the usual neXt S.r.l. / Tholus Dome
Immersive Media Research
Technical Report / Preprint, Version 5

System Paper

From LLM Prompts to Runtime-Validated Immersive Experiences: Constraint-Guided Co-Authoring, Behavioral Validation, and Multisensory Orchestration

Dario Riccio[1] and Edoardo Guarnieri[1]

[1] *the usual neXt S.r.l. (TUN) / Tholus Dome, Castel di Sangro, AQ, Italy / Martigny, VS, Switzerland*

Correspondence should be addressed to Dario Riccio; ceo@theusualnext.com

15 July 2026

LLM-assisted authoring can accelerate immersive production, but a convincing output is not necessarily executable, behaviorally correct, or ready for public use. We present *TholusOS*, a system for co-authoring multi-user immersive experiences across dome projection, audio, lighting, participant phones, physical effects, media services, and a live scheduler. TholusOS separates creative generation from operational authority. A macro compiler derives an experience profile, temporal windows, a system-owned cue skeleton, and readiness requirements; the LLM fills only exposed creative fields. A micro compiler maps accepted moments through semantic anchors and modality lanes into synchronized runtime cues. Family-aware audits inspect progression, state transitions, action-consequence chains, service flow, and audience-perceptible outcomes. Cue timing contracts identify elements that are locked, narrowly adjustable, window-flexible, or optional, so repair remains proposal-only until review, application, and revalidation. Audience-readiness gates distinguish authoring, simulation, controlled participation, and production. We describe the implemented architecture, its uneven integration maturity, and a reproducible software snapshot, and define an evaluation protocol for authorial control, behavioral validity, multisensory timing, repair safety, and gate calibration. *Executable Experience Grammar* denotes a typed intermediate representation and transformation rules, not a formal language with proved soundness; `liveReady` is a software state rather than physical certification. This system report does not claim comparative or hardware-in-the-loop results not yet measured.

Keywords: LLM-assisted authoring; mixed-initiative authoring; behavioral validation; XR authoring; multisensory orchestration; runtime assurance; human-in-the-loop repair.

CCS Concepts: Human-centered computing—Interactive systems and tools; Human-centered computing—Mixed / augmented reality; Software and its engineering—Domain-specific languages; Computer systems organization—Embedded and cyber-physical systems.

1. Introduction

LLM-assisted authoring is moving beyond text and images toward interactive artifacts that respond to people and control external systems. That shift changes the engineering question. Authors need to know which decisions remain theirs; developers need to catch state and progression errors that survive parsing and compilation; integrators must coordinate heterogeneous media and devices; and live operators need evidence that an artifact is suitable for the audience in front of them. Similar concerns now appear in work on human-AI agency, mixed-initiative authoring, executable narrative, generated GUI applications, and pre-deployment assurance [18, 19, 10, 23, 25].

Three failures motivate this work. Prompt instructions alone cannot stop a model from changing timing, dependencies, or required interaction steps. Schema conformance and successful execution do not establish *behavioral correctness*: a system may record a choice without showing its consequence, update a score without feedback, or reach a valid state through an incoherent sequence. Studies of LLM-generated interactive applications report

the same gap between code that runs and code that behaves correctly over a sequence of user actions [23, 24]. Finally, readiness depends on context. An artifact that is useful in an editor or simulator may still be unsuitable for controlled participants or a public performance.

We study these problems through TholusOS, a production system for multi-user immersive performance. An experience may coordinate panoramic visuals, audio, lighting, participant phones, wind or low-frequency effects, generated and pre-rendered media, operator actions, and a common scheduler. The implementation does not follow a single "prompt-to-show" route. It combines two compilation levels, explicit ownership boundaries, family-aware behavioral diagnostics, cue-specific timing contracts, and graduated deployment gates. Its maturity is uneven across experience families; the paper reports that boundary rather than treating all generation routes as equivalent.

> **Thesis.** Plausibility is not executability, and executability is not deployment readiness. LLM-authored interactive experiences require system-owned structure, behavioral validation, multisensory compilation, and assurance that increases with audience exposure.

The paper makes five contributions:

1. a **constraint-guided co-authoring model** in which deterministic code owns execution-critical structure and exposes a bounded creative interface to the LLM;
2. a **hierarchical experience compiler** that separates macro progression obligations from micro multisensory choreography through semantic anchors and modality lanes;
3. a **behavioral validation layer** that examines family-specific progression and distinguishes internal state change from audience-perceptible consequence;
4. a **human-in-the-loop assurance process** combining timing-contract-preserving repair proposals, revalidation, lifecycle-sensitive severity, and audience-specific deployment gates; and
5. a **transparent implementation account and evaluation protocol** covering maturity, route provenance, reproducibility, matched baselines, ablations, hardware-in-the-loop measures, and creator/operator outcomes.

Accordingly, this manuscript is a system paper and technical report. It documents implemented mechanisms and a testable evaluation protocol, but it does not present a completed comparative benchmark, creator/operator study, audience study, or hardware-in-the-loop campaign. The current evidence establishes implementation and internal consistency, not empirical superiority.

Table 1: Practical failures, research questions, and the corresponding mechanisms studied in TholusOS.

Stakeholder	Failure in current workflows	Research question	Mechanism
Creative author	Prompt-based generation can silently rewrite structure or obscure consequential model decisions.	How can an LLM add creative variation while preserving authorial agency over execution-critical commitments?	System-owned skeleton, locked fields, bounded creative fills.
Developer / QA	Valid JSON and runnable code can still contain broken progression, state logic, or invisible consequences.	How can generated interactive behavior be validated beyond syntax and compilation?	Family-aware progression audit and perceptual-consequence checks.
Integrator	One creative moment must become synchronized cues across heterogeneous media, devices, and shared resources.	How can high-level intent be compiled into conflict-aware multisensory choreography?	Semantic anchors, modality lanes, temporal placement, conflict diagnostics.
Author / reviewer	Automatic repair may move a protected cue or change narrative meaning while fixing a local error.	How can repair restore validity without violating intent or critical timing?	Typed timing contracts, proposal-only repair, human/tool review, revalidation.
Live operator	Editor preview is often treated as evidence of production readiness.	What evidence should permit authoring, simulation, controlled participants, or public deployment?	Mode-dependent severity and audience-readiness gates.

2. Related Work

2.1. Human-AI co-creation, agency, and authorial control

The distribution of control between a person and a generative system is now a central HCI question. A scoping review of 134 top-tier HCI and CSCW papers organizes co-creation around agency configurations, control mechanisms, and interaction contexts [18]. Flowco similarly argues that real-world LLM-assisted work requires fine-grained control, explicit verification of intermediate results, and iterative refinement rather than opaque end-to-end generation [19]. HiLDe shows a complementary interaction pattern: exposing critical model decisions and allowing users to intervene locally can improve goal alignment and reduce harmful outcomes [20].

Creative systems make the same issue visible. PlayWrite moves beyond text prompting by representing embodied

actions as editable intent frames in XR [21]; WhatELSE gives authors control over a narrative possibility space before compiling it into executable events [10]; and Drama Llama combines LLM generation with authorable storylets and triggers [11]. TholusOS contributes a stricter operational boundary: execution-critical fields are absent from the model's writable interface, while creative fields remain generative. The research question is therefore not whether the AI is a "tool" or "collaborator" in general, but which commitments each party is authorized to make and how those commitments are enforced.

2.2. Behavioral correctness of generated interactive artifacts

Structured decoding can enforce grammar or schema conformance [12, 13], but interactive correctness is temporal and stateful. PlayCoder shows that GUI applications may compile and execute while silently violating event handling, state updates, or interaction logic; its evaluation therefore separates execution correctness from end-to-end playability [23]. GameGen-Verifier likewise treats generated games as state machines and verifies specification-derived behavioral keypoints through runtime state injection and bounded interaction [24]. These works establish an active shift from output validity toward behavioral verification.

TholusOS addresses a related but distinct artifact. It does not synthesize arbitrary application code or claim exhaustive runtime verification. Instead, it statically examines progression structures, required action-consequence chains, service transitions, and perceptual evidence, then compiles the result into a multi-device performance. This makes "the phone stored the choice, but the audience perceived no consequence" a first-class diagnostic rather than a subjective post-hoc observation.

2.3. Immersive authoring and multisensory orchestration

DART connected storyboard-like authoring with executable augmented-reality behavior and synchronized sensor/media replay [1]. More recent systems explicitly bring generative AI into immersive authoring. PlayWrite supports spatial co-authoring through direct manipulation [21], while OrchestrXR coordinates structured schemas and multiple agents to turn an XR study idea into a runnable Unity prototype while seeking to preserve researcher intent across stages [22]. Dome research such as Pinch-the-Sky demonstrates simultaneous multi-user interaction with omnidirectional projection [2].

Mulsemedia research studies synchronization, quality of experience, and device heterogeneity when audiovisual content is combined with additional sensory effects [14]. MPEG-V supplies adjacent representation and interoperability concepts: ISO/IEC 23005-2 covers control information and device or user capabilities, Part 3 sensory information, and Part 5 command and sensed-information formats [15, 16, 17]. TholusOS does not claim MPEG-V compliance. Its focus is the authoring-to-runtime boundary: compiling semantic moments into modality lanes, tracking readiness evidence, surfacing conflicts, and controlling when a generated artifact may reach participants.

2.4. Mixed reality performance and pre-deployment assurance

Live immersive systems also have an organizational and operational dimension. The analysis of *Desert Rain* showed that mixed reality performance depends on backstage monitoring, intervention, and carefully timed physical and virtual actions [3]. Benford and Giannachi describe such performances through interleaved physical, virtual, temporal, and social trajectories [4]. These observations motivate operator-visible diagnostics and an explicit review boundary rather than unattended AI-to-actuator control.

A parallel assurance problem appears in AI systems deployed beyond a sandbox. Recent pre-deployment work distinguishes capability benchmarks from operational authorization and proposes graduated deployment verdicts rather than a single pass/fail result [25]. TholusOS applies this principle narrowly to immersive authoring: defects can escalate from report to warning to blocker, and separate gates authorize continued authoring, simulation, controlled participants, or production. These gates remain software evidence, not safety certification or a substitute for hardware-in-the-loop testing.

2.5. PCG, mixed initiative, and partial programs

Procedural content generation (PCG) creates game content under constraints, with objectives such as playability, diversity, adaptation, or player experience [5, 6]. Tanagra is a canonical mixed-initiative example: a designer fixes constraints while the generator completes a playable level or reports that the constraints cannot be satisfied [7]. This division of labor resembles the system-owned skeleton, but the target here is an orchestrated live performance across media, devices, people, and operational fallbacks.

The architecture also resembles program sketching, where a developer supplies a partial program and a synthesizer fills unknown portions against a specification [8]. We use the analogy carefully. TholusOS is not a general program synthesizer and provides no proof of sound completion. The transferable design principle is that deterministic software owns the partial program and interface, while the LLM fills bounded semantic holes whose outputs remain subject to compilation and behavioral checks.

3. Conceptual Model and Claim Boundary

3.1. A typed intermediate representation, not a formal grammar

The term *Executable Experience Grammar* names an engineering abstraction. An experience is represented conceptually as

$$E = \langle T, K, F, A, L, C, M, G \rangle, \tag{1}$$

where T is the target and family profile; K is the system-owned structural skeleton; F is the set of AI-authored creative fills; A and L are semantic anchors and modality lanes; C is the compiled cue set; M represents media and device readiness evidence; and G represents diagnostics, policies, and promotion gates.

The key ownership constraint is that creative fills may populate only fields explicitly exposed by the skeleton:

$$\mathrm{dom}(F) \subseteq \mathrm{CreativeFields}(K), \tag{2}$$

and must not change identifiers, dependencies, required surfaces, protected timing, or other locked fields. This is a typed conceptual model and an implementation contract. We do not provide denotational or operational semantics, typing judgments, or a proof of soundness or completeness.

3.2. Readiness is distributed and staged

Readiness is not a single invariant owned by one validator. Evidence is distributed among the profile and window planners, cue compiler, media classifier, progression audit, modality-specific checks, operator-facing diagnostics, and runtime adapters. A final gate aggregates these signals for a particular use case.

The system also distinguishes two axes that are often collapsed. A lifecycle mode expresses how strictly defects should be interpreted during authoring: draft, editor preview, runtime preview, and `liveReady`. An audience gate expresses what the artifact may currently be used for: continued authoring, preview or simulation, controlled participants, or production. A `liveReady` value is a software classification, not certification that a physical dome, network, phone fleet, audio chain, or actuator installation will behave correctly.

3.3. Artifact layers and scope

The implementation separates a portable experience package, an editable manifest used by authoring tools, and a normalized runtime manifest. The pure planning and validation core is also separated from user-interface frameworks, backend services, media buses, and device protocols. This separation enables deterministic tests and alternative adapters, although it does not by itself demonstrate portability to another venue.

Multi-projector calibration, geometric warping, radiometric blending, color seams, MPCDI interchange, and atomic provenance of calibration artifacts are intentionally out of scope. They form a separate scientific problem and should be evaluated independently from AI-to-runtime compilation.

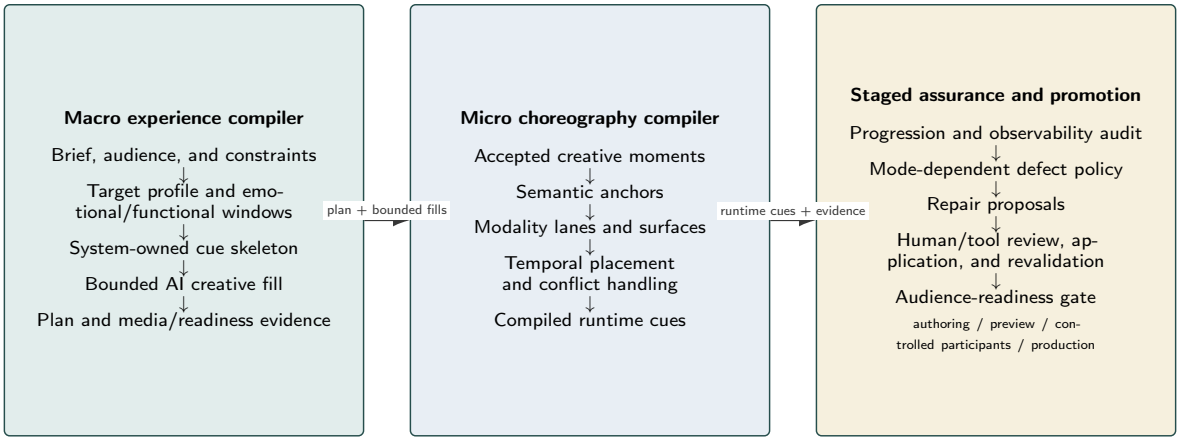

Figure 1: TholusOS separates constrained macro planning, multisensory cue compilation, and staged assurance. Applied repairs are revalidated before an artifact can advance to a higher audience-exposure level.

4. Hierarchical Compilation

4.1. Macro compilation and constrained skeletons

The macro compiler transforms a brief into a typed plan before the AI is asked for detailed content. It derives an audience and duration target, selects a family profile, partitions the experience into emotional or functional windows, instantiates required cue slots, and records media and modality requirements. The skeleton determines what must exist and where creative freedom is permitted.

This order matters. A model asked to invent a complete manifest must simultaneously solve narrative design, state progression, device routing, media selection, timing, and operator feasibility. The skeleton decomposes that problem. It can require, for example, a prompt, a bounded response window, a decision lock, a perceptible consequence, and a payoff while leaving the wording, motif, visual style, sound character, and sensory interpretation open to the AI.

Table 2: Ownership boundaries in the constrained skeleton.

Owner	Typical fields	Responsibility
System	IDs, dependencies, required slots, surface, timing class, safety/density budgets	Preserve executable structure and interfaces.
AI	wording, motif, style, media intent, sensory role, creative rationale	Supply bounded semantic and aesthetic variation.
Compiler	expansion, anchor binding, lane selection, normalization, conflict diagnostics	Convert accepted fills into runtime cues.

The fill adapter rejects unknown, duplicated, or missing required slots and rejects attempts to modify locked fields. This is more reliable than expressing the same restrictions only in natural-language instructions because the boundary is enforced after generation.

4.2. Micro compiler: from moments to multimodal choreography

A second compiler resolves creative moments into runtime choreography. Each moment is associated with a semantic anchor, such as phase entry, participant prompt, first perceptible change, reveal, feedback, service handoff, or release. The compiler then maps the moment to one or more modality lanes, including dome video, audio, light, physical effects, phones, and logic. It places cues within anchor windows, manages density and conflict, and either emits normalized runtime cues or requests repair.

This level is important because a moment is not equivalent to a cue. “The room acknowledges the choice” may require a coordinated phone close, an audio transient, a visual reveal, and a lighting transition. Conversely, a long audio bed may span multiple macro windows while remaining a single media object. Semantic anchors preserve intent across those representations, and lanes make resource and timing conflicts explicit.

The micro compiler also supports a distinction between logical progression and perceptual evidence. A state transition recorded only in logic is not counted automatically as audience-visible coverage. A decision, score change, puzzle outcome, or narrative consequence must be expressed through at least one perceptual lane when the experience design requires observable feedback. This condition prevents a technically executed but experientially silent transition.

4.3. Why two levels are preferable to one manifest pass

The macro level owns progression and coverage obligations; the micro level owns choreography and runtime placement. Keeping them separate supports localized diagnostics. A missing consequence is a structural problem; an overloaded lighting lane is a choreography problem; an unavailable video is a readiness problem. A single flat manifest validator tends to collapse these into undifferentiated errors and makes repairs more destructive because it cannot identify the level at which intent was violated.

The separation also supports alternative implementations. A family may use the full skeleton-and-fill path, a skeleton compiled by a compatibility assembler, or only window planning while integration matures. The architecture can therefore expose incomplete coverage instead of silently routing all families through a legacy generator.

5. Staged Assurance

5.1. Family-aware progression analysis

A single structural analysis is inappropriate for experiences with different progression models. The implementation selects among five diagnostic strategies:

Table 3: Progression strategies and the questions they test.

Strategy	Diagnostic focus
Dependency graph	Reachability, cycles, evidence for dependencies, terminal outcomes, progression depth.
State machine	Decisions, state continuity, consequences, and final outcomes across rounds or acts.
Action chain	Prompt, options, response, feedback, and score or state update.
Service flow	Operational phases, observable transitions, handoffs, interruptions, and fallbacks.
Window coverage	Required perceptual roles, unassigned cues, logic-only coverage, and intentional silence.

The analysis is static and heuristic. It can identify unreachable nodes or an unobserved decision consequence, but it cannot prove that a puzzle is humanly solvable, a

story is emotionally coherent, a service team can maintain pace, or a real group will understand the interaction. The defensible claim is *family-aware progression assurance*, not automatic proof of experience quality.

5.2. Mode-dependent defect escalation

Diagnostics are separated from the operational decision they cause. The same defect may be recorded without blocking a draft, shown as a warning in an editor, and promoted to a blocker before runtime preview or public use. This avoids two common failure modes: validators that block creative work too early, and permissive authoring tools that allow unresolved structural defects to reach participants.

The policy uses three outcomes: report-only, warning, and blocker. Severity depends on both the defect and the lifecycle mode. For example, a missing score update in an unfinished question sequence may remain a report during drafting but block a controlled-player session. This is a form of staged assurance: required evidence increases as the cost of failure rises.

5.3. Audience-specific readiness gates

A separate gate translates technical findings into an operator-facing authorization for four use cases: continued generation or authoring, preview or simulation, controlled participants, and production. Each level can be allowed, allowed with warnings, or blocked.

This capability ladder is stronger than a global Boolean `valid`. A manifest may be suitable for editor preview while still containing mandatory repair proposals, human QA requirements, or media readiness gaps that block participants. Conversely, a warning relevant to production may not prevent continued authoring. The gate therefore states what the team is currently authorized to do with the artifact, not whether the artifact is universally "good."

The gate remains a software gate. It cannot replace venue commissioning, physical safety review, accessibility checks, operator sign-off, or hardware-in-the-loop evidence.

5.4. Cue timing contracts

Automatic repair cannot treat every cue as fungible. The system assigns each cue one of four timing contracts, inferred from explicit metadata and conservative semantic rules.

Table 4: Cue timing contracts and permitted automatic transformations.

Contract	Automatic transformations and meaning
Locked	No retiming, split, reshape, or removal. Used for state, safety, explicit synchronization, video start/stop, timers, and decision locks.
Narrow tolerance	Only a small bounded shift; no structural repair or removal. Used for reveals, input proof, feedback, and required hero or finale moments.
Window-flexible	May be moved or reshaped inside its valid window, but not removed. Used for supporting perceptual cues whose exact instant is not semantic.
Optional	May be moved, reshaped, or removed. Used for decorative texture and other nonessential layers.

The current default tolerance for narrow cues is approximately 0.35 seconds unless more specific metadata is present. A locked or narrow cue cannot receive a structural split, reshape, or large retiming proposal. Instead, the system requests QA or human revision. This yields a general design principle: *repairability is a typed property of each temporal element, not a universal capability of the AI*.

Contract inference is itself a limitation. When metadata are incomplete, semantic inference may be overly permissive or overly conservative. Critical cues should therefore carry explicit contracts, and future evaluation should measure inference accuracy.

5.5. Proposal-only repair and revalidation

The shared repair policy does not mutate the manifest. It converts validation findings into candidate operations such as a bounded shift, fallback substitution, cue split, cue merge, optional removal, or a human-review request. The correct lifecycle is

$$\text{validate} \rightarrow \text{propose} \rightarrow \text{review/apply} \rightarrow \text{revalidate}. \quad (3)$$

Conceptually, a proposal seeks a small change $\Delta(E, E')$ while satisfying current validation rules and preserving protected contracts:

$$\min_{E'} \Delta(E, E') \quad \text{s.t.} \quad V(E') = \text{pass},\ P(E, E') = \text{true}. \quad (4)$$

Here V denotes the applicable validation and readiness checks, while P denotes preservation of locked structure and timing contracts. The implementation does not solve this as a general mathematical optimizer; the equation states the design objective used to rank and constrain proposals.

Human review is not an implementation gap to hide. It is a safety and authorship boundary. Automatic application would make the repair engine a silent co-author and could alter narrative meaning while satisfying local constraints.

5.6. Route provenance and compatibility fallbacks

The codebase contains legacy or compatibility paths that can assemble deterministic sensory coverage. Such paths are useful during migration but conflict with a strong claim that all creative fields originate from the bounded AI interface. The production routing policy disables several legacy creative paths, yet compatibility assemblers remain part of some family integrations.

A scientific evaluation must therefore log route provenance for every generated artifact and either remove deterministic creative fillers, make them unreachable in evaluated configurations, or declare and ablate them as compatibility fallbacks. Otherwise, apparent generation quality may come from hidden deterministic scaffolding rather than the claimed architecture.

6. Implementation Snapshot

6.1. Core isolation and implementation traceability

The implementation places target modeling, progression analysis, timing contracts, readiness gates, validation, and repair proposals in a reusable core package that is independent of the primary UI framework, backend service, media bus, and venue-specific light or control protocols. Runtime adapters translate the normalized manifest to concrete devices. This organization makes the core amenable to deterministic unit testing and limits the amount of platform code that must be trusted when testing authoring semantics.

The main text intentionally uses conceptual labels rather than class names. A supplementary traceability document maps those labels to representative implementation symbols, source paths, and test locations.

6.2. Integration maturity matrix

The current system does not support every experience family at the same level. The production router exposes four maturity levels across fourteen registered families.

Table 5: Current integration maturity, reported as capability rather than product family names.

Level	Families	Implemented route
Complete constrained fill	1	System-owned slots, AI fill, compilation, and downstream assurance.
Skeleton-ready	6	Cue-slot skeleton with registered compatibility assembly.
Window-plan only	3	Macro window planning with compatibility assembly.
Unsupported	4	No production generation route; legacy creative fallback disabled.

This matrix is a validity instrument, not merely project management metadata. It prevents conclusions obtained from the most mature path from being generalized to all families. The complete path should be treated as the primary case study; lower levels should be evaluated separately or reported as ongoing integration.

6.3. Snapshot and test evidence

The reproducibility snapshot refers to full repository commit `413646be`. Counts were produced from tracked files in the full checkout, excluding untracked build products and dependencies. At that snapshot the repository contained 3,248 tracked files, including 1,506 Swift files, 12 Metal files, 245 Markdown files, and 95 JSON files. The two primary test trees contained 163 Swift files and 845 declarations using the Swift Testing `@Test` attribute.

An author-run command, `swift test -package-path Packages/TholusCore`, completed with 271 tests passed and zero failed. These tests support the existence and internal consistency of the core mechanisms; they do not establish comparative generation quality, hardware reliability, audience safety, or deployment readiness. Exact counting commands, directory filters, and exclusions are included in the supplementary artifact note. A filtered review archive may produce smaller counts and must not be confused with the full commit snapshot.

7. Generalizable Design Principles and Novelty

Five design principles emerge from the implementation and are relevant to other generative products that control interactive or cyber-physical experiences.

1. Enforce ownership, do not merely request compliance. Natural-language prompts can ask a model not to alter critical fields. A structural interface can make those fields unavailable. Constraint-owned skeletons transform authorial and operational policy into an enforceable boundary.

2. Compile intent at more than one scale. Macro progression and micro choreography fail in different ways. Separating them produces better diagnostics, supports targeted repair, and allows each layer to evolve independently.

3. Replace binary validity with staged assurance. Requirements should grow from draft to preview, controlled use, and production. A system that blocks every incomplete draft is unusable; a system that treats preview success as production evidence is unsafe.

4. Type repairability. Some elements may move freely, some may move only within a tolerance, and others encode semantics or safety that must not be changed automatically. Repair contracts should be first-class data and should survive serialization, editing, and runtime compilation.

5. Treat maturity and provenance as part of the claim. A route that uses full constrained generation is not evidentially equivalent to one that relies on compatibility assembly. Generated artifacts should record the model, prompt, seed, compiler path, fallback path, validation version, and repair history.

Table 6: Difference from adjacent generation paradigms.

Paradigm	Primary guarantee	Remaining gap and contribution here
Prompt-only	Semantic plausibility	Structure and runtime constraints remain implicit; add enforced ownership and compilation.
Schema-only	Syntax and type conformance	Progression, observability, timing, and readiness remain unchecked; add domain analysis and gates.
Classical PCG	Content under game constraints	Live devices, operators, and physical exposure are often outside scope; add multisensory choreography.
Narrative AI with authored structure	Author control over narrative possibility	Physical modalities and deployment assurance may remain external; add lanes, timing contracts, and audience gates.

The novelty claim is therefore deliberately narrow. We do not claim to invent executable narrative events, schemas, PCG constraints, or multisensory metadata. We contribute their composition into a practical typed intermediate representation and staged operational process for generative, multisensory performance systems.

8. Evaluation Framework

The following six questions turn the architectural claims into measurable comparisons with prompt-only, schema-only, and component-ablated baselines.

RQ1 - Authorial control. Does a system-owned partial program reduce missing required elements, hallucinated elements, illegal modifications, and invalid dependencies while increasing creators' perceived control relative to prompt-only and schema-constrained direct generation?

RQ2 - Behavioral correctness. How accurately do progression audits detect unreachable states, broken action-consequence chains, missing outcomes, invalid service transitions, and logically executed events with no audience-perceptible consequence?

RQ3 - Multisensory orchestration. Does the semantic-anchor and modality-lane compiler reduce cue conflicts, cross-modal skew, temporal deviation, and manual correction when one creative moment spans heterogeneous devices and media?

RQ4 - Safe human-in-the-loop repair. Do reviewed repair proposals restore validity with lower creative distance and fewer timing-contract violations than regeneration or unrestricted automatic repair?

RQ5 - Deployment assurance. How accurately do lifecycle-sensitive policies and audience gates distinguish artifacts suitable for authoring, simulation, controlled participants, and production, including under injected media, device, and network failures?

RQ6 - Workflow and transfer. Does constrained co-authoring reduce time-to-valid artifact and operator workload without producing over-trust, and how do results vary across progression models, maturity levels, durations, audience sizes, and hardware profiles?

8.1. Offline benchmark

The benchmark should cross experience structure with operational stressors. Briefs should vary duration, participant count, progression model, modality set, media availability, device capability, and failure conditions. Each brief should be generated multiple times with fixed model versions and logged seeds.

Recommended conditions are: (B0) prompt-only direct manifest; (B1) schema-constrained direct manifest; (B2) system-owned skeleton and AI fill without the micro compiler; (B3) hierarchical compilation without staged assurance; and (B4) the full system. Model, temperature, token budget, retry policy, and human editing allowance must be matched. Legacy deterministic fillers must be disabled or logged as a separate condition.

Table 7: Core benchmark metrics.

Metric	Operational definition
Structural containment	Required slots completed, illegal or hallucinated elements, protected-field changes.
Behavioral validity	Reachable states, complete action-consequence chains, outcomes, and service transitions.
Perceptual consequence coverage	Required state changes with audience-observable feedback.
Anchor deviation	Absolute timing error from the required semantic anchor.
Cross-modal skew	Dispatch or onset difference between cues intended as one moment.
Lane conflict density	Resource, modality, or timing conflicts per runtime minute.
Repair safety	Revalidation yield, timing-contract violations, and proposal acceptance rate.
Intent preservation	Blind expert and creator rating of semantic, narrative, and emotional preservation.
Deployment-gate calibration	False-allow and false-block relative to expert and HIL adjudication.
Human workflow	Time-to-valid artifact, manual interventions, workload, perceived control, calibrated trust.

Ablations should remove the window planner, skeleton enforcement, semantic anchor compiler, progression audit, timing contracts, and audience gate one at a time. This is necessary to show which components contribute to readiness rather than attributing all improvement to a larger prompt or more post-processing.

8.2. Hardware-in-the-loop evaluation

Software classification must be followed by physical tests. A hardware-in-the-loop campaign should report p50 and p95 cue dispatch latency, cross-modal skew, scheduler drift, phone delivery success, actuator success, media-start failure, crash-free completion, fallback activation, and recovery time. Tests should include nominal runs and injected failures such as a missing asset, disconnected phone cohort, delayed media service, unavailable actuator, and operator pause or abort.

The gate should then be evaluated against observed runs. A strong system should rarely allow production when the physical run fails, but it should also avoid blocking artifacts that operate correctly. This creates measurable false-allow and false-block rates instead of treating the gate as self-validating.

8.3. Creator and operator study

A within-subject study with experienced creative technologists, XR authors, or AV operators should compare schema-only authoring with full constrained co-authoring. Tasks should include producing an artifact from a brief, resolving injected defects, preparing a controlled run, and responding to a simulated runtime incident. Measures should include time to valid artifact, residual defects, accepted and rejected repair proposals, manual interventions, NASA-TLX workload [26], perceived authorial control, trust calibration, decision ownership, and qualitative explanations of why users accepted or overrode the system. The study should distinguish actual structural control from a merely reassuring interface, consistent with agency-centered evaluation in human-AI co-creation [18, 20].

An audience study is secondary but useful. Presence, social presence, coherence, comfort, emotional alignment, comprehension of interaction consequences, and simulator sickness [27] should be reported separately from technical readiness. Better runtime reliability may improve experience by reducing visible failures without increasing aesthetic originality; the analysis should preserve that distinction.

9. Discussion

The architectural lesson is broader than immersive domes. AI is increasingly asked to control workflows with external consequences: museum rooms, location-based XR, theme-park scenes, smart environments, live broadcasts, robotic installations, and collaborative simulations. In each case, a direct prompt-to-action design gives the model both creative authority and operational authority. The proposed separation gives it the former through a bounded interface while retaining the latter in deterministic structure, compilers, policies, and human approval.

The approach introduces costs. System-owned skeletons require domain modeling and can constrain unexpected creative forms. Family-aware audits require multiple strategies rather than one universal validator. Timing contracts and gates add metadata and workflow friction. The

staged design is intended to make that friction proportional: early drafts remain permissive, while higher-risk exposure requires stronger evidence.

The implementation also shows why maturity must be visible. A clean architecture diagram can imply uniformity that a real production codebase does not possess. Reporting route levels and compatibility paths allows readers to distinguish the general design from the subset currently realized end-to-end. That distinction is necessary for meaningful replication and ablation.

Finally, the term *executable* should be used with care. The runtime manifest is executable by the software, but deployment depends on media, hardware, network, operators, venue calibration, and audience conditions. The audience gate is best understood as a software capability ladder whose upper levels require external evidence.

10. Threats to Validity, Ethics, and Limitations

No comparative results. The present manuscript does not demonstrate that the architecture outperforms matched baselines. The evaluation protocol is prospective. Unit tests establish core behavior, not scientific effectiveness.

Static analysis limits. Progression audits use graphs, state machines, action chains, service-flow proxies, and coverage checks. They do not prove human solvability, narrative quality, emotional effect, operator feasibility, or accessibility.

Inferred timing contracts. Some contracts are inferred from actions, categories, priorities, anchors, and metadata. Incorrect or incomplete metadata can lead to permissive or conservative repair behavior. Critical contracts should be explicit and independently reviewed.

Uneven maturity. Only one registered family currently exercises the complete constrained-fill route. Results from that route cannot be generalized to skeleton-ready, window-only, or unsupported families without separate evidence.

Compatibility contamination. Deterministic creative fillers or compatibility assemblers can inflate apparent generation performance. Route provenance must be logged, and such paths must be excluded, isolated, or ablated.

Single system and venue. The architecture was derived from one production codebase and dome ecosystem. Adapter independence is not proof of transfer to another venue or platform.

Model and media dependence. Results may vary with language model, media generator, prompt, seed, moderation policy, latency, and provider availability. Reproducible studies should archive model identifiers, prompts, generated artifacts, and scoring procedures where licensing permits.

Safety and accessibility. Software gates are not safety certification. Public deployment requires limits and review for light, strobing, volume, low-frequency energy, airflow, motion imagery, photosensitivity, anxiety triggers, and accessibility. Operators need emergency stop and override. Low-sensory variants should be authored as first-class profiles rather than improvised after generation.

Privacy. Participant phones and operational logs can expose personal or behavioral data. Studies and deployments should minimize identifiers, separate safety logs from analytics, communicate consent clearly, and define retention and deletion policies.

11. Conclusion

TholusOS is best understood not as a prompt that produces a show, but as a hierarchical compiler and staged assurance system. Deterministic structure owns progression, dependencies, surfaces, and protected timing; AI supplies bounded creative intent; a micro compiler maps that intent through semantic anchors and modality lanes; and diagnostics, proposal-only repair, revalidation, and audience-specific gates increase assurance before deployment. The claims remain deliberately bounded: the grammar is a typed intermediate representation rather than a proved formal language, and `liveReady` is a software state rather than physical certification. These boundaries are the mechanisms by which generative creativity can coexist with authorial and operational responsibility.

References

[1] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter, "DART: a toolkit for rapid design exploration of augmented reality experiences," in *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, 2004, pp. 197–206. doi: 10.1145/1029632.1029669.

[2] H. Benko and A. D. Wilson, "Multi-point interactions with immersive omnidirectional visualizations in a dome," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, 2010, pp. 19–28. doi: 10.1145/1936652.1936657.

[3] B. Koleva, I. Taylor, S. Benford, M. Fraser, C. Greenhalgh, H. Schnädelbach, D. vom Lehn, C. Heath, J. Row-Farr, and M. Adams, "Orchestrating a mixed reality performance," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2001, pp. 38–45. doi: 10.1145/365024.365033.

[4] S. Benford and G. Giannachi, *Performing Mixed Reality*. Cambridge, MA: MIT Press, 2011.

[5] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. Cham: Springer, 2016. doi: 10.1007/978-3-319-42716-4.

[6] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011. doi: 10.1109/T-AFFC.2011.6.

[7] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: a mixed-initiative level design tool," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 2010, pp. 209–216. doi: 10.1145/1822348.1822376.

[8] A. Solar-Lezama, L. Tancau, R. Bodik, V. Saraswat, and S. A. Seshia, "Combinatorial sketching for finite programs," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2006, pp. 404–415. doi: 10.1145/1168857.1168907.

[9] M. Mateas and A. Stern, "Structuring content in the Façade interactive drama architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 1, no. 1, 2005, pp. 93–98. doi: 10.1609/aiide.v1i1.18722.

[10] Z. Lu, Q. Zhou, and Y. Wang, "WhatELSE: shaping narrative spaces at configurable level of abstraction for AI-bridged interactive storytelling," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025, article 598. doi: 10.1145/3706598.3713363.

[11] Y. Sun, P. J. Wang, J. J. Y. Chung, M. Roemmele, T. Kim, and M. Kreminski, "Drama Llama: an LLM-powered storylets framework for authorable responsiveness in interactive narrative," arXiv:2501.09099, 2025.

[12] S. Geng, M. Josifoski, M. Peyrard, and R. West, "Grammar-constrained decoding for structured NLP tasks without finetuning," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 10932–10952. doi: 10.18653/v1/2023.emnlp-main.674.

[13] S. Geng, H. Cooper, M. Moskal, S. Jenkins, J. Berman, N. Ranchin, R. West, E. Horvitz, and H. Nori, "Generating structured outputs from language models: benchmark and studies," arXiv:2501.10868, 2025.

[14] G. Ghinea, C. Timmerer, W. Lin, and S. R. Gulliver, "Mulsemedia: state-of-the-art, perspectives, and challenges," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 11, no. 1s, article 17, 2014. doi: 10.1145/2617994.

[15] ISO/IEC, *ISO/IEC 23005-2:2018, Information Technology – Media Context and Control – Part 2: Control Information*, 4th ed., 2018.

[16] ISO/IEC, *ISO/IEC 23005-3:2019, Information Technology – Media Context and Control – Part 3: Sensory Information*, 4th ed., 2019.

[17] ISO/IEC, *ISO/IEC 23005-5:2019, Information Technology – Media Context and Control – Part 5: Data Formats for Interaction Devices*, 4th ed., 2019.

[18] S. Zhang, H. Wang, and X. Yi, "Exploring collaboration patterns and strategies in human-AI co-creation through the lens of agency: a scoping review of the top-tier HCI literature," *Proceedings of the ACM on Human-Computer Interaction*, vol. 9, no. 7, article CSCW413, pp. 1–43, 2025. doi: 10.1145/3757594.

[19] S. N. Freund, B. Simon, E. D. Berger, and E. Jun, "Flowco: mixed-initiative authoring of reliable end-to-end data analysis workflows," in *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, 2025, article 182, pp. 1–20. doi: 10.1145/3746059.3747636.

[20] E. A. González, R. Rothkopf, S. Lerner, and N. Polikarpova, "HiLDe: intentional code generation via human-in-the-loop decoding," in *2025 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2025, pp. 222–233. arXiv:2505.22906.

[21] E. K. Tütüngçü, Q. Zhou, F. Brudy, G. W. Fitzmaurice, and F. Anderson, "PlayWrite: a multimodal system for AI supported narrative co-authoring through play in XR," in *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 2026, article 1032, pp. 1–26. doi: 10.1145/3772318.3791159.

[22] S. Liao, C. Zhu, K. Ramani, and V. Popescu, "OrchestrXR: a multi-agent system for idea-to-prototype XR study authoring," arXiv:2607.01588, 2026.

[23] Z. Peng, W. Tao, X. Yin, C. Ying, Y. Luo, and Y. Guo, "PlayCoder: making LLM-generated GUI code playable," arXiv:2604.19742, 2026.

[24] C. Jia, R. Wan, T. Sun, W. Tan, B. Wan, Y. Tong, G. Sheng, and H. Xu, "GameGen-Verifier: parallel keypoint-based verification for LLM-generated games via runtime state injection," arXiv:2605.07442, 2026.

[25] T. L. Tuan and A. Sanyal, "Toward pre-deployment assurance for enterprise AI agents: ontology-grounded simulation and trust certification," arXiv:2606.04037, 2026.

[26] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): results of empirical and theoretical research," in *Human Mental Workload*, P. A. Hancock and N. Meshkati, Eds. Amsterdam: North-Holland, 1988, pp. 139–183.

[27] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: an enhanced method for quantifying simulator sickness," *The International Journal of Aviation Psychology*, vol. 3, no. 3, pp. 203–220, 1993. doi: 10.1207/s15327108ijap0303_3.