per user, not global rules
- learning from overrides as feedback

6.3.2 False negatives (missed attacks)

Mitigate via:

- cryptographic safeguards (spending limits, timelocks, multisig)
- ensemble approaches (heuristics + anomaly + signature patterns)
- high-stakes always require extra confirmation
- post-incident learning loops

6.3.3 Drift (user behaviour changes)

Mitigate via:

- rolling baselines and recency-weighted learning
- drift monitoring (override rate, alert fatigue)
- staged adaptation (avoid sudden threshold swings)

6.3.4 Adversarial evasion

Mitigate via:

- cumulative behaviour checks (many small drains)
- mixed strategies (multiple detectors)
- adversarial evaluation and simulation during testing

6.4 Example: Approval-risk mitigation (UX3 + ML)

Scenario: user attempts unlimited token approval to an unfamiliar, newly deployed contract.

Pipeline:

1. Observe: approval intent + unlimited scope + new spender
2. Encode: contract age/reputation signals + user baseline + novelty

3. Infer: high deviation severity + elevated contract uncertainty
4. Act: enter protected state (cap exposure), surface simulation and explanation, offer safe defaults

Example intervention pattern:

- default to bounded approval aligned to intended action
- require explicit acknowledgement for unlimited approvals
- apply temporary caps for first-time interactions
- allow override with clear consequences

Builder takeaway

- Implement ML as bounded mediation: decide when to activate UX3 primitives, not replace them.
 - Design for failure: override, drift, adversarial evasion, alert fatigue.
 - Keep realtime flows fast and interpretable; preserve user sovereignty.
-

7. Failure by Human Substitution

7.1 The missing user class: ease-seekers

Web3 stacks often optimise for:

1. builders/technical users
2. traders/speculators

A third class—ease-seeking users—is under-served: users who want safe, low-cognitive-load experiences without deep technical study or speculative intent. This class represents a large

share of mainstream adoption potential. Without UX3, systems implicitly force ease-seekers to become either experts or gamblers—or to leave.

7.2 Humans as infrastructure substitutes

In the absence of systemic mediation, ecosystems lean on:

- influencers/KOLs as risk interpreters
- educators as complexity translators
- community moderators as operational support

These roles fill infrastructural gaps (risk assessment, safe usage patterns, recovery guidance), but are inherently fragile:

- humans don't scale globally or consistently
- humans can be compromised, biased, or wrong
- humans burn out
- incentives misalign (paid promotions vs user safety)

7.3 Trust-market failure modes (formerly “InfoFi”)

When trust and safety are mediated by humans, attention becomes a proxy for safety and can be monetised. This creates predictable cascade failure modes: a compromised or mistaken trusted voice can trigger mass harmful action. The result is not isolated churn but ecosystem-level trust damage.

7.4 The search-engine analogy (scalability)

Early discovery systems relied on human-curated directories; they failed to scale. Large-scale adoption required algorithmic infrastructure to rank, filter, and protect users from spam. Web3 remains closer to the directory stage when it relies on human translation rather than systemic mediation.

7.5 Negative network effects

As user volume grows, adversarial opportunity grows: more targets, more compromised trust channels, more public failures, more ecosystem-wide distrust. Without UX3 + bounded ML mediation, growth can amplify fragility rather than value.

Builder takeaway

- If your UX depends on users reading threads and trusting intermediaries, you have a scalability and security problem.
 - Human mediation concentrates trust and creates cascade failure modes.
 - UX3 + bounded ML shifts trust from individuals to systems and reduces systemic fragility.
-

8. Ethics, Safeguards, and Limits

8.1 Overreach risk: ML as control infrastructure

ML must not become a silent control layer that overrides user intent. Boundaries:

- transparency (why intervention?)
- overrideability (user sovereignty)
- proportionality (match severity to risk)
- contestability (recourse and explanation)

8.2 False confidence from automation

ML interventions can create “if it allowed it, it must be safe” assumptions. Mitigate via:

- explicit uncertainty communication
- pairing ML with cryptographic safeguards
- graceful degradation modes when ML is uncertain or unavailable

- avoidance of absolute safety claims

8.3 Privacy and surveillance risk

Behaviour modelling can become surveillance if centralised. Principles:

- data minimisation by default
- local execution where possible
- federated/cryptographic privacy where appropriate
- user control over profiles and data retention

8.4 Centralisation-by-mediation

If a single provider controls models or data, decentralisation erodes via new choke points. Mitigate via:

- open and auditable tooling
- model diversity and user choice
- client-side execution
- robust fallback modes

8.5 Accountability

Users should be able to ask “why” and receive meaningful answers. Avoid: “the model said so.” Provide interpretable triggers and allow overrides with logging.

Builder takeaway

- ML must preserve sovereignty: transparent, overrideable, proportional, contestable.
- Avoid surveillance and choke points: prioritise local and auditable approaches.
- Design for uncertainty; never imply automation guarantees safety.

9. Implications and Careful Future Convergence

9.1 UX3 as a design discipline

Adopting UX3 changes practice across the stack:

- protocols treat state-transition UX as first-class design
- accounts and token contracts embed recovery and revocation semantics
- wallets implement bounded mediation with explainability and user control
- researchers measure intervention performance (false positives/negatives, comprehension, trust)

9.2 Possible convergence toward specialised behavioural models (carefully framed)

As builders solve UX3 problems at scale, ML systems may converge toward specialised transaction- and graph-aware behavioural models optimised for onchain constraints (latency, privacy, adversarial robustness). This paper does not propose a formal new category; it treats this as an emergent possibility driven by practical needs, not terminology.

9.3 Coordination challenges

UX3 primitives benefit from ecosystem coordination:

- standards for protected states and revocation semantics
- shared, privacy-preserving risk signals
- interoperability across wallets/frontends/protocols

This is a collective action problem: fragmented solutions dilute user trust and weaken

coverage.

9.4 Research directions

- behavioural baselines across cohorts and time
- evaluation of friction: what users tolerate and understand
- privacy-preserving modelling
- adversarial robustness and model governance
- measurable impact on loss reduction and adoption

9.5 Final framing

This paper diagnoses a present failure: Web3 often outsources complexity reduction to humans instead of building systems that absorb complexity and bound risk. UX3 and bounded ML are practical responses available now. The crucial question is not whether mediation will exist, but whether it will be designed to preserve sovereignty, privacy, and decentralisation.

Builder takeaway

- UX3 is actionable now: build primitives for safety, recovery, and bounded risk mediation.
- Specialised models may emerge, but focus on present pipelines and measurable outcomes.
- Coordination and standards matter; fragmentation is a UX failure mode.
- Privacy and sovereignty are not defaults—they must be designed.

Appendix: Builder Checklist

A. UX3 primitives

- State transition constraints: guarded transitions for high-risk actions
- Protected states: anomaly-triggered safety modes and manual review paths
- Deletion taxonomy: explicit hide / revoke / burn semantics
- Recovery paths: compromise and mistake handling mechanisms
- Revocation mechanisms: approvals, permissions, and capability rollback

B. ML integration (bounded mediation)

- Valid interventions defined: what ML can and cannot trigger
- Interactive performance: inference fits signing-flow latency expectations
- Interpretability: users can understand why friction appeared
- Failure handling: false positives, drift, evasion, degraded modes
- Sovereignty: override paths and clear consent boundaries

C. Privacy + decentralisation

- Data minimisation and retention controls
- Local computation prioritised where feasible
- Federated / cryptographic privacy considered where appropriate
- Avoid single-provider choke points: auditable, forkable approaches
- User control: inspect/export/delete behavioural profiles

D. User sovereignty

- No silent blocking: interventions are visible and contestable

- Proportional friction: severity matches risk
- Clear consequence communication before irreversible actions
- Audit trails for interventions and overrides

E. Ecosystem coordination

- Standards proposals for UX3 primitives (where relevant)
- Open tooling and documentation
- Collaboration with UX/security/ML researchers for evaluation
- Shared, privacy-preserving risk signals where appropriate

End of manuscript.