applied sciences

MDPI

Article

BreastfeedingSupport Training Using Guard-Mediated LLM-VR Scene Control: Implementation and Technical Evaluation

Nobuyoshi Hashimoto [1,*] and Kumiko Iwatani [2]

[1] Department of Mechanical Systems Engineering, University of Shiga Prefecture, Hikone 522-8533, Shiga, Japan
[2] Faculty of Nursing & Social Welfare Sciences, Fukui Prefectural University, Eiheiji 910-1195, Fukui, Japan; kiwatani@g.fpu.ac.jp
* Correspondence: hashimoto.n@mech.usp.ac.jp

Abstract

While variable-latency AI dialogue processing proceeds asynchronously, the VR state continues to be updated frame by frame. In this study, we implemented "guard-mediated LLM-VR scene control" for breastfeeding support training, in which LLM-derived ActionClass and PhaseCandidate are not used to update the phase directly, but are instead provided as inputs to phase-specific deterministic guards together with the VR state and the ActionClass-mediated mother-avatar state on the Unity side. We conducted human-in-the-loop sessions with 20 nursing students, post-study guard/Turn-ID tests, a 2-expert semantic evaluation of 639 cases, and state-binding and local guard sensitivity analyses. There were no failures preventing session continuation, and the median of the participant-specific median latencies was 2.61 s. Phase guards matched a separately implemented specification-based oracle in 1600 out of 1600 cases, and the Turn-ID gate eliminated cross-turn stale results. LLM-expert strict agreement was 74.5%/77.8% for ActionClass and 16.9%/35.2% for PhaseCandidate. The local outcome divergence in cases where semantic labels differed between the two experts was 85.2% in Phase 1, 1.9% in Phase 2, and 56.8% in Phase 3; the impact of semantic variability differed across phases. In particular, the current topology, which uses PhaseCandidate as a standalone transition-enabling disjunct, requires reevaluation. This study is a technical pilot and mechanism-specific characterization; it does not evaluate educational effectiveness, clinical efficacy or safety, or superiority over alternative architectures.

Keywords: large language models; virtual reality; nursing education; breastfeeding support; asynchronous interaction; semantic classification; deterministic guards; human-in-the-loop

Check for updates

Academic Editor: Douglas O'Shaughnessy

Received: 21 July 2026
Revised: 23 August 2026
Accepted: 26 August 2026
Published: 28 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

When free-form dialogue with an avatar and physical object manipulation are performed concurrently in VR training, the two processes evolve on different time bases. While STT processing, LLM-based response generation, TTS processing, and post-response semantic analysis proceed asynchronously with variable delays, object grasping, region detection, state updates, and rendering continue on a per-frame basis. This integration requires mapping delayed results to turns, discarding stale results, linking them to the VR state at the time of application, restricting phase update permissions, and maintaining a traceable log after execution.

We addressed this engineering problem in the context of breastfeeding support training, a setting that combines dialogue with physically embodied assistance within the same

scene. Simulation and VR have been widely explored in nursing and midwifery education [1–4]. In medical communication education, virtual patients are used [5,6], and in recent years, LLM-based virtual patients and generative AI-enhanced immersive training systems have also been reported [7–11]. In the field of breastfeeding support, in addition to VR simulations that address the breastfeeding experience, high-fidelity lactation simulations for health-professional breastfeeding education have been developed, and educational evaluations—including assessments of self-efficacy and reflection—targeted at midwifery students have been reported [12–15]. Breastfeeding support involves not only explanations and emotional support for the mother but also newborn handling and positioning, breastfeeding posture, and post-feeding safety.

On the other hand, previous research has already demonstrated that integrating LLMs into VR—specifically, using LLM-derived evaluation to modify avatar behavior and task progression, or converting natural language into structured VR actions—is feasible. Therefore, the objective of this study is not merely the simple integration of LLMs and VR, but rather to explicitly evaluate how uncertain dialogue-derived semantic evidence is integrated with the VR state—which is updated by learner operations and physics—and the mother-avatar state—which is updated in a limited manner by ActionClass—and to what extent that variability propagates to local scene-control decisions.

Research on spoken dialogue has focused on incremental dialogue processing [16], turn-taking control [17], and embodied conversational agent architectures that separate multimodal behaviors [18,19]. Testing of finite-state machines and record/replay of execution traces have been used to track state transitions and support regression testing [20–22]. In this study, we apply these existing concepts to a breastfeeding support training system that integrates variably delayed LLM-derived semantic labels, human-in-the-loop VR interactions, VR scene states, and multiple logs, and evaluate its operational behavior and local control sensitivity.

Turn-ID matching, "latest-wins" policy input buffering, finite-state machines, apply-time state binding, and frame-indexed logging are existing concepts; this study does not claim that these individual mechanisms themselves are novel. In this study, we integrate LLM-derived semantic evidence with the physical VR state, the avatar state (which is updated only under limited conditions), and Unity-side phase-specific guards, and evaluate their operational behavior, semantic concordance, and local control sensitivity.

In this implementation, the ActionClass and PhaseCandidate obtained from post-response analysis are retained as LLM-derived semantic inputs; they are not directly assigned to phase variables but are treated as inputs to Unity's phase-specific guards. PhaseCandidate is used as an input to the phase guard, while ActionClass, in addition to serving as an input to the phase guard, can update the mother-avatar state within a predefined range. On the other hand, the LLM-derived labels themselves do not directly commit phase transitions; the final authority for phase transitions rests with Unity's phase-specific deterministic guards.

Therefore, in this paper, rather than focusing on the architectural components themselves, we separately evaluate (i) the implementation behavior of each control mechanism, (ii) the concordance between LLM-derived semantic labels and expert references, and (iii) the extent to which semantic-label variability propagates to local phase decisions under the observed VR state.

In this study, we distinguished between the operational behavior observed during the original participant sessions and the mechanism-specific, semantic, and control-sensitivity evaluations conducted after the participant study concluded, and established the following research questions.

- **RQ1: Operational Behavior:** In the 20 human-in-the-loop sessions, what kind of operational behavior did the system exhibit, and to what extent were key logs acquired and retrospectively reconciled?
- **RQ2: Software latency**: What is the text-commit-to-playback-command latency (the time from when the finalized utterance text is committed to the Unity conversation state until the audio playback command is issued to the patient avatar), and what is the distribution tail of this latency?
- **RQ3: Mechanism-specific control behavior**
  - **RQ3a:** Does the phase-guard implementation match a separately implemented specification-based oracle on a predefined discrete test grid?
  - **RQ3b:** Can the Turn-ID gate eliminate artificially injected cross-turn stale asynchronous results, and what is the scope of this guarantee?
  - **RQ3c:** In observed traces, to what extent does the difference between the request-proxy state and the apply-time state affect the local next-phase outcome?
- **RQ4: Semantic concordance and local control sensitivity**
  - **RQ4a:** To what extent do LLM-derived ActionClass and PhaseCandidate align with reference annotations assigned independently by two domain experts?
  - **RQ4b:** If we fix the observed VR state and replace the LLM-derived labels with each expert's reference labels, to what extent does the local next-phase outcome change?
  - **RQ4c:** When semantic labels differ between experts, to what extent do expert-specific local phase decisions differ, and how does this divergence vary by phase?

The primary engineering contribution of this research is the explicit definition of a domain-specific operational implementation that limits the authority of LLM-derived semantic evidence and integrates it under a continuously updated physical VR state and phase-specific deterministic guards.

Second, we evaluated the behavior and assurance boundaries of key control mechanisms on a mechanism-specific basis using guard oracle testing, Turn-ID fault injection, observed-trace state-binding sensitivity, and revised logging integration tests.

Third, we compared all 639 semantic outputs with those of two expert references and, by performing a local guard sensitivity analysis with a fixed observed apply-frame state, separately characterized semantic concordance and local scene-control sensitivity. This analysis showed that, under the observed state distribution, the extent to which semantic variability is reflected in local phase decisions differs across phases, consistent with the implemented phase-specific guard structure.

This study is a technical human-in-the-loop pilot and does not evaluate educational or clinical effectiveness, or the superiority of this architecture over existing methods.

2. Related Work and Positioning

2.1. LLM-Enhanced VR for Training and Healthcare

Efforts to integrate LLMs and generative AI into immersive environments are rapidly expanding in fields such as education, language learning, and medical training. ELLMA-T placed an embodied LLM tutor within social VR to provide language assessment, role-play, and feedback as a system-managed learning workflow [23]. El Hajji et al. present a VR tutoring architecture that combines STT, TTS, and LLM-based NPCs with hand tracking, gaze tracking, and haptic interaction [24]. Furthermore, Geris and Alce integrated the GPT-4o Realtime API into Unity VR to implement voice-native task tutoring [25].

Similar integration is also progressing in medical and nursing education. He et al. applied generative AI-enhanced immersive VR patient encounters to nursing students,

enabling free oral interaction [10]. Furthermore, in a VR simulation of school-nursing health consultations, Geng et al. have implemented trust-based interaction control, in which an LLM evaluates learners' verbal strategies and dynamically adjusts the virtual child's trust level and behavior [11]. Therefore, reflecting LLM-derived evaluations in VR avatar states or scenario progression is not, in itself, a novel contribution of this study.

Furthermore, Mikkonen et al. presented a hybrid AI-XR architecture that combines conversational generative AI with deterministic safety mechanisms in VirtualHealEd, a system designed for healthcare communication training [26]. In particular, by separating life-critical medical instructions into deterministic dialogue paths, they distinguish between AI flexibility and safety-critical control. While this design philosophy is similar to the "bounded AI authority" concept in this study, our research focuses on evaluating, at the turn level, both the expert concordance of the LLM-derived semantic labels themselves and the extent to which their variability propagates to local phase decisions under the observed VR state.

2.2. LLM-Mediated XR Action and Scene Control

There is also research on converting natural language or multimodal input into XR scene control. Voice2Action hierarchically converts voice/text commands into action and entity representations and executes VR actions using environment feedback [27]. In GesPrompt, speech and co-speech gestures are interpreted by an LLM and converted into function calls and parameters that the XR backend can execute [28]. These are representative approaches for connecting LLMs to VR interaction by converting language-derived intent into structured, executable representations.

Regarding more general XR-AI coordination, INTELLECTUM presents a conceptual reference architecture that includes asynchronous agent inference, event semantics, deterministic state evolution, and an event-store-based authoritative state [29]. Thus, the principle of separating asynchronous AI inference from deterministic XR-side state authority is not unique to this study.

2.3. Positioning of the Present Study

Based on the above prior research, this study does not position LLM-VR integration, language-to-action conversion, deterministic state control, Turn-ID, or logging mechanisms themselves as novel contributions.

The characteristic of the control pattern addressed in this study lies in the fact that it does not treat LLM output as a direct scene command or phase command, but rather maintains it as predefined discrete semantic evidence—namely, `ActionClass` and `PhaseCandidate`—and combines it during guard evaluation with the VR state (which is updated by user input and physics) and the mother-avatar state (which is updated only to a limited extent by `ActionClass`). While `ActionClass` can update the restricted mother-avatar state, the final authority to commit a phase transition is held by Unity-side phase-specific guards.

Furthermore, this study does not merely describe this authority boundary at the implementation level; instead, it empirically characterizes how strongly semantic variability is reflected in local control decisions across phases, using full-sample expert reference evaluation and trace-conditioned local guard sensitivity. Therefore, the main contributions of this paper lie in the operational implementation of domain-specific guard-mediated semantic control and an evaluation that combines mechanism-specific and semantic-control evidence. Table 1 shows this positioning and a feature-level comparison with representative LLM/AI–XR systems.

Table 1. Feature-level comparison of representative LLM/AI-XR systems and the present study.

Study/System	AI Role/XR Coupling	Final Control Authority	Async-Result Handling	Evaluation Focus
Voice2Action [27]	Action/entity representations; environment-feedback coupling	VR action-execution pipeline	NR *	Language-to-action execution and interaction efficiency
GesPrompt [28]	Function calls/parameters from speech and co-speech gestures	XR backend executes generated functions	NR *	Multimodal prompting and XR interaction performance
ELLMA-T [23]	Embodied tutor responses, assessment, role-play, and feedback in social VR	System-managed learning workflow	LLM interaction supported; exclusion of stale results not a primary target	Language-learning interaction and learner experience
El Hajji et al. [24]	Tutoring dialogue coupled with speech, gaze, hand tracking, and haptics	Tutoring application logic	NR *	Architecture and multimodal immersive tutoring
He et al. [10]	Generative AI patient dialogue in immersive VR	Simulation/application logic	Not a primary mechanism-specific evaluation target	Nursing student experience in generative AI-enhanced VR
Geng et al. [11]	LLM evaluation of learner verbal strategies; trust/behavior update	VR scenario/trust-based interaction logic	NR *	School nursing training and dynamic virtual patient behavior
VirtualHealEd [26]	Conversational AI separated from deterministic safety-critical dialogue	Hybrid AI-XR logic with deterministic safety mechanisms	Hybrid separation described; turn-level stale-result handling not a primary target	Safe healthcare communication and interaction training
INTELLECTUM [29]	Agent/AI events coupled with XR state through an event-oriented architecture	Authoritative event/state layer	Explicit asynchronous inference and deterministic state evolution	Reference architecture for hybrid AR-VR/AI systems
Present study	ActionClass/PhaseCandidate combined with VR state and ActionClass-mediated mother-avatar state	Unity-side phase-specific deterministic guards	Turn-ID stale-result exclusion; apply-time state binding evaluated separately	Human-in-the-loop operation; mechanism tests; expert concordance; local guard sensitivity

Note. NR * = not reported as an explicit stale-result control mechanism in the cited publication. "Not reported" indicates only that the corresponding feature was not explicitly described in the cited publication and does not necessarily imply that the feature was absent from the implementation.

3. System and Study Design

3.1. Study Scope and Engineering Evaluation Design

This study is an observational technical pilot of a human-in-the-loop system that integrates asynchronous AI processing with frame-indexed VR scene processing. In these single-group, single-session participant sessions, we collected logs saved during execution, processing timestamps, final-state snapshots, and exploratory user reports. The participant sessions themselves did not incorporate any comparison conditions or mechanism-specific ablation/fault injection. Separately, after the participant study concluded, we conducted a post-study reanalysis using the saved logs, as well as software evaluations of the guard, Turn-ID gate, state-binding policy, and revised logging implementation. These post-study evaluations were treated separately from the observational results of the 20 participant sessions. This study did not aim to test hypotheses regarding educational effectiveness or clinical effectiveness.

3.2. Runtime Architecture and Frame-Snapshot Model

The key design elements of this system and the scope of evidence obtained in this study are shown in Table 2.

Table 2. Key design elements and scope of evidence obtained in this study.

Design Element	Implementation Details	Quantities Observed in This Study	Outside the Scope of This Study
Turn-ID gate	Verify the source turn ID against the active turn ID to reject cross-turn stale results	Original-session logs and post-study stale-result fault injection	Actual occurrence rate of stale callbacks in original sessions, same-turn deduplication, and general fault tolerance
Apply-time state binding	Use the current VR state at the time of semantic result application for guard evaluation	Comparison of request-proxy and apply snapshots across 632 aligned turns	Causal effects on future trajectories; general superiority over earlier-state binding
Unity phase guards	Evaluation of predefined phase-transition conditions based on predefined discrete semantic labels, VR states updated by learner actions and physics, and ActionClass-mediated mother-avatar states	Local guard sensitivity across a 1600-case separately implemented oracle test and 526 guard-active turns	formal verification, clinical safety, and generalization to the entire unobserved state space
LLM-derived semantic labels	using ActionClass and PhaseCandidate as predefined discrete evidence channels	All 639 reference annotations assigned independently by 2 experts	adjudicated gold standard, clinical appropriateness of generated responses
Logging and provenance	post hoc alignment of original logs and a post-study revised common-ID schema	alignment of 19 out of 20 original sessions, revised schema integration tests	retroactive common-ID traceability of original logs

We refer to a configuration that coordinates asynchronous events with variable completion times and frame-indexed processing on the Unity main thread within the same execution system as a dual-time-base architecture. This term is distinct from continuous-discrete hybrid dynamical systems in control theory. Figure 1 illustrates the functional architecture and information flow. Here, 10,000 and 20,000 are relative order values assigned to scripts in Unity's Script Execution Order, with scripts having smaller values executed first. These values do not represent time, frame numbers, the number of executions, or absolute processing positions.

The system consists of the STT Integration Unit (A), the Response Generation Unit (B), the TTS Integration Unit (C), the Post-Response Analysis Unit (D), the Scene State Transition Control Unit (E), the Progress and Log Recording Unit (F), and the VR Input and State Update Unit (G). Units A–D execute the corresponding cloud API requests asynchronously. E–G perform input acquisition, state updates, guard evaluations, and logging within Unity's frame processing. A post-response analysis result is reflected in the current callback values—and `GestureID` is updated—only if it passes turn-ID matching. Thereafter, the Scene State Transition Control Unit captures it as a held semantic label in the next execution of `TrainingFlowController.LateUpdate` and combines it with the current VR state. `GestureID` is a legacy variable name retained from an earlier implementation and functions as a dialogue-state version counter that notifies the Unity side when the callback values produced by dialogue processing or post-response analysis have been updated.

We refer to the process of evaluating discrete states and logical conditions on the Unity side and monotonically progressing through phases 1 through 4 as discrete scene control. The LLM is connected as an external subsystem that generates `ActionClass` and `PhaseCandidate`; accepted values are then provided as inputs to the predefined Unity-side

guards. In Phase 1→2 and Phase 3→4, one of the disjunctive conditions may hold based on these labels alone.

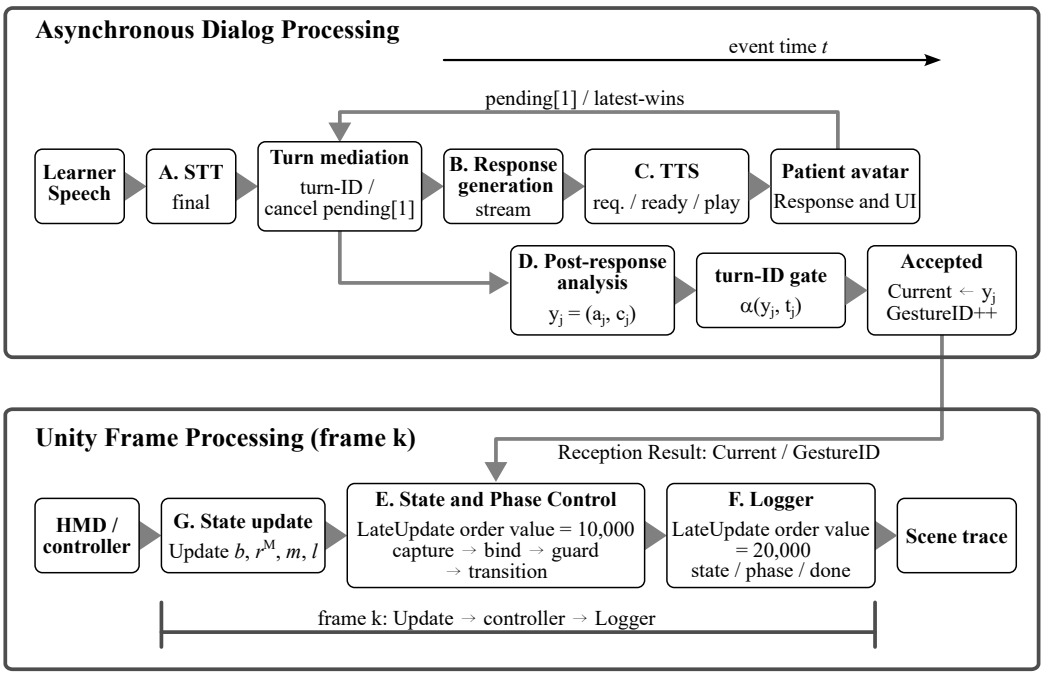


Figure 1. Dual-time-base architecture of the proposed system. The upper section represents the asynchronous event plane, which completes with variable delays. The lower section represents the Unity frame-indexed scene-processing plane, ordered by Unity frame number k. Only post-response analysis results accepted via turn-ID matching are reflected in the current callback values, and `GestureID` is incremented. Within the same frame, following a normal `Update`, processing proceeds in the order of the `TrainingFlowController` (with a Script Execution Order setting of 10,000) and the `TrainingFlowLogger` (with a setting of 20,000). The logger evaluates the runtime frame snapshot at execution time and records only the differential events generated relative to the previous frame, each tagged with the Unity frame number.

Let the Unity frame number be k. The runtime frame snapshot evaluated when `TrainingFlowLogger` executes is defined as

$$\mathcal{S}_k^{\mathrm{rt}} = \left(\phi_k,\, b_k,\, r_k^{\mathrm{M}},\, m_k,\, \ell_k,\, \bar{a}_k,\, \bar{c}_k,\, d_k\right). \tag{1}$$

Equation (1) represents a conceptual state vector that summarizes the state, input, and derived values observed within the same frame. Rather than saving the entire vector in every frame, the training-progress log records, for selected components, only the events generated from the difference between the current and previous frame snapshots. The set of events logged in frame k is expressed as

$$\mathcal{E}_k^{\log} = \Delta_{\log}\left(\mathcal{S}_k^{\mathrm{rt}}, \mathcal{S}_{k-1}^{\mathrm{rt}}\right).$$

Here, $\Delta_{\log}$ maps changes in the logged state components, phase transitions, and the rising edge of the completion predicate to the corresponding log events. This representation captures only value relationships observable in the saved logs; the atomicity of shared variable writes, inter-thread memory visibility, idempotency against duplicate callbacks, and parallel execution characteristics were addressed as separate evaluation topics.

Let t_j denote the asynchronous reception-event time of the post-response analysis result y_j. The acceptance condition is that the source turn ID attached to the result matches the active turn ID at time t_j. This is expressed as

$$\alpha(y_j, t_j) = \begin{cases} 1, & \text{sourceTurnID}(y_j) = \text{activeTurnID}(t_j), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Results for which $\alpha = 0$ are not applied because they are delayed results from a stale turn; consequently, the current callback values A and C, and `GestureID`, are not updated. Only results for which $\alpha = 1$ are reflected in the current callback values, and `GestureID` is incremented.

Let G_k denote the `GestureID` observed when `TrainingFlowController.LateUpdate` executes in frame k, and let A_k and C_k denote the current classification values. The acceptance test is performed on the asynchronous event-time axis t, whereas capture of the accepted values, apply-time state binding, and finite-state guard evaluation are performed on the Unity frame-number axis k.

The held semantic labels are given by

$$(\bar{a}_k, \bar{c}_k) = \begin{cases} (A_k, C_k), & G_k \neq G_{k-1}, \\ (\bar{a}_{k-1}, \bar{c}_{k-1}), & G_k = G_{k-1}. \end{cases} \tag{3}$$

Therefore, $\bar{a}_k$ and $\bar{c}_k$ are discrete states that retain their values until the next change in `GestureID`.

Within the same frame k, due to Unity's event function order, the normal `Update` first performs a state update on the inputs and states related to b_k, r_k^{M}, m_k, and ℓ_k. In the subsequent `LateUpdate` phase, the `TrainingFlowController`—set to a Script Execution Order value of 10,000—detects a change in `GestureID` and captures and holds $\bar{a}_k$ and $\bar{c}_k$. Next, these are combined with the latest b_k, r_k^{M}, m_k, and ℓ_k to perform the finite-state guard evaluation for the current phase. If the guard holds, a transition is made from ϕ_{k-1} to ϕ_k at most once, and d_k is evaluated using the post-transition state ϕ_k. Afterward, the `TrainingFlowLogger`—set to a value of 20,000—records the difference between the runtime frame snapshot $\mathcal{S}_k^{\mathrm{rt}}$ at the time the Logger is executed and the snapshot of the previous frame.

During data collection, `allowPhase1To2ByAction` was set to `true`.

The `allowPhase3To4ByAction` option was also set to `true`. The runtime phase-transition conditions evaluated at frame k are based on the logical structure of the source code,

$$\begin{aligned} g_{1\to2}^{\mathrm{rt}}(k) &= [\phi_{k-1} = 1] \wedge \Big([\bar{c}_k \geq 2] \vee [b_k = \text{HeldByStudent}] \vee [r_k^{\mathrm{M}} = 1] \\ &\qquad \vee [b_k = \text{InMotherArms}] \vee [m_k = 1] \vee [\bar{a}_k \in \{1, 2\}] \Big), \\ g_{2\to3}^{\mathrm{rt}}(k) &= [\phi_{k-1} = 2] \wedge [b_k = \text{InMotherArms}] \wedge [m_k = 1], \\ g_{3\to4}^{\mathrm{rt}}(k) &= [\phi_{k-1} = 3] \wedge \Big([\bar{c}_k \geq 4] \vee [\bar{a}_k = 3] \vee [m_k = 0] \vee [b_k = \text{InCot}] \Big). \end{aligned} \tag{4}$$

A `switch` statement processes only one case corresponding to the current phase. Therefore, at most one phase transition is allowed per Unity frame.

The mother avatar's breastfeeding-preparation state m_k is defined as

$$m_k = \begin{cases} 1, & A_k = 1, \\ 0, & A_k = 3, \\ m_{k-1}, & \text{otherwise} \end{cases} \tag{5}$$

and is retained across frames. `ActionClass` = 3 can contribute to the Phase 3→4 transition through both the direct condition $\bar{a}_k = 3$ and the state-mediated pathway through $m_k = 0$. Since both paths originate from the same input, they were treated as a single guard source.

The current system-defined completion predicate and the completion-entry event recorded in the log are defined separately as follows:

$$d_k = [\phi_k \geq 4] \wedge [b_k = \text{InCot}] \wedge [m_k = 0], \tag{6}$$

$$e_k^{\text{done}} = [d_{k-1} = 0] \wedge [d_k = 1] \tag{7}$$

`TRAINING_DONE` is recorded only at the rising edge when $e_k^{\text{done}} = 1$. Therefore, if the system re-enters the state after having once departed from the completion condition, it may be recorded multiple times within the same session. The correspondence between states, frame inputs, derived predicates, execution order, and phase-transition conditions is also shown in Supplementary Tables S17 and S18.

3.3. Dialogue Coordination and Runtime Configuration

We assigned a turn ID to each utterance and canceled the previous turn when starting a new turn. In the first-sentence streaming callback and the post-response analysis callback, we performed turn-ID matching with `activeTurnId`. Results that did not match were not applied.

During data collection, `deferNewTurnUntilTtsEnd` was set to `true`.

The `keepLatestPendingInputWhileSpeaking` option was also set to `true`. If a new STT-final result arrived while the patient avatar's speech was playing, only one input was held in `PendingInput`. If multiple inputs were confirmed during playback, we used a latest-wins policy, where the input held first was replaced by the latest input. After the playback-completion event or a text-only fallback, the held input was started as the next turn. The number of pending-input overwrites was not recorded in the log.

The state at utterance time was not saved; instead, the post-response analysis result was combined with the current VR state in the frame in which that result was applied. This process is referred to as apply-time state binding.

Event-Driven Path: STT finalization, input validation, LLM-based response generation, finalization of the first response sentence, the TTS request, completion of the full response, and post-response analysis proceed asynchronously. The first-sentence callback is passed to the Unity main thread via an inter-thread queue within the author-developed Unity C# class `ChatGPTManager`. Unity's `Update` handles this callback and connects it to the TTS request. The use of this queue was limited to the process of forwarding streaming text to the Unity main thread.

The learner's speech input was finalized based on the STT-final result from speech-to-text recognition and the end-of-speech event. If the STT-final result was available and the local volume remained below −45 dBFS for 500 ms, the utterance was finalized using the local silence fallback.

After speech confirmation, the `Validation` module for input validation classified the `ConvType` as 0 (invalid input), 1 (practical-care utterance), 2 (emotional-support utterance), or 3 (scene-management utterance). For `ConvType = 0`, no normal response generation

occurred; instead, a fixed clarification response was presented. For valid inputs, the patient avatar's response was generated in a streaming manner using prompts specific to each classification. The inputs for response generation were the learner's utterance for that turn and the fixed persona settings. Conversation history, the current phase, and the VR scene state were not included as inputs. Specifications regarding the patient's response—such as character count, number of sentences, question format, and advice content—were requirements specified in the prompt; no deterministic checks or enforcement were performed at runtime.

A TTS request was issued as soon as the first response sentence was processed on the Unity main thread. After full-response completion, post-response analysis was initiated. In its codebook, `ActionClass` was defined as 1 (breast exposure), 2 (infant holding or posture), 3 (breast covering or clearing of the breastfeeding-preparation state), and 4 (other or ambiguous). `PhaseCandidate` was defined as 1 (greeting), 2 (breastfeeding preparation), 3 (breastfeeding practice), and 4 (ending or post-breastfeeding care).

During data collection, `useUnifiedAnalysis = true` was set. After full-response completion, the learner's utterance and the generated patient-avatar response were submitted in a single unified post-response analysis request that returned a JSON object. The settings were `temperature` = 0.0 and `max_tokens` = 120, and one JSON response containing multiple output fields was obtained. We did not use the API-level structured-output feature. The JSON format was specified in the system prompt and subjected to client-side extraction and parsing. In the state transition and log analysis described in this paper, we used only `classification` (`ActionClass`) and `situation` (`PhaseCandidate`). Other fields returned by the unified post-response analysis were not used in this study. `classification` and `situation` are two logical fields obtained from a single JSON response for the same turn. The interpretation of the generated patient avatar response may affect both output fields.

Empty outputs and outputs that could not be parsed as integers were assigned a value of 0. Range checks for values 1–4 were not performed during execution. In the post-audit, the stored values were limited to 0–4; there were no negative numbers or values of 5 or higher. However, it was not possible to distinguish from the stored CSV alone whether a value of 0 resulted from an explicit output, an empty output, a JSON extraction failure, or an integer parsing failure.

Once post-response analysis for a valid input was complete, the results accepted for the current turn were written to `CurrentClassification` and `CurrentSituation`, and `GestureID` was incremented. During session initialization and when invalid input was confirmed, `GestureID` was updated while retaining the current callback values corresponding to that execution path. The `TrainingFlowController`, set to a Script Execution Order value of 10,000, detected changes in `GestureID` via `LateUpdate` and copied the current callback values to the held semantic labels. Next, the checklist and phase-transition conditions were recalculated, and at most one phase transition per frame was performed. Subsequently, the `TrainingFlowLogger` with a Script Execution Order setting of 20,000 compared the current state snapshot with that of the previous frame during `LateUpdate` and recorded events tagged with the Unity frame number.

Details of each functional component and runtime prompts are shown in Supplementary Tables S1–S3.

The processing flow within a single turn is shown in Figure 2, and the corresponding asynchronous result-transfer and frame-indexed scene-update procedure is summarized in Algorithm 1.

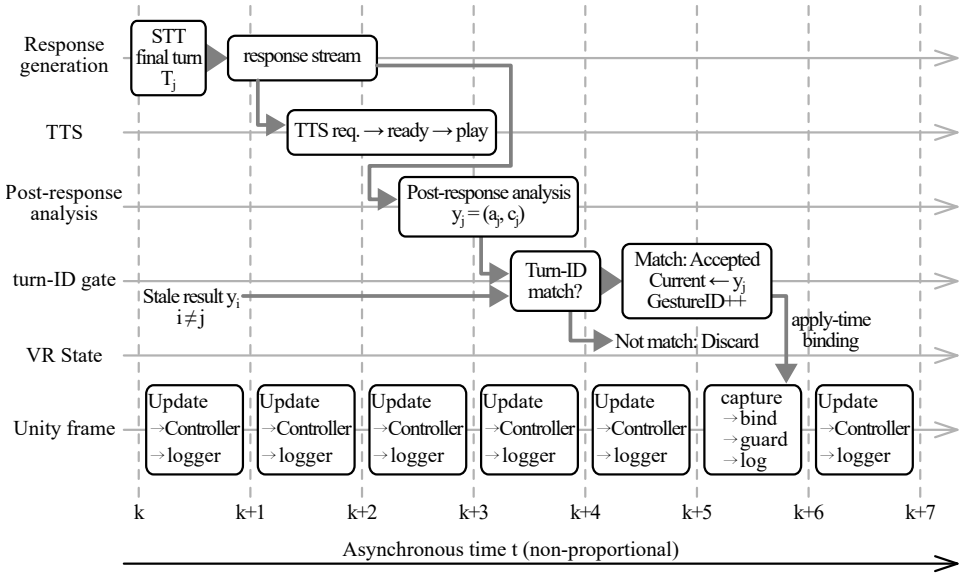


Figure 2. Schematic diagram of a single turn's execution, integrating asynchronous callbacks and Unity frame processing. Delayed results from a stale turn do not update the current callback values or the `GestureID`. Accepted results update the current callback values. Subsequently, the `TrainingFlowController` updates the held semantic labels, and the `TrainingFlowLogger` records the differential events. The timeline in the figure is a schematic representation.

Algorithm 1 GestureID-Based Asynchronous Result Transfer and Frame-Indexed Scene Update

Persistent state: active turn T_a, single-slot pending input I_p, current callback values (A, C), current GestureID G, last observed GestureID G^*, held labels $(\bar{a}, \bar{c})$, frame snapshot $\mathcal{S}^{\mathrm{rt}}$
STT-final event(u): if avatar speech is active, replace the single pending input I_p with u (latest-wins); otherwise, start a new turn and update T_a
First-sentence callback(j, s): if $j = T_a$, pass s to the main-thread display/TTS path
Full-response callback(j, r): if $j = T_a$, start post-response analysis for (j, r)
Post-analysis completion(j, a, c): if $j \neq T_a$, ignore the stale result; otherwise set $(A, C) \leftarrow (a, c)$ and increment G
Initialization/invalid-input finalization: update the current dialogue-state snapshot according to the runtime path and increment G; invalid input does not trigger unified post-response analysis
for each Unity frame k **do**
    normal `Update`: update infant placement, mother state, and latch-related states
    `TrainingFlowController.LateUpdate` (order value 10,000):
    if $G \neq G^*$ **then**
        $(\bar{a}, \bar{c}) \leftarrow (A, C)$; $G^* \leftarrow G$
    end if
    acquire current infant placement, mother-zone occupancy, and mother preparation state
    evaluate the guard for the current phase and apply at most one phase transition
    evaluate the current completion predicate d_k
    `TrainingFlowLogger.LateUpdate` (order value 20,000): compare the current snapshot with the previous snapshot and record state/phase changes and the rising edge e_k^{done}
end for
Playback-complete/fallback event(j): if $j = T_a$, release the turn; if $I_p \neq \emptyset$, consume the pending input and start the next turn

We refer to the association of state evaluation, phase updates, and log differences with `Time.frameCount` on the Unity main thread as "frame-indexed processing." This definition represents the execution order and log references based on frame numbers.

The newborn-avatar placement state was managed by `BabyPlacementState`. While being held, `HeldByStudent` took precedence. The placement destination was determined only at the moment of grasp release. If the avatar was within the mother's area during grasp release, it was set to `InMotherArms`; if within the cot area, `InCot`; and if neither, `Unknown`.

The mother avatar's holding posture was not maintained as an independent scenario state. Instead, it was derived in every frame from whether the newborn-avatar placement

state was `InMotherArms`. The mother avatar's breastfeeding-preparation state was established when `CurrentClassification`=1 and cleared when it was 3. For all other values, the state from the previous frame was retained.

During data collection, `requirePhase3 = true` was set. In Phase 3, simulated latch was initiated when all of the following conditions were met: the newborn-avatar placement state was `InMotherArms`, the mother avatar's breastfeeding-preparation state was active, and the distance between the mouth anchor and the nipple reference position was 50 mm or less. If the same distance was 100 mm or greater during simulated latch, the latch was released. `snapToLatchOnRelease = true` was set. If the conditions for allowing simulated latch were met at the moment of grasp release within the mother's area, and the distance was 150 mm or less, the newborn avatar was translated so that the mouth anchor coincided with the nipple reference position. This process is referred to as "release-snap." `snapDetectDistance` = 20 mm was set as the threshold exclusively for recording the `SNAP_ON_RELEASE` diagnostic event. In this scenario, simulated latch was not included as a prerequisite for transitioning to Phase 4.

The main runtime phase-transition guards are defined in Equation (4), while the completion predicate and completion-entry event are defined in Equations (6) and (7), respectively; their relationships within the scene model are shown in Figure 3. A summary of the state variables and update rules is provided in Table 3 and Supplementary Tables S17 and S18.

Unity 6000.2.6f2 was used for development and execution. The VR hardware consisted of a Meta Quest Pro and Meta Quest Touch Pro Controllers (Meta Platforms Technologies, LLC, Menlo Park, CA, USA). The specifications of the experimental PC were as follows: CPU—Core i7-13700H; GPU—GeForce RTX 4060; memory—16 GB; OS—Windows 11 Home.

The access period for the cloud API services was from 24 February to 22 April 2026. For speech-to-text recognition, we used streaming recognition from Google Cloud Speech-to-Text V2, with the language set to Japanese (`ja-JP`). For patient-avatar response generation and post-response analysis, we used `gpt-4o-mini` from the OpenAI Chat Completions API. For text-to-speech synthesis, we used `gpt-4o-mini-tts` from the OpenAI Speech API, with the voice set to `shimmer`. The same Unity Inspector settings were used for all 20 sessions, with scene-transition operations and state paths, half-duplex turn management, unified post-response analysis, and simulated latch assistance enabled. The main configuration values are shown in Supplementary Table S20.

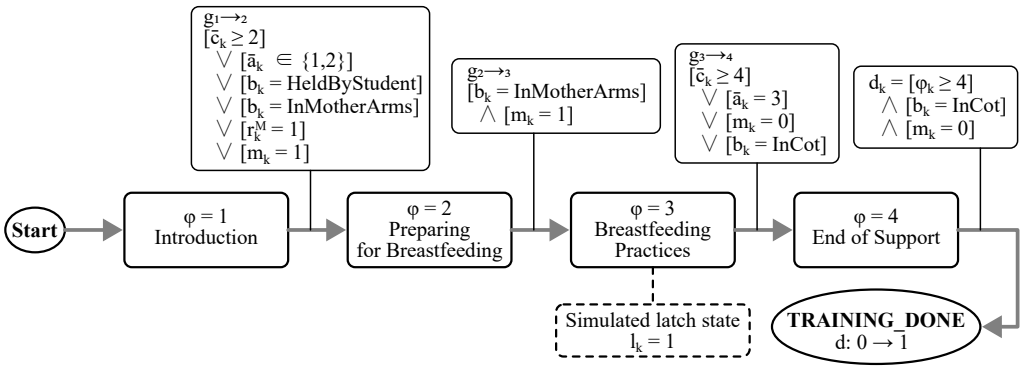


Figure 3. Finite-state scene model based on the Unity implementation and key phase-transition conditions. $\bar{a}$ and $\bar{c}$ are held semantic labels that are captured when a `GestureID` changes and are retained until the next change. r^{M} represents the frame input for the mother-zone occupancy flag; m represents the mother avatar's breastfeeding-preparation state; and b represents the newborn avatar's placement state. $\ell = 1$ denotes the simulated latch state within Phase 3 and is not a prerequisite for transitioning to Phase 4. In each Unity frame, only the finite-state guards for the current phase are evaluated, with at most one phase transition per frame.

Table 3. Unity runtime state variables, frame inputs, held semantic labels, and derived events.

Subject	Type	Domain/Retention	Update/Usage	Interpretation/Notes
ϕ_k	Discrete state	$\{1, 2, 3, 4\}$	Evaluated by the guard for the current phase; increases monotonically with at most one phase transition per frame	Discrete phase-control state
b_k	Discrete state	`Unknown`, `HeldByStudent`, `InMotherArms`, `InCot`	Updated from area determination at grasp release; used in all phases and in the completion predicate	Entering an area alone does not confirm placement
r_k^{M}	Frame input	$\{0, 1\}$	Updated from enter/exit events of the mother-zone trigger; used in Phase 1→2	Distinct from b_k; not used as a state input for offline recomputation
m_k	Discrete state	$\{0, 1\}$; set by Class 1 and cleared by Class 3	Used in Phase 2→3, Phase 3→4, and the completion predicate	Class 3 contributes through both direct and state-mediated pathways
ℓ_k	Discrete state	$\{0, 1\}$	Records the simulated latch state within Phase 3	Recording only; not required for the Phase 4 transition
$\bar{a}_k, \bar{c}_k$	Held semantic labels	Integers including 0; captured when `GestureID` changes and retained until the next change	Inputs to the Phase 1→2 and Phase 3→4 guards	Semantic validity is evaluated separately by expert review
d_k	Derived predicate	$\{0, 1\}$	Evaluated every frame as $[\phi_k \geq 4] \wedge [b_k = \mathrm{InCot}] \wedge [m_k = 0]$	Not latched; returns to 0 when the condition no longer holds
e_k^{done}	Completion-entry event	$\{0, 1\}$	Records `TRAINING_DONE` on the rising edge of d_k	Derived event; recorded again after a false→true re-entry

The LLM request timeout was set to 30 s, and no retries were performed on the LLM side. For TTS, only one retry was performed for the initial voice retrieval. If TTS playback could not begin within 8.0 s, that turn was continued using a text-only fallback.

A mobile Wi-Fi (5G) network was used for the experiments. Communication speed, RTT (Round Trip Time), and packet loss rate were not fixed and were not recorded. Therefore, the measured runtime latency was treated as a measure reflecting the entire evaluation environment, including the device, cloud service, network, and Unity implementation.

In frame-indexed execution, state evaluation, phase updates, and differential logs were associated with Unity frame number k. After state-transition processing, the `TrainingFlowLogger` evaluated the final state snapshot for that frame and recorded only the differential events generated relative to the previous frame. Consequently, intermediate states within the same frame were not stored, and the causal processing order cannot be inferred from the CSV row order of multiple events sharing the same Unity frame number.

The pose-and-interaction log was recorded at approximately 20 Hz. Scene state evaluation was performed in each Unity frame. The rendering frame rate and HMD refresh rate were not included in the recording. The training-progress log recorded `Time.frameCount` and `Time.unscaledTime` after the session began.

Since the timestamps of all rendering frames were not saved, jitter, drops, and deadline misses were not evaluated. We described the operational characteristics of frame-indexed execution in terms of log references based on common Unity frame numbers, state events during the post-response-analysis waiting interval, and frame mappings obtained through auxiliary offline recomputation.

The data sent to external cloud APIs varies depending on the processing purpose. For speech-to-text, the participant's spoken audio was sent to Google Cloud Speech-to-Text V2. For input validation, response generation, and post-response analysis, the finalized utterance text and the corresponding prompts were sent to the OpenAI API. For post-response analysis, the generated patient-avatar response text was also sent. For text-to-speech synthesis, the patient avatar response text was sent to the OpenAI Speech API.

3.4. Scenario, Procedure, and Ethics

The training scenario was set in a breastfeeding guidance session with a primiparous woman on postpartum day 1. The co-author specializing in maternal nursing and midwifery education developed the scenario by referring to WHO/UNICEF documents related to breastfeeding support and competencies [30–32]. However, no systematic evaluation of the scenario content validity by multiple independent experts was conducted.

The patient avatar was designed as a primiparous woman experiencing postpartum fatigue, as well as anxiety and confusion regarding her first breastfeeding experience. The scenario included an explanation upon entering the room, assistance with holding the newborn, preparation for breastfeeding, simulated latch and adjustment of the breastfeeding position, and post-breastfeeding safety. Post-breastfeeding operations included returning the newborn avatar to the cot and clearing the mother avatar's breastfeeding-preparation state, thereby returning her to a breast-covered state. These operational states were defined for the purpose of managing scene state transitions and log analysis. They do not serve as criteria for comprehensively evaluating the learner's breastfeeding guidance skills.

The role of the dialogue partner is referred to as the "patient avatar." When referring to physical states within the VR space—such as holding posture, breast exposure, and breast covering—the term "mother avatar" is used. The hospital room was set up with a mother avatar, a newborn avatar, and a cot. This allows learners to practice assisting with the movement of the newborn avatar while explaining the process to the mother, all within the same scene (Figure 4). The 3D model of the mother avatar was created using MakeHuman. Commercially available 3D assets were used for the newborn avatar, hospital room, furniture, and other elements.

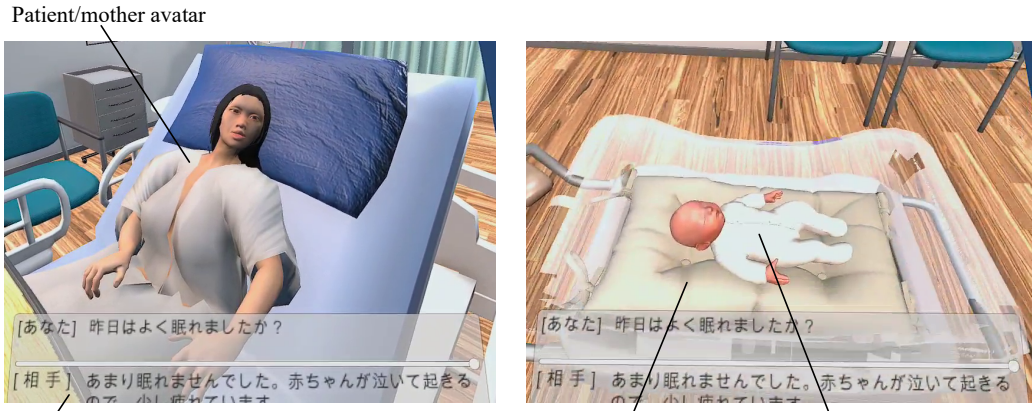


Figure 4. Example of a VR breastfeeding support training scene.

Nursing students who had completed their maternal nursing clinical practicum were recruited by convenience sampling, primarily from educational institutions in the Hokuriku region. Students who had not completed their maternal nursing clinical practicum were excluded. Students who had also received midwifery education were included in the study. We explained that participation, nonparticipation, or withdrawal from the study would not result in any educational disadvantages, such as negative effects on grades or clinical-practicum evaluations, and kept the research assessment separate from educational assessment.

Twenty nursing students participated, and each participant took part in this experiment only once. The experiment was conducted from 24 February to 22 April 2026, in a laboratory on the university campus. Participant demographics are presented in Section 5.1.

Prior to the experiment, we explained how to operate the VR equipment and provided an overview of the training scenario. Since this study did not evaluate the system's response to natural barge-ins, participants were instructed to speak only after the patient avatar had finished responding. Participants wore HMDs and interacted verbally with the patient avatar and performed assistive actions in the VR scene at their own discretion. They indicated the end of the session once they had completed breastfeeding guidance or determined that continuing was difficult. We did not present the system's defined completion state—namely, the newborn avatar being in the cot area and the clearing of the breastfeeding-preparation state—as an explicit achievement criterion.

After the experience, participants completed a post-experiment questionnaire, and a debriefing was conducted using first-person video from the HMD wearer's viewpoint and the conversation history. We drew on the general approach of linking post-simulation reflection to specific events [33]. However, the content of the debriefing and the facilitator's interventions were not included in the analysis for this paper. Criteria were established to immediately terminate the experiment if VR motion sickness or physical discomfort occurred during the experiment.

The VR experience duration was defined as the time from when the participant began the scenario within the VR space until they reported completion and the experimenter performed the termination procedure.

This study was conducted after obtaining ethics approval from the Ethics Subcommittee of the Department of Nursing, Fukui Prefectural University (Review No. M2024006). Prior to implementation, the purpose, methods, data to be collected, anticipated burdens and risks, and the handling of personal information were explained in writing and verbally to the facility director and research participants. Participation was voluntary, and it was explained that withdrawing from the study at any time would not result in any academic disadvantages, such as effects on grades, practical training evaluations, or other educational consequences. It was also explained that participants could immediately stop the experience if they experienced VR motion sickness or physical discomfort. Only those who provided written informed consent were included in the study.

This system utilized external cloud APIs for speech-to-text, LLM-based response generation and post-response analysis, and text-to-speech synthesis. Participants were informed, both in writing and verbally, that their spoken audio during the VR experience, the finalized utterance text confirmed by speech-to-text technology, and the patient avatar's response text would be transmitted to external cloud services provided by Google Cloud and OpenAI. Written informed consent was also obtained regarding this matter. Prior to the experiment, participants were instructed not to utter their full names, student ID numbers, contact information, information about real patients, or any other information that could directly identify them.

During the experiment, we used the standard data processing settings for each service. We did not enable the opt-in setting that allows research data sent to the API to be used for training foundation models. Data retention conditions on the service provider's side followed the contracts and service settings in effect at the time of the experiment.

Research data was managed using a study ID. Questionnaire responses, free-text responses, dialogue logs, training-progress logs, pose-and-interaction logs, first-person video from the HMD wearer's viewpoint, and end-of-session records were stored in restricted-access storage that was accessible only to the principal investigator and co-investigators.

For analysis and presentation in the paper, only non-identifiable aggregated values or anonymized text excerpts were used.

4. Evaluation Methods

4.1. Original-Study Logging and Post Hoc Cross-Log Alignment

In the latency analysis, we evaluated the internal software latency—the time from when the finalized utterance text was reflected in the VR runtime system until the audio playback command was issued to the patient avatar. In this implementation, we operationally defined the time at which the finalized utterance text entered the VR runtime as the time at which it was committed to the conversation state on the Unity main thread. To describe the internal runtime latency of the system, we recorded the following times: t_{STT} is an implementation-level log field representing this commit time. t_{AIstart} is the start time of response generation, t_{sent1} is the time when the first-sentence callback was processed by Unity's `Update`, and t_{TTSreq} is the time of the first TTS request. t_{ready} is the time when the TTS streaming prebuffer reached a playable amount, and t_{play} is the time immediately before the call to `AudioSource.Play()`. The primary metric is defined as the internal software latency from the main-thread commit of the finalized utterance text to the audio playback command, and

$$L_{\mathrm{STT}\rightarrow\mathrm{play}} = t_{\mathrm{play}} - t_{\mathrm{STT}}. \tag{8}$$

This metric does not include the time from the acoustic end of the learner's utterance to the finalization of STT. It is not a value directly measured from the actual acoustic onset from the loudspeaker.

The dialogue log recorded the spoken text, the patient avatar's response, the processing times for each step, and the runtime labels. The training-progress log recorded the held semantic labels, phase, newborn-avatar placement state, mother's status, simulated latch, checklist, progress signals, and completion-entry events. The pose-and-interaction log recorded tracking values and input operations for the HMD and controller at approximately 20 Hz, assigning them common Unity frame numbers.

We conducted a post hoc audit of the 20 dialogue logs. After excluding the 20 `START` rows, one from each session, 786 input rows remained. The breakdown was 639 valid input lines and 147 invalid input lines. The saved `MessageID`s were unique within each file and increased monotonically. However, there were a total of 75 missing IDs across 11 sessions. The `MessageID` increases each time the STT-confirmed text is reflected in the Unity main thread. On the other hand, the conversation CSV records only those inputs that reached the saving stage after passing through the dialogue processing. The denominator for the analysis was set to the 786 saved dialogue-log input rows. The total number of STT finalization events that were not saved cannot be reconstructed from the dialogue log. Missing entries may include cases where pending-input replacement was performed using the latest-wins policy, prior processing was canceled upon the start of a new turn, or other non-saving paths. However, it was not possible to reconstruct the number of cases by cause using only the dialogue log CSV.

Each row in the conversation CSV was finalized only once, upon completion of post-response analysis. If t_{ready} or t_{play} had not been recorded at that point, the row was saved with a value of -1. Even if TTS playback began afterward, the existing row was not updated. Therefore, the 21 cases represent instances where the time was not recorded at the time the CSV was finalized, and it is impossible to determine whether audio playback actually occurred. There may be right-censoring-like missingness because post-response analysis and the associated CSV write could be completed before the TTS prebuffer-ready signal or the audio playback command. Since the complete-case analysis is biased toward turns

where TTS events were obtained before the CSV was finalized, we did not assume that the missing values occurred randomly.

`BOT_GESTURE` is a state snapshot of the dialogue and analysis state recorded by `TrainingFlowLogger` when it detects a change in `GestureID`. For valid inputs, it contains the saved `ActionClass` and `PhaseCandidate` values produced by the unified post-response analysis. However, because `BOT_GESTURE` is also recorded at session initialization and when an invalid input is confirmed, we did not treat all `BOT_GESTURE` entries as independent LLM-based post-response analysis results. The first `BOT_GESTURE` in all 20 sessions had `gid = 1`, `llm_sit = 1`, and `llm_cls = 0`; these were excluded from the correspondence with the saved dialogue-log input row as initialization snapshots.

Neither the dialogue logs nor the training-progress logs stored a common turn ID or request ID. Therefore, for the dialogue logs, we sorted all saved dialogue-log input rows (excluding the `START` line) in the order of their appearance in the file (hereinafter referred to as "save order"); for the training-progress logs, we sorted the `BOT_GESTURE` entries (excluding the initialization snapshot) by Unity frame number and mapped them to the save order within each session. In the 19 sessions, the 779 non-`START` dialogue log lines and the 779 post-initialization `BOT_GESTURE` entries had one-to-one correspondence. For the 632 valid input lines among these, we cross-referenced the `ActionClass/PhaseCandidate` values in the dialogue logs with the `llm_cls/llm_sit` values in the training-progress logs. A semantic evaluation of the LLM-derived semantic labels was conducted separately as a two-expert reference evaluation covering all 639 cases.

For the 147 invalid input lines, no unified post-response analysis was performed, and the dialogue logs stored `Situation = 0` and `Classification = 0`. In the corresponding training-progress logs, we confirmed `llm_cls = 0` for all 147 lines; however, since `llm_sit` recorded the current phase at that point in time, we excluded these from the Situation value comparison. For the 632 valid input lines, we compared the label values within the logs after mapping them in the order they were saved.

The training-progress log and the pose-and-interaction log were referenced using a common Unity frame number. For sessions where the elapsed-time origin differed between the dialogue log and the training-progress log, t_{STT} was converted to the training-progress log's timeline using a constant-offset conversion calculated from the session start event. In the remaining one session, there were 82 instances of `BOT_GESTURE` after initialization, whereas there were only 7 lines of non-`START` dialogue logs that could serve as matching candidates, and the start time of the dialogue log was 173 s earlier than that in the training-progress log. Therefore, we determined that unique post hoc alignment was not possible and excluded this session from analyses using time series derived from the post hoc alignment of multiple logs.

`SESSION_START` was recorded in all 20 files, and an explicit `SESSION_END` was confirmed in 19 out of 20 sessions. Even for the remaining one session, we were able to obtain the state, dialogue events, and the final state snapshot. In the supplementary final-state audit, the audit was conducted based on the final-state snapshot and the experiment completion record; state reconstruction through the accumulation of event sequences was not performed.

The combination method, exclusion rules, and missingness profile are shown in Supplementary Table S20.

When evaluating LLM-based virtual patients and AI/VR educational systems, it is important to distinguish between technical/system performance, user-reported experience, educational outcomes, and clinical validity [7,34,35]. In this paper, we distinguished the provenance of evidence used for each evaluation. First, we evaluated operational behavior, logging, cross-log alignment, and software latency based on the original participant sessions.

Second, after the participant study concluded, we conducted mechanism-specific software evaluations of phase guards, Turn-ID gates, the state-binding policy, and the revised logging schema. Third, we evaluated the agreement of all 639 semantic labels with annotations from two expert references and examined trace-conditioned local guard sensitivity for turns amenable to cross-log alignment. End-state audits and trace-driven reduced event recalculation were treated as auxiliary analyses, while questionnaires, free-text responses, and VR-related physical symptoms were treated as exploratory user-reported analyses.

Input records, turns, software test cases, semantic evaluation cases, and log events were distinguished as units of analysis. Observations at the turn and event levels were not treated as independent participants.

4.2. Auxiliary Offline State-Transition Recalculation and Observed-Trace Comparison

The auxiliary trace-driven reduced event recalculation used two state inputs. These were `BOT_GESTURE` and `BABY_STATE_CHANGE`. Only phase transitions and entries into final states were recalculated using reduced transition rules. The analysis focused on ordered event sequences within the same observation session. We did not re-execute the Unity runtime code, physics simulations, cloud processing, STT, LLM, or TTS.

The fixed initial state for the offline recomputation is defined as

$$\phi_0^{\mathrm{R}} = 1, \quad b_0^{\mathrm{R}} = \mathrm{InCot}, \quad m_0^{\mathrm{R}} = 0, \quad a_0^{\mathrm{R}} = 0, \quad c_0^{\mathrm{R}} = 1, \quad \rho_0^{\mathrm{R}} = 0, \quad d_0^{\mathrm{R}} = 0 \tag{9}$$

Here, $m^{\mathrm{R}} = 1$ indicates that the mother avatar's breastfeeding-preparation state is active, and $d^{\mathrm{R}} = 1$ indicates that the system-defined completion predicate holds. The first `BOT_GESTURE` of each session is an initialization snapshot with `llm_sit = 1` and `llm_cls = 0`. It was processed using the same procedure as the other `BOT_GESTURE`s, but since its values were identical to the fixed initial state, no state transitions occurred.

Each input line was assigned an internal index representing its row position in the original CSV. The data was sorted in ascending order by pairs of Unity frame numbers and internal indices, then grouped by frame number. For each frame, all offline recomputation inputs were applied in the order they were saved, and then the internal state, phase-transition conditions, and completion predicate were evaluated. If `BOT_GESTURE` is present, the saved values—including 0—were overwritten each time as

$$a_k^{\mathrm{R}} \leftarrow \mathrm{llm_cls}, \qquad c_k^{\mathrm{R}} \leftarrow \mathrm{llm_sit}. \tag{10}$$

If $a_k^{\mathrm{R}} = 1$, then $m_k^{\mathrm{R}} \leftarrow 1$; if $a_k^{\mathrm{R}} = 3$, then $m_k^{\mathrm{R}} \leftarrow 0$. In all other cases, the previous value of m^{R} was retained. If a `BABY_STATE_CHANGE` occurred, b_k^{R} was overwritten with the saved `baby_state_name`. A `BOT_GESTURE` is a state snapshot taken when the `GestureID` changes; it is also recorded at session initialization and when an invalid input is confirmed. For this reason, we did not treat every line as an independent LLM semantic inference event. In the 20 observed sessions, there were no instances where multiple offline recomputation input events were recorded within the same frame. Therefore, we were unable to evaluate same-frame input-order conflicts.

The Unity execution code does not include an explicit preparation start flag. On the other hand, for offline recomputation, we introduced the auxiliary analysis variable ρ_k^{R} to summarize the conditions that can be reconstructed from the saved logs.

$$\rho_k^{\mathrm{R}} = \rho_{k-1}^{\mathrm{R}} \vee [c_k^{\mathrm{R}} \geq 2] \vee [b_k^{\mathrm{R}} \neq \mathrm{InCot}] \vee [m_k^{\mathrm{R}} = 1] \tag{11}$$

The main reduced transition rules under the reference reduced recomputation condition (implementation name: `full_hybrid`) are

$$\begin{aligned} g^{\mathrm{R}}_{1\to 2}(k) &= [\phi^{\mathrm{R}}_{k-1} = 1] \wedge [\rho^{\mathrm{R}}_{k} = 1], \\ g^{\mathrm{R}}_{2\to 3}(k) &= [\phi^{\mathrm{R}}_{k-1} = 2] \wedge [b^{\mathrm{R}}_{k} = \mathrm{InMotherArms}] \wedge [m^{\mathrm{R}}_{k} = 1], \\ g^{\mathrm{R}}_{3\to 4}(k) &= [\phi^{\mathrm{R}}_{k-1} = 3] \wedge \Big([c^{\mathrm{R}}_{k} = 4] \vee [a^{\mathrm{R}}_{k} = 3] \vee [b^{\mathrm{R}}_{k} = \mathrm{InCot}] \Big). \end{aligned} \tag{12}$$

The third condition checks whether the current placement state, retained after input application, is `InCot`. Transition determination was implemented using a series of `if/elif` statements, and only the single step corresponding to the current phase was permitted in each frame.

After updating the phase,

$$d^{\mathrm{R}}_{k} = [\phi^{\mathrm{R}}_{k} = 4] \wedge [b^{\mathrm{R}}_{k} = \mathrm{InCot}] \wedge [m^{\mathrm{R}}_{k} = 0] \tag{13}$$

was evaluated. The corresponding completion-entry indicator was defined as

$$e^{\mathrm{done,R}}_{k} = [d^{\mathrm{R}}_{k-1} = 0] \wedge [d^{\mathrm{R}}_{k} = 1] \tag{14}$$

and was used to generate `TRAINING_DONE` in the offline recomputation only on the rising edge. The event was not regenerated while the completion condition remained true. If the system left the condition and subsequently re-entered it, the event was generated again within the same session.

These reduced transition rules use high-level states that can be reconstructed from the stored logs. The preparation-start conditions for Phase 1→2 were aggregated into ρ^{R}, whereas Phase 3→4 was evaluated using $c^{\mathrm{R}} = 4$, $a^{\mathrm{R}} = 3$, or $b^{\mathrm{R}} = \mathrm{InCot}$. The subject of evaluation in this supplementary analysis is the correspondence of ordered event sequences when this abbreviation rule is applied to the 20 target sessions. Formal equivalence to the runtime model and generalization to unobserved conditions require independent verification.

Verification was performed on a session-by-session basis. We extracted the ordered event sequence of `PHASE_CHANGE` from the execution log as

$$\mathcal{L}^{\phi}_{s} = \left[(f_i, \phi^{-}_{i}, \phi^{+}_{i}) \right]_{i=1}^{n_s} \tag{15}$$

and generated a sequence of the same format, $\mathcal{R}^{\phi}_{s}$, from the offline recomputation output. For `TRAINING_DONE`, we generated the sequence of occurrence frames from the execution logs, $\mathcal{L}^{\mathrm{done}}_{s} = [f_i]$, and the recomputation sequence, $\mathcal{R}^{\mathrm{done}}_{s}$. Using Python 3.13.14, we checked for list equality between $\mathcal{L}^{\phi}_{s}$ and $\mathcal{R}^{\phi}_{s}$, as well as between $\mathcal{L}^{\mathrm{done}}_{s}$ and $\mathcal{R}^{\mathrm{done}}_{s}$, for the ordered Python lists of each session. We did not perform comparisons using counters or multisets. We concluded that there were no events missing from the replay output or extra events when the lengths, order, and all elements of the two sequences matched. The differences between the input/output boundaries and the Unity runtime model are shown in Supplementary Tables S18 and S19.

4.3. Post-Study Mechanism-Specific Software Evaluation

Separate from the participant sessions, we conducted post-study deterministic software tests to examine the operational range of the implemented control mechanisms. For phase guards, we constructed 1600 test cases from a predefined discrete input grid and compared the next-phase output of the Unity runtime implementation with that of a specification-based oracle, an alternative implementation that does not call the production guard method. These tests were designed to examine implementation consistency across

the defined discrete test grid and were not intended as formal verification of the entire software system.

For Turn-ID gates, cross-turn stale asynchronous callbacks were artificially injected. We tested four stale TTS callbacks and configured paired gate-ON/test-only-bypass conditions S1–S4 for semantic results. Furthermore, as a negative control to verify the scope of Turn-ID guarantees, we conducted test S5, in which duplicate semantic callbacks were provided. The evaluation criteria were the impact on semantic labels, `GestureID`, bounded avatar state, and local phase outcome.

Given that the original participant-session logs lacked a common runtime primary key spanning multiple components, we implemented a revised logging schema in the post-study phase that shares session, turn, request, callback, and semantic event provenance. For this schema, we performed 10 predefined integration checks using a synthetic integration trace independent of participant sessions. Evaluation criteria included required common-ID fields, shared session ID, global event-ID uniqueness, turn-level and request-level deterministic joins between Bot/TrainingFlow/Pose logs, linkage to accepted semantic callbacks, provenance of discarded stale callbacks, CSV round-trip, and revised logger source integration. This test targeted the post-study revised implementation and did not retroactively modify or repair the logging provenance of the original 20 participant sessions.

For 632 turns that were uniquely mapped across the original logs, we compared the "earlier request-proxy snapshot" from semantic processing with the "apply snapshot" at the time the result was applied. The request-proxy snapshot was constructed retrospectively based on the post-response semantic-analysis request timestamp for the turn in question, using the latest values recorded up to that point for each guard-relevant state. The comparison included newborn holding/placement, mother-zone occupancy, and the mother-avatar breastfeeding-preparation state prior to the application of semantic labels. We compared the guard-relevant physical states in both snapshots and identified turns in which at least one selected physical state had changed. Furthermore, for each turn, using the same LLM-derived semantic labels, we recalculated the local next-phase outcome from both the request-proxy state and the apply-time state. The focus of this analysis is the local guard outcome within a single turn; it does not counterfactually reconstruct subsequent participant behavior or scene trajectories.

4.4. Full-Sample Expert Reference Evaluation of Semantic Labels

For all 639 valid dialogue turns, two domain experts specializing in maternal nursing and midwifery education independently evaluated the ActionClass and PhaseCandidate. Expert 1 is a co-author of this paper. Expert 2 is a colleague of Expert 1 and a researcher specializing in maternal nursing. Neither expert referred to the other's evaluation results; they evaluated each case independently. These two experts were the same individuals who had previously conducted independent evaluations in the focused 200-case evaluation. In this full-sample evaluation, neither expert referred to their own results from the previous 200-case evaluation. Furthermore, during the evaluation, the original LLM-derived labels, participant identity, actual phase, VR state, and observed scene transitions were not presented to either evaluator. No adjudicated consensus label was created; instead, the evaluation results from the two evaluators were retained as parallel expert reference annotations. Therefore, this evaluation was not positioned as external validation independent of the research team, but rather as a reference-specific concordance evaluation based on independent assessments by two domain experts. The proportion of LLM outputs ranging from 1 to 4 was defined as coverage. In the primary analysis, LLM output 0 was included as an error or abstention, and we calculated strict agreement, class-wise precision, recall, F1,

and macro-F1 for each expert reference. The conditional analysis, which excluded output 0, was treated as a secondary analysis.

For inter-expert agreement, we calculated raw agreement and Cohen's κ. Additionally, for subsets where both experts assigned the same label, we descriptively calculated the agreement between the LLM and that common label. In cases where the two experts assigned different labels, we did not define adjudicated accuracy.

We calculated the 95% confidence intervals for strict agreement and macro-F1 using 2000 bootstrap resamples with participants as clusters, and reported percentile intervals.

4.5. Trace-Conditioned Local Guard Sensitivity Analysis

To examine the potential impact of differences in expert semantic labels on local scene control, we conducted a trace-conditioned local guard sensitivity analysis using 632 turns for which cross-log alignment was possible. For each turn, we fixed the observed physical VR state of the frame to which the semantic result was applied, as well as the mother-avatar state prior to the application of the semantic label.

First, we recalculated the local next-phase outcome based on runtime guard rules from the original LLM-derived ActionClass and PhaseCandidate, and verified its consistency with the observed apply-time outcome. Next, using Expert 1 and Expert 2 as separate references, we replaced (i) only ActionClass, (ii) only PhaseCandidate, and (iii) both labels with the expert labels, and recalculated the local next-phase outcome under the same observed physical state.

We applied the bounded mother-avatar state update based on ActionClass in the same manner as the runtime rule: when ActionClass = 1, the breastfeeding preparation state was established; when ActionClass = 3, the breastfeeding-preparation state was cleared; and in all other cases, the pre-label state was maintained. The primary sensitivity denominator was defined as the turns in Phases 1–3 during which the phase transition guard was active.

This analysis evaluates the sensitivity of local next-phase decisions within the same observed frame; it is not a counterfactual causal analysis that estimates participant behavior, subsequent scene trajectories, educational outcomes, or clinical consequences when expert labels are used.

4.6. Statistical Analysis and Exploratory Measures

In the objective log analysis of the original participant sessions, we distinguished between participants, sessions, input records from dialogue logs, dialogue turns, and log events. As descriptive statistics, we calculated the median, interquartile range (IQR), mean, standard deviation, and distribution percentiles. Since values at the saved dialogue-log input row, turn, and event levels are nested within participants, we did not treat them as independent participants and focused our reporting on descriptive statistics. Clopper–Pearson exact binomial confidence intervals were calculated for binary reachability metrics, with participants or sessions as the units of analysis. A value of -1 was treated as missing, indicating that the corresponding value or timestamp was unavailable in the saved record.

The delay from the main thread commit of the finalized utterance text to the audio playback command was first calculated using a complete-case analysis after removing missing values. Among the valid inputs, for the 21 cases where the TTS prebuffer ready time or the audio playback command time was not recorded at the time the CSV was finalized, a sensitivity analysis was conducted using four conditions—the observed median, observed P95, observed maximum, or 8.0 s—for the time from the TTS request to the audio playback command. The upper limit for the text-only fallback is 8.0 s. The purpose of the imputation is to verify the sensitivity of representative values to the assumptions made in handling missing data.

In the auxiliary trace-driven reduced event recalculation, the ordered event sequences from the execution log and the offline recomputation output were compared for each session. For `PHASE_CHANGE`, we created a sequence of tuples consisting of the Unity frame number, the phase before the phase transition, and the phase after the phase transition. For `TRAINING_DONE`, we created a sequence of occurrence frames. We performed equality comparisons on the number, order, and all elements of the Python lists; we did not perform multiset comparisons. The results were reported descriptively, and no hypothesis tests were conducted.

In the full-sample semantic evaluation, we calculated strict agreement, class-wise precision, recall, F1, and macro-F1 for each expert reference. For uncertainty intervals, we used participant-cluster bootstrap to calculate 95% percentile confidence intervals based on 2000 resamplings. For inter-expert agreement, we used raw agreement and Cohen's κ.

For mechanism-specific software tests, results were reported as the number of instances using predefined deterministic test cases; hypothesis testing was not performed using statistical samples. Regarding local guard sensitivity, the number and percentage of local outcome divergences for each replacement condition were reported descriptively.

User reports were treated as exploratory findings and summarized. The custom questionnaire consisted of 13 items on a 4-point scale, with scores of 3 or 4 considered positive responses; however, its validity and reliability as a scale were not verified. A single researcher performed thematic coding of the free-text responses. VR-related physical symptoms were assessed using the Simulator Sickness Questionnaire (SSQ) [36] before and after use, and comparisons were made using the Wilcoxon signed-rank test. No correction for multiple comparisons was performed for these exploratory user-reported analyses; the detailed items, aggregation methods, and results are presented in Supplementary Tables S4–S6.

5. Results

5.1. Participants and Operational Profile

There were 20 participants: 18 women and 2 men. The mean age was 21.60 ± 0.60 years. Five were third-year students and fifteen were fourth-year students; three had received midwifery education. The median duration of the VR experience was 12.6 min (IQR 7.0–15.7 min).

Dialogue logs, training-progress logs, and pose-and-interaction logs were saved for all 20 sessions under the evaluation settings; no logging interruptions or terminations due to safety reasons were recorded. No session was forcibly terminated because of a software or hardware failure (0/20). The results of the cross-log alignment are shown in Section 5.2 and Table 4.

The dialogue logs for all 20 sessions contained 786 input lines, excluding the 20 `START` lines. Of these, 639 were valid inputs and 147 were invalid inputs. We calculated the proportion of valid inputs among the saved dialogue-log input rows for each participant; the median across participants was 88.2% (IQR: 79.9–96.4%). This proportion represents the percentage of input rows saved in the dialogue log that were processed as valid inputs. Unrecorded STT-finalization events and the accuracy of utterance-intent understanding were not included in the evaluation of this metric.

The saved `MessageID` values were unique within each file and increased monotonically; however, there were a total of 75 missing IDs across 11 sessions. Therefore, the 786 lines were not regarded as the total number of STT-finalized inputs. It was not possible to determine from the dialogue logs alone whether the missing IDs resulted from pending-input replacement under the latest-wins policy, cancellation of previous turns, or other non-saved pathways.

The `ActionClass` and `PhaseCandidate` values in the saved dialogue logs were limited to 0–4; there were no negative values or values of 5 or higher. Of the 639 valid

inputs, ActionClass was within the intended range of 1–4 in 633 lines (99.1%), and PhaseCandidate was within that range in 631 lines (98.7%); there were 6 and 8 lines with a value of 0, respectively. Based on the saved CSV alone, it was not possible to distinguish between explicit "0" outputs and "0" values resulting from empty outputs, failed JSON extraction, or failed integer parsing. The breakdown of utterance types was as follows: practical-care utterances (270 lines), scene-management utterances (203 lines), emotional-support utterances (166 lines), and invalid inputs (147 lines). Among the 633 valid inputs with a nonzero ActionClass, the breakdown was as follows: "Other/Ambiguous" (549 lines), "Holding Methods" (48 lines), "Breast Exposure" (25 lines), and "Clearing of the Breastfeeding-Preparation State" (11 lines). Execution, missingness, and post hoc alignment details are summarized in Supplementary Table S20. The main operational results are summarized in Table 4.

Table 4. Key results from 20 human-in-the-loop sessions regarding operation, original log acquisition, and post hoc cross-log alignment.

Metric	Result	Denominator/Interpretation
Three primary logs/logging interruptions/safety-related terminations	20/20/0/20/0/20	Session
Explicit SESSION_END	19/20	1 session used a final state snapshot
Saved dialogue-log input row (excluding START)	786	Does not include events where an unsaved STT was finalized
Valid Inputs/Invalid Inputs	639/147	Denominator is the number of saved dialogue-log input rows
Missing MessageID	75 IDs (11 sessions)	Counts by cause cannot be reconstructed
ActionClass 1–4/PhaseCandidate 1–4	633/639/631/639	The remainder is 0; there are no negative numbers or values of 5 or higher
TTS timestamp missing	21/639	May include right-censoring-like missingness due to the CSV finalization point
Post-reconciliation time series for multiple logs can be constructed	19/20	1 session shows a mismatch of 82 records versus 7 lines
After initialization, BOT_GESTURE-non-START conversation lines	779/779	One-to-one correspondence in the order of the logs across 19 sessions
Matching of saved semantic label values after order-based correspondence	632/632	This reflects the consistency of saved values across the original logs and is not an indicator of semantic validity

5.2. Operational Behavior and Text-Commit-to-Playback-Command Latency

In 19 sessions, after excluding the initial BOT_GESTURE from each training-progress log, we were able to establish a one-to-one correspondence, in the order they were saved, between the 779 lines of non-START dialogue logs and the remaining 779 BOT_GESTURE entries. The breakdown was 632 valid inputs and 147 invalid inputs. The denominator for these 632 valid inputs consists of valid inputs saved in the dialogue logs and does not include unsaved inputs corresponding to missing MessageID numbers.

For the 632 valid inputs, the saved values for ActionClass/PhaseCandidate in the dialogue log and llm_cls/llm_sit in the training-progress log matched for all 632 lines. For the 147 invalid input lines, while llm_cls=0 was observed in 147 out of 147 lines, llm_sit recorded the current phase at that point in time; therefore, these lines were excluded from the Situation value comparison. The result of 632/632 indicates that the label values within the logs matched after post hoc reconciliation based on the order of storage.

Figure 5 illustrates the process of post hoc log reconciliation and supplementary analysis.

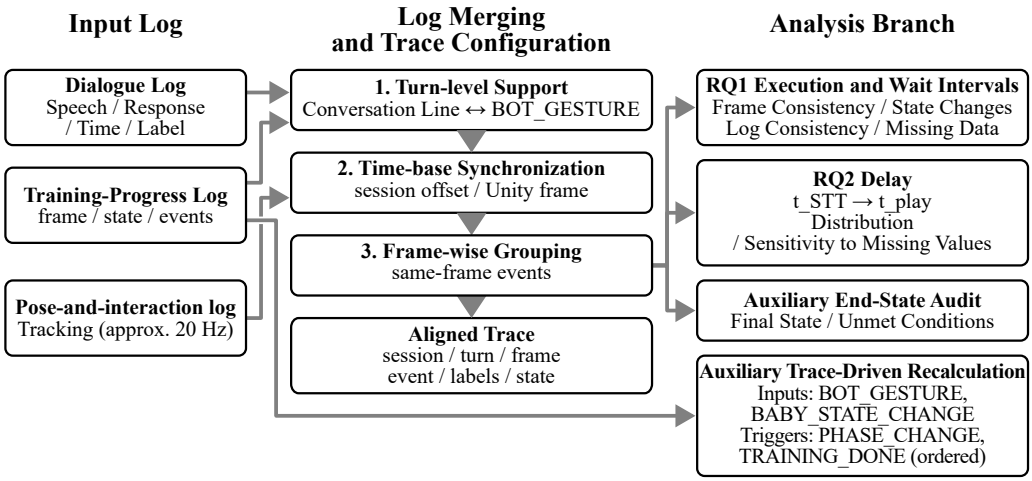


Figure 5. Post hoc alignment of dialogue logs and training-progress logs; reference to pose-and-interaction logs based on Unity frame numbers; evaluation of key technologies; and the workflow for auxiliary analysis. The final state audit utilized a final state snapshot and records from the end of the experiment. In the auxiliary trace-driven reduced event recalculation, we applied reduced transition rules using `BOT_GESTURE` and `BABY_STATE_CHANGE` as inputs, and reconciled the ordered event sequences of `PHASE_CHANGE` and `TRAINING_DONE` on a frame-by-frame basis.

The primary delay measured in this study is the "text-commit-to-playback-command latency"—the time from when the finalized utterance text is committed to the conversation state on the Unity main thread until just before the `AudioSource.Play()` call. When the participant-specific medians were aggregated across participants, the median was 2.61 s (IQR: 2.39–2.82 s). This endpoint refers to the software-based audio playback command, not the actual acoustic onset from the loudspeaker.

Of the 639 valid input lines saved in the dialogue log, 21 lines (3.3%) were excluded from the complete-case analysis because the TTS prebuffer ready time or the audio playback command time had not been recorded at the time the CSV was finalized; 618 lines were analyzed. The median per saved input line was 2.65 s, P90 was 5.20 s, P95 was 7.49 s, P99 was 11.78 s, and the maximum value was 15.07 s. The median of the participant-specific 95th percentiles was 4.91 s (IQR: 3.55–8.80 s).

Across participants, the median of the participant-specific medians was 1.458 s, and the time from the first TTS request to "prebuffer ready" was 1.072 s (Table 5, Panel A). The connection interval from the first-sentence callback processing to the first TTS request was in the sub-millisecond range. This interval describes the execution time of the connection path in question; we did not evaluate whether this connection path reduced latency relative to alternative connection methods.

The 21 missing cases may include right-censoring-like missingness in which the conversation CSV was finalized upon completion of post-response analysis, before the TTS prebuffer was ready or the audio playback command was issued. Therefore, these cases were not considered instances where audio was not played. In sensitivity analyses using the observed median, observed P95, observed maximum, or 8.0 s as the imputed value, the median did not change substantially, but the distribution tails were affected by the imputation condition.

Table 5. Latency components, distribution, and missing-data sensitivity analysis from finalized-text commit to playback command.

Panel A. Latency Component: Summary of the Participant-Specific Medians ($N = 20$)			
Metric	Log Field/Operational Definition	Analysis N	Median [IQR] (s)
From text commit to start of response generation	`lat_stt_to_chat`	639	0.0003 [0.0003–0.0003]
From the start of response generation to the first-sentence callback processing	`lat_chat_to_first`	639	1.458 [1.372–1.527]
From the first-sentence callback processing to the first TTS request	$t_{\text{TTSreq}} - t_{\text{sent1}}$	639	0.000 [0.000–0.000]
From TTS request to prebuffer ready	`lat_ttsreq_to_ready`	618	1.072 [0.984–1.116]
From prebuffer ready to play command	`lat_ready_to_play`	618	0.000 [0.000–0.000005]
From TTS request to play command	`lat_ttsreq_to_play`	618	1.072 [0.984–1.116]
From Final Text Commit to Play Command	`lat_stt_to_play`	618	2.612 [2.385–2.823]
Participant-specific 95th percentile for the same delay	95th percentile of each participant's `lat_stt_to_play`	20 participants	4.914 [3.550–8.795]

Panel B. Distribution of Saved Dialogue Logs by Saved Dialogue Log Input Row and Sensitivity Analysis of Missing Data				
Analysis Conditions	Input rows	Per-input-row median (s)	Saved-row P95 (s)	Participant-specific median (s)
Complete Cases	618	2.65	7.49	2.61
Observed Median Imputation	639	2.66	7.45	2.61
Observed P95 Imputation	639	2.68	7.59	2.66
Observed Maximum-Value Imputation	639	2.68	7.97	2.66
8.0-s imputation	639	2.68	9.30	2.66

Note: t_{STT} is the main thread commit time of the finalized utterance text; t_{sent1} is the main thread processing time of the first-sentence callback; t_{TTSreq} is the time of the first TTS request; t_{ready} is the prebuffer ready time of the streaming audio; and t_{play} is the time immediately before the call to `AudioSource.Play()`. In the interval from the processing of the first-sentence callback to the first TTS request, the timestamps for both events matched in 612 out of 639 input lines, and the remaining 27 lines also had a difference of less than 1 ms (maximum 0.294 ms). For all 20 participants, the participant-specific median was 0 s, and the summary across participants was 0.000 [0.000–0.000] s. The actual acoustic onset from the loudspeaker was not measured. The 21 missing entries may include right-censoring-like missingness caused by CSV finalization upon completion of post-response analysis. The denominator in this table consists of valid input lines saved in the dialogue log and does not include unsaved inputs corresponding to missing `MessageID` numbers.

5.3. Results of the Post-Study Mechanism-Specific Software Evaluation

In the post-study software evaluation, conducted separately from the participant sessions, we compared the Unity runtime's phase guard output against an alternative implementation of a specification-based oracle—which does not invoke the production guard method—across a predefined discrete input grid. The two implementations matched for all 1600 test cases (1600/1600). This result demonstrates consistency in the guard implementation within the defined discrete test grid and does not imply system-wide formal verification.

For the Turn-ID gate, we conducted mechanism-specific tests in which cross-turn stale asynchronous results were artificially injected. In 4 out of 4 cases, the stale TTS callback had no impact on the active turn. For semantic results, we conducted paired tests S1–S4. Under conditions where the standard Turn-ID gate was enabled, stale results were rejected; however, under the "test-only bypass" condition, the retained semantic label, `GestureID`, or local scene state changed. On the other hand, in the case of same-turn duplicate semantic callbacks, both callbacks belonging to the same active turn were accepted. Therefore, the scope of Turn-ID functionality supported by this experiment is cross-turn stale-result exclusion, not same-turn semantic-callback deduplication.

For the 632 cases where we could compare the request-proxy snapshot with the apply snapshot at the time of semantic result application, we compared the endpoint values of selected physical VR states related to the phase guard. At least one selected guard-relevant physical-state endpoint differed between the request-proxy and apply-time snapshots in 10 of 632 turns (1.6%). However, none of the 632 turns showed a difference in the local next-phase outcome between the two states when the same semantic labels were used. Furthermore, the local outcomes recalculated from the apply-time state matched the observed outcomes in all 632 out of 632 cases. Therefore, within the scope of the traces observed in this study, no additional local decision benefit of apply-time state binding over earlier request-proxy state binding was demonstrated. However, these results do not negate the usefulness of apply-time binding in the presence of unobserved states, longer processing delays, or different guard topologies.

Since the logs from the original participant sessions lacked a common primary key spanning multiple logs, we implemented a revised logging schema in the post-study phase to preserve session, turn, request, callback, and semantic event provenance. All 10 integration tests for the revised schema passed, and in synthetic integration traces, the three types of component logs could be uniquely joined on a per-turn and per-request basis. This verification is a post-study test of the revised implementation and does not retroactively repair the logs of the original 20 sessions.

5.4. Full-Sample Expert Evaluation of LLM-Derived Semantic Labels

For all 639 valid inputs, two domain experts independently evaluated the ActionClass and PhaseCandidate. For ActionClass, there were 534 out of 639 cases (83.6%) of exact agreement between the experts, and Cohen's κ was 0.681. The reference-specific strict agreement between the LLM and Expert 1 was 476 out of 639 cases (74.5%, 95% CI: 69.4–79.2%), and with Expert 2, it was 497 out of 639 cases (77.8%, 95% CI: 72.0–82.7%). The macro-F1 scores were 0.545 (95% CI: 0.484–0.600) and 0.589 (95% CI: 0.511–0.658), respectively. When limited to the 534 cases where both experts assigned the same ActionClass, the LLM matched that common label in 444 out of 534 cases (83.1%).

For PhaseCandidate, the exact agreement rate between experts was 339 out of 639 cases (53.1%), and Cohen's κ was 0.318. The reference-specific strict agreement between the LLM and Expert 1 was 108 out of 639 cases (16.9%, 95% CI: 13.3–20.8%), and with Expert 2, it was 225 out of 639 cases (35.2%, 95% CI: 30.9–39.9%). The macro-F1 scores were 0.172 (95% CI: 0.140–0.201) and 0.293 (95% CI: 0.264–0.318), respectively. Furthermore, even among the 339 cases where both experts assigned the same PhaseCandidate label, the LLM matched that common label in only 73 out of 339 cases (21.5%).

The class-wise performance of ActionClass was uneven; while the recall for Classes 1–3 ranged from 25.8% to 43.5% according to the reference experts, the recall for the high-frequency Class 4 ranged from 98.0% to 99.3%. Detailed confusion matrices and class-wise precision, recall, and F1 scores are shown in the Supplementary Materials.

Even in a secondary conditional analysis excluding output 0, agreement for ActionClass was 75.2%/78.5%, macro-F1 was 0.550/0.592, and for PhaseCandidate, agreement was 17.1%/35.7% and macro-F1 was 0.173/0.295; the interpretation remained consistent with that of the primary strict analysis.

5.5. Trace-Conditioned Local Guard Sensitivity to Semantic Labels

Of the 632 cross-log aligned turns, we focused on the 526 turns that occurred in Phases 1–3 and for which the local phase guard was active. While keeping the observed apply-frame VR state fixed, we replaced the LLM-derived labels with each expert reference label and recalculated the local next-phase outcome.

When only `ActionClass` was replaced with the label from Expert 1 or Expert 2, the local guard outcome changed in 19 out of 526 cases (3.6%) and 10 out of 526 cases (1.9%), respectively. When only PhaseCandidate was replaced, the figures were 182 out of 526 cases (34.6%) and 106 out of 526 cases (20.2%); when both labels were replaced simultaneously, the figures were 190 out of 526 cases (36.1%) and 111 out of 526 cases (21.1%).

Furthermore, there were 305 guard-active turns where the semantic labels assigned by Expert 1 and Expert 2 differed in at least one dimension; of these, the local guard outcome also differed between the experts in 133 out of 305 cases (43.6%). By phase, expert-specific local outcomes differed in 52 out of 61 cases (85.2%) in Phase 1, 2 out of 105 cases (1.9%) in Phase 2, and 79 out of 139 cases (56.8%) in Phase 3. Therefore, the proportion of semantic disagreement reflected in local phase decisions varied markedly across phases.

5.6. Auxiliary End-State Audit

We audited the operational state at the time of termination based on the final state snapshot from the training-progress log and the records at the end of the experiment. Phase 4 was reached in 12 of 20 sessions, and the system-defined completion state was achieved in 9 of 20 sessions. The eleven cases that did not meet the criteria were divided into one case where the breastfeeding preparation condition was not met, seven cases where the transition to Phase 4 was not recorded, and three cases where the clearing of the breastfeeding-preparation state did not occur within Phase 4 (Table 6). This analysis is a supplementary end-state audit based on the saved final state. The evaluation of skill attainment and educational effectiveness was not included in the scope of this analysis.

Table 6. Rates of reaching and achieving operational status, and breakdown of non-achievement cases based on final state snapshots ($N = 20$).

Evaluation Item	Number of Cases	Percentage (%)	95% CI (%)
Achievement and fulfillment among all participants ($N = 20$)			
Reached Phase 2	20	100.0	83.2–100.0
Reached Phase 3 (Breastfeeding Readiness Criteria Met)	19	95.0	75.1–99.9
Reached simulated latch	10	50.0	27.2–72.8
Reached Phase 4	12	60.0	36.1–80.9
Return of the newborn to the cot completed	12	60.0	36.1–80.9
Clearing of the breastfeeding-preparation state completed	9	45.0	23.1–68.5
System-defined completion state achieved	9	45.0	23.1–68.5
Final phases and unmet conditions among sessions in which the system-defined completion state was not achieved ($n = 11$)			
Ended in Phase 2: Breastfeeding Preparation (condition not met)	1	9.1	0.2–41.3
Ended in Phase 3 (no record of transition to Phase 4)	7	63.6	30.8–89.1
Ended in Phase 4 (breastfeeding readiness release not met within Phase 4)	3	27.3	6.0–61.0

Note: The 95% CI represents the Clopper–Pearson exact binomial confidence interval. The proportion of cases where conditions were not met and the confidence interval were calculated using the 11 participants who did not meet the conditions as the denominator.

5.7. Auxiliary Trace-Driven Reduced Event Recalculation

We applied the reference reduced recomputation condition (`full_hybrid`) defined in Section 4.2 to the training-progress logs from all 20 sessions. Records of `PHASE_CHANGE`, `TRAINING_DONE`, checklists, progress signals, and derived mother state records were not

used for state updates; instead, phase transitions and completion-entry events were recalculated using only `BOT_GESTURE` and `BABY_STATE_CHANGE`.

In 20 out of 20 sessions, the same ordered event sequence of `PHASE_CHANGE` events was obtained from both the execution logs and the offline recomputation. For all 51 elements, the Unity frame numbers, pre-transition phase, and post-transition phase matched. The breakdown was as follows: 20 instances of Phase 1→2, 19 instances of Phase 2→3, and 12 instances of Phase 3→4.

For `TRAINING_DONE` as well, the same sequence of occurrence frames was obtained in 20 out of 20 sessions, and all 10 elements corresponded to the same Unity frame numbers. The system-defined completion predicate was satisfied in nine sessions; in one of these sessions, the completion condition became false and then true again, so both the execution log and the offline recomputation contained two `TRAINING_DONE` events.

Within the scope of the 20 observed sessions and the explicitly defined reduction rules, the ordered event sequences obtained from the execution logs and trace-driven reduced recomputation matched for 51 instances of `PHASE_CHANGE` and 10 instances of `TRAINING_DONE`. This agreement demonstrates observed-trace consistency using saved higher-level states; it does not indicate equivalence with or formal verification of the Unity runtime code.

5.8. Exploratory User-Reported Findings

User reports are presented briefly as exploratory findings. In the custom questionnaire, all 20 respondents gave a positive response regarding "usefulness as nursing training" and "intention to continue using the system," while 13 out of 20 respondents gave a positive response regarding "utterance-intent understanding" and "response-onset timing." In the free-text responses, 11 participants each cited "response-onset timing" and "observational cues," and 9 cited "utterance-intent understanding" as areas for improvement. The difference in the SSQ total symptom score before and after use was not significant ($p = 0.180$). These findings were treated as exploratory perception and symptom reports based on a single-session experience. Details are shown in Supplementary Tables S4–S6.

6. Discussion

6.1. Principal Findings and Evidence Boundaries

In this study, we distinguished between original participant sessions and post-study evaluations to assess operational behavior, control mechanisms, semantic concordance, and local control sensitivity. Mechanism-specific evaluations characterized tested operating boundaries but did not demonstrate the superiority of the overall architecture. The most significant additional finding was that semantic-label concordance and local control sensitivity varied greatly depending on label type and phase.

6.2. Semantic Concordance and Phase-Dependent Control Sensitivity

Although ActionClass and PhaseCandidate are derived from the same LLM-based post-response analysis, their concordance with expert references differed substantially. For ActionClass, strict agreement between the LLM and each expert was 74.5% for Expert 1 and 77.8% for Expert 2; in the subset where both experts assigned the same label, 83.1% of the results matched that common label. On the other hand, class-wise performance was uneven, with recall for the less frequent Classes 1–3 ranging from 25.8% to 43.5%. Therefore, ActionClass cannot be regarded as uniformly accurate based solely on the overall agreement rate.

For PhaseCandidate, strict agreement between the LLM and each expert was 16.9% for Expert 1 and 35.2% for Expert 2, and agreement between the two experts was also

lower than that for ActionClass. Furthermore, even when limited to the 339 cases where both experts assigned the same PhaseCandidate, the LLM matched that common label only 21.5% of the time. These results suggest that the low concordance of PhaseCandidate cannot be explained by expert disagreement alone, and that there is room for improvement in the prompt, codebook, label definition, or the classification task design itself. Since this study did not establish an adjudicated gold standard, these values should be interpreted as reference-specific concordance rather than clinical accuracy.

The impact of semantic-label disagreement on scene control was also not uniform. In the 526 turns where the guard was active, the local outcome divergence when only ActionClass was replaced with the expert label was 3.6% and 1.9% for Expert 1 and Expert 2, respectively, whereas replacing only PhaseCandidate resulted in divergences of 34.6% and 20.2%. In this observed dataset, the semantic sensitivity of local phase decisions was more strongly associated with differences in PhaseCandidate than with differences in ActionClass.

Furthermore, in the 305 turns where the two experts differed on at least one semantic label, the proportion of cases where the expert-specific local guard outcomes also differed was 43.6%; however, this varied markedly across phases: 85.2% in Phase 1, 1.9% in Phase 2, and 56.8% in Phase 3. This phase dependence is consistent with the implemented guard structure and the observed state distribution. In Phase 1 and Phase 3, semantic conditions can directly satisfy transition conditions as part of multiple OR conditions. In contrast, Phase 2 requires the conjunction of `InMotherArms` and the mother-avatar breastfeeding preparation state. Although the latter can be updated by ActionClass, the transition from Phase 2 to Phase 3 cannot occur based on the semantic label alone. Under this structure, the expert-specific local outcome divergence in our observed dataset was notably low in Phase 2 and high in Phases 1 and 3.

However, this result does not indicate that deterministic guards in general prevent LLM errors. Rather, in this observed dataset, the extent to which semantic variability is reflected in local control decisions varied markedly across phases, and these differences corresponded to how semantic evidence and physical state were combined in each phase, as well as to the observed state distribution. From a system-reliability perspective, guard mediation should not be equated with attenuation of semantic uncertainty. Although Unity retains final execution authority, `PhaseCandidate` ≥ 2 in Phase 1 and `PhaseCandidate` ≥ 4 in Phase 3 can each satisfy an OR-based transition guard without corroborating evidence from the physical VR state or the mother-avatar state. Therefore, when `PhaseCandidate` does not represent the intended scene phase, it may permit a premature or semantically unsupported local transition. The present analysis quantifies one-step trace-conditioned sensitivity; it does not estimate the frequency of inappropriate transitions in the participant sessions or evaluate their downstream trajectories. Accordingly, the reliability of phase-transition decisions under semantic uncertainty has not been established. Future work should prospectively evaluate guard topologies in which `PhaseCandidate` cannot by itself enable a transition and instead requires corroborating state evidence.

6.3. Mechanism-Specific Findings for Asynchronous Control

While Turn-ID fault injection supported cross-turn stale-result exclusion, same-turn duplicate callbacks were not excluded. Therefore, the scope of the Turn-ID guarantee should be interpreted as limited to cross-turn stale-result exclusion and cannot be regarded as a general idempotency mechanism.

Regarding apply-time state binding, while there were turns where differences in the selected physical state were observed, no differences in the local next-phase outcome due to the request-proxy state and the apply-time state were observed in the observed traces.

Therefore, this study did not demonstrate an additional local decision benefit of apply-time binding over earlier-state binding, and prospective evaluation is needed to identify the conditions under which such a benefit may emerge.

The 1600 out of 1600 matches in phase-guard oracle testing support implementation consistency within the defined discrete test grid; however, they do not constitute formal verification and do not cover unobserved runtime states or clinical safety.

6.4. Logging, Reproducibility, and Traceability

The original participant-study logs lacked a common runtime primary key that linked conversation, training-flow, and pose logs end-to-end. Therefore, for 19 sessions, conversation logs were matched to training-flow logs based on recording order, and pose/operator events were referenced via Unity frame indices. While this method enabled post hoc analysis, it did not provide complete event-level provenance for the original data.

Given this limitation, we implemented a revised common-ID logging schema post-study that shares provenance at the session, turn, request, callback, and semantic-event levels. All 10 predefined integration checks passed, and for synthetic traces, we were able to deterministically join component logs on a turn/request basis. However, this is a verification of the revised implementation and does not retroactively improve the traceability of the original 20 sessions.

The ordered event sequences comprising 51 PHASE_CHANGE events and 10 TRAINING_DONE events matched between the two analyses. This analysis demonstrates regression consistency between saved top-level states and explicitly defined reduction rules; it does not demonstrate formal equivalence or complete record/replay of the Unity runtime implementation.

6.5. Implications for Nursing and Midwifery VR Training

The participant sessions in this study were conducted as a technical human-in-the-loop pilot to confirm that free dialogue and the grasping and positioning of newborns could be combined within the same VR scene. While user reports showed many positive responses regarding the usefulness of the training and the intention to continue using the system, requests for improvements were noted regarding utterance-intent understanding and the timing of response initiation. While these serve as useful exploratory feedback for improving future interface and dialogue design, they do not constitute evidence of learning effectiveness or clinical competence.

This study is not designed to evaluate educational or clinical effectiveness. In the future, it will be necessary to separate technical verification from educational and clinical validation and to use objective performance measures, validated educational outcomes, and evaluations by domain experts.

6.6. Limitations

This study has several limitations.

First, this study is a single-session technical human-in-the-loop pilot involving 20 nursing students, a single breastfeeding support scenario, and a single system configuration. It did not evaluate a control group, randomization, pre- and post-assessments of educational outcomes, clinical performance, retention, or transfer to real-world clinical practice. Therefore, it does not demonstrate educational effectiveness, clinical effectiveness, or generalizability to other learners, scenarios, or system configurations.

Second, there were limitations regarding the concordance between LLM-derived semantic labels and expert references. In particular, PhaseCandidate showed low concordance with two expert references, and for ActionClass, recall for low-frequency classes was low. Furthermore, since Expert 1 is a co-author of this paper and Expert 2 is a non-author evalu-

ator who is a colleague of Expert 1, this evaluation does not constitute external validation that is completely independent of the research team. Since no adjudicated gold standard has been established, the reported agreement should be interpreted as reference-specific concordance rather than clinical accuracy. Furthermore, since the post-response semantic analysis takes both the learner's utterance and the generated patient response as inputs, the semantic control input may be influenced not only by the learner's intent but also by model-generated content.

Third, mechanism-specific evaluations of guard, Turn-ID, and common-ID logging consist of deterministic or synthetic software tests conducted after the participant study concluded. While these evaluate implementation behavior with respect to specific failure modes, they do not estimate the actual occurrence rates of stale callbacks or concurrency faults in the original participant sessions, the causal benefits of each mechanism, or their superiority over alternative architectures. The tests also showed that the Turn-ID gate does not eliminate same-turn duplicate semantic callbacks, and general idempotency is not guaranteed.

Fourth, trace-conditioned local guard sensitivity is a one-step local recomputation with the observed apply-frame state fixed. Importantly, the low reference-specific concordance of `PhaseCandidate` is relevant to system reliability because `PhaseCandidate` can independently satisfy the Phase 1→2 and Phase 3→4 guards; deterministic execution therefore does not by itself prevent semantic variability in this input from affecting local guard outcomes. We did not annotate the appropriateness of the observed transitions or evaluate their downstream trajectories. It is not a counterfactual simulation that regenerates subsequent participant behavior, future scene trajectories, educational consequences, or clinical harm when using expert labels. Therefore, this study does not estimate the frequency of inappropriate transitions or establish the reliability of phase-transition decisions under semantic uncertainty. The reported divergence rates cannot be interpreted as clinical risk probabilities.

Fifth, the original logs lacked a unified turn/request primary key shared across components, contained 75 missing MessageIDs, and in one session, the conversation log and training-flow log could not be uniquely reconciled post hoc. Although the revised common-ID schema was validated in post-study integration tests, it does not retroactively repair the original participant-session logs.

Sixth, the latency metric is the software interval from the Unity main-thread commit of the finalized utterance text to immediately before the AudioSource.Play() call. It does not include the acoustic endpoint of the learner's speech, the portion of cloud STT processing that occurs before text commit, or the physical audio onset from the speaker. Furthermore, since we did not record network round-trip time, packet loss, jitter, CPU/GPU load, full-frame timing, or deadline misses, we cannot decompose observed latency variation into network, cloud service, hardware, or Unity processing components. Since there were cases where complete timing endpoints could not be obtained, this missing data must be taken into account when interpreting the distribution tail.

Seventh, auxiliary trace-driven reduced event recalculation does not use the same execution model as Unity runtime code; rather, it is an observed-trace consistency analysis that applies reduction rules to stored high-level state events. This consistency is not formal verification and does not cover unobserved state combinations, multiple events within the same frame, memory visibility, atomicity, callback ordering, or other unobserved runtime behaviors.

Finally, the clinical appropriateness of the generated patient responses themselves, the content validity of the scenarios, safety during long-term repeated use, and deployment-level privacy and security were outside the scope of this study's evaluation. These as-

pects require independent validation prior to practical implementation as a medical training application.

7. Conclusions

In this study, we applied a guard-mediated LLM-VR scene-control implementation—which integrates LLM-derived semantic evidence obtained from spontaneous speech, the VR state updated by learner actions and physics, and the mother-avatar state updated within a predefined range by ActionClass—to breastfeeding support training under phase-specific deterministic guards. The LLM-derived labels themselves did not directly commit phase transitions; final transition authority remained with Unity.

While mechanism-specific evaluations supported guard implementation consistency and cross-turn stale-result exclusion, observed traces did not reveal any additional local decision benefits from apply-time binding. Separately, the semantic concordance and local guard sensitivity analyses showed that the extent to which semantic variability is reflected in local phase decisions varies across phases in a manner consistent with the observed state distribution and phase-specific guard structure.

These results suggest that the use of deterministic guards alone cannot uniformly prevent semantic-label variability from propagating to local control decisions, and that the guard topology—which determines the authority assigned to each semantic input and its relationship to the physical state—can influence local decision sensitivity. In particular, the low PhaseCandidate concordance and high local sensitivity in Phases 1 and 3 observed in this study support the need to reexamine the current guard topology, which uses PhaseCandidate as a standalone transition-enabling disjunct. This study was a technical human-in-the-loop pilot and does not demonstrate educational effectiveness, clinical effectiveness, clinical safety, or superiority over alternative architectures.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app16178591/s1: Table S1: Key Functional Components of the Proposed System; Table S2: Definitions and Examples of Speech Categories Used for Input Validation; Table S3: Summary of Runtime Prompts and Their System Uses; Table S4: Results by Item for the VR Educational Material Evaluation Questionnaire; Table S5: Categorization of Free-Text Responses by Theme; Table S6: Comparison of VR-Related Physical Symptoms Before and After Use; Table S7: Design of the Full-Sample Expert Reference Evaluation of LLM-Derived Semantic Labels; Table S8: Full-Sample Inter-Expert Agreement and Overall Concordance of LLM-Derived Semantic Labels with Each Expert Reference; Table S9: Full-Sample Confusion Matrices and Class-Wise Performance for LLM-Derived ActionClass; Table S10: Full-Sample Confusion Matrices and Class-Wise Performance for LLM-Derived PhaseCandidate; Table S11: LLM Concordance in Expert-Agreement and Expert-Disagreement Subsets; Table S12: Analysis Populations and Denominators for the Full-Sample Semantic Evaluation and Trace-Conditioned Local Guard Analyses; Table S13: Trace-Conditioned Local Guard Sensitivity to Replacement of LLM-Derived Semantic Labels with Expert Reference Labels; Table S14: Phase-Specific Propagation of Inter-Expert Semantic Disagreement to Local Guard Outcomes; Table S15: Post-Study Mechanism-Specific Software Tests for Phase Guards and Turn-ID Handling; Table S16: Observed-Trace State-Binding Sensitivity and Post-Study Verification of the Revised Common-ID Logging Schema; Table S17: State Variables, Held Semantic Labels, and Frame Inputs for the Unity Runtime Scene Model; Table S18: Update Rules, Phase-Transition Conditions, and Rules for the Completion-Entry Event During Unity Execution; Table S19: Trace-Driven Reduced Event Recalculation: Input/Output Boundary, Observed-Sequence Agreement, and Differences from the Unity Runtime Model; Table S20: Data Collection Settings and Profiles for Execution, Missing Values, Log Consolidation, and Time-Base Profiles. The current supplementary package contains Tables S1–S20 only. Data and code availability are described in the Data Availability Statement.

Author Contributions: Conceptualization, N.H. and K.I.; Methodology, N.H. and K.I.; Software, N.H.; Validation, N.H. and K.I.; Formal Analysis, N.H.; Investigation, N.H. and K.I.; Data Curation, N.H.; Writing—Original Draft Preparation, N.H.; Writing—Review and Editing, N.H. and K.I.; Visualization, N.H.; Supervision, K.I.; Project Administration, N.H. and K.I.; Funding Acquisition, K.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Japan Society for the Promotion of Science (JSPS) KAKENHI, grant number JP22K10926. The funder had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki and approved by the Ethics Subcommittee of the Department of Nursing, Fukui Prefectural University (Review No. M2024006; approved on 8 July 2024).

Informed Consent Statement: Written informed consent was obtained from all participants involved in the study.

Data Availability Statement: The participant-level raw data collected in this study include spoken audio and utterance text, dialogue and behavior logs, free-text responses, body-motion trajectories, and first-person video from the HMD wearer's viewpoint. These data are not publicly available because of the scope of participant consent and the risk of re-identification. De-identified examples and aggregate results are included in the main text and Supplementary Tables S1–S20. The complete Unity project is not publicly available because it contains commercially licensed 3D assets and other third-party materials. The author-created Unity C# scripts, Python scripts for offline recomputation, original Japanese runtime prompt files, synthetic input traces, and related configuration information, excluding third-party licensed materials and API credentials, are not publicly posted but are available from the corresponding author upon reasonable request.

Acknowledgments: We would like to express our gratitude to the nursing students who participated in this study, as well as to the faculty and staff who assisted with participant recruitment and data collection. We would like to express our deep gratitude to Aoi Makino of the Faculty of Nursing & Social Welfare Sciences at Fukui Prefectural University for her cooperation in the expert evaluation of ActionClass and PhaseCandidate in this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. World Health Organization Regional Office for Europe. *Simulation in Nursing and Midwifery Education*; World Health Organization Regional Office for Europe: Copenhagen, Denmark, 2018.
2. Foronda, C.L.; Fernandez-Burgos, M.; Nadeau, C.; Kelley, C.N.; Henry, M.N. Virtual Simulation in Nursing Education: A Systematic Review Spanning 1996 to 2018. *Simul. Healthc.* **2020**, *15*, 46–54. [CrossRef] [PubMed]
3. Shorey, S.; Ng, E.D. The Use of Virtual Reality Simulation among Nursing Students and Registered Nurses: A Systematic Review. *Nurse Educ. Today* **2021**, *98*, 104662. [CrossRef] [PubMed]
4. Saab, M.M.; McCarthy, M.; O'Mahony, B.; Cooke, E.; Hegarty, J.; Murphy, D.; Walshe, N.; Noonan, B. Virtual Reality Simulation in Nursing and Midwifery Education: A Usability Study. *Comput. Inform. Nurs.* **2023**, *41*, 815–824. [CrossRef] [PubMed]
5. Lee, J.; Kim, H.; Kim, K.H.; Jung, D.; Jowsey, T.; Webster, C.S. Effective Virtual Patient Simulators for Medical Communication Training: A Systematic Review. *Med. Educ.* **2020**, *54*, 786–795. [CrossRef] [PubMed]
6. Chaby, L.; Benamara, A.; Pino, M.; Prigent, E.; Ravenet, B.; Martin, J.C.; Vanderstichel, H.; Becerril-Ortega, R.; Rigaud, A.S.; Chetouani, M. Embodied Virtual Patients as a Simulation-Based Framework for Training Clinician-Patient Communication Skills: An Overview of Their Use in Psychiatric and Geriatric Care. *Front. Virtual Real.* **2022**, *3*, 827312. [CrossRef]
7. Jo, Y.W.; Lee, M.; Yang, H.J. Large Language Model-Based Virtual Patient Simulations in Medical and Nursing Education: A Review. *Appl. Sci.* **2025**, *15*, 11917. [CrossRef]
8. Cook, D.A.; Overgaard, J.; Pankratz, V.S.; Del Fiol, G.; Aakre, C.A. Virtual Patients Using Large Language Models: Scalable, Contextualized Simulation of Clinician-Patient Dialogue With Feedback. *J. Med. Internet Res.* **2025**, *27*, e68486. [CrossRef] [PubMed]

9. Weisman, D.; Sugarman, A.; Huang, Y.M.; Gelberg, L.; Ganz, P.A.; Comulada, W.S. Development of a GPT-4–Powered Virtual Simulated Patient and Communication Training Platform for Medical Students to Practice Discussing Abnormal Mammogram Results With Patients: Multiphase Study. *JMIR Form. Res.* **2025**, *9*, e65670. [CrossRef] [PubMed]
10. He, H.; Xu, X.; Li, S.; Bueno-Vesga, J.A.; Duan, Y.; Gu, Y. Training Nursing Skills in a Generative Artificial Intelligence-Enhanced Virtual Reality Patient Encounter Simulation: A Qualitative Study from a Student Perspective. *Int. J. Educ. Technol. High. Educ.* **2026**, *23*, 12. [CrossRef]
11. Geng, X.; Eto, M.; Yamada, M. Generative AI-Driven Virtual Reality for Health Consultation Training in School Nursing. In *Virtual, Augmented and Mixed Reality*; Chen, J.Y.C., Fragomeni, G., Eds.; Springer: Cham, Switzerland, 2026; Lecture Notes in Computer Science; Volume 16717, pp. 58–76. [CrossRef]
12. Tang, K.; Gerling, K.; Geurts, L. Virtual Feed: Design and Evaluation of a Virtual Reality Simulation Addressing the Lived Experience of Breastfeeding. In Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '22), New York, NY, USA, 29 April–5 May 2022; pp. 1–17. [CrossRef]
13. Sadovnikova, A.; Chuisano, S.A.; Ma, K.; Grabowski, A.; Stanley, K.P.; Mitchell, K.B.; Eglash, A.; Plott, J.S.; Zielinski, R.E.; Anderson, O.S. Development and Evaluation of a High-Fidelity Lactation Simulation Model for Health Professional Breastfeeding Education. *Int. Breastfeed. J.* **2020**, *15*, 8. [CrossRef] [PubMed]
14. Grabowski, A.; Chuisano, S.A.; Strock, K.; Zielinski, R.; Anderson, O.S.; Sadovnikova, A. A Pilot Study to Evaluate the Effect of Classroom-Based High-Fidelity Simulation on Midwifery Students' Self-Efficacy in Clinical Lactation and Perceived Translation of Skills to the Care of the Breastfeeding Mother-Infant Dyad. *Midwifery* **2021**, *102*, 103078. [CrossRef] [PubMed]
15. Grabowski, A.; Anderson, O.S.; Zielinski, R.; Scott, M.; Hammer, L.; Bassil, M.; Chuisano, S.A.; Sadovnikova, A. Midwifery Students Better Approximate Their Self-Efficacy in Clinical Lactation after Reflecting in and on Their Performance in the LactSim OSCE. *Adv. Simul.* **2020**, *5*, 28. [CrossRef] [PubMed]
16. Schlangen, D.; Skantze, G. A General, Abstract Model of Incremental Dialogue Processing. *Dialogue Discourse* **2011**, *2*, 83–111. [CrossRef]
17. Skantze, G. Turn-Taking in Conversational Systems and Human-Robot Interaction: A Review. *Comput. Speech Lang.* **2021**, *67*, 101178. [CrossRef]
18. Kopp, S.; Krenn, B.; Marsella, S.; Marshall, A.N.; Pelachaud, C.; Pirker, H.; Thórisson, K.R.; Vilhjálmsson, H. Towards a Common Framework for Multimodal Generation: The Behavior Markup Language. In *Proceedings of the Intelligent Virtual Agents*; Springer: Cham, Switzerland, 2006; Lecture Notes in Computer Science; Volume 4133, pp. 205–217.
19. Thiebaux, M.; Marsella, S.; Marshall, A.N.; Kallmann, M. SmartBody: Behavior Realization for Embodied Conversational Agents. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, Estoril, Portugal, 12–16 May 2008; Volume 1, pp. 151–158.
20. Harel, D. Statecharts: A Visual Formalism for Complex Systems. *Sci. Comput. Program.* **1987**, *8*, 231–274. [CrossRef]
21. Chow, T.S. Testing Software Design Modeled by Finite-State Machines. *IEEE Trans. Softw. Eng.* **1978**, *SE-4*, 178–187. [CrossRef]
22. LeBlanc, T.J.; Mellor-Crummey, J.M. Debugging Parallel Programs with Instant Replay. *IEEE Trans. Comput.* **1987**, *C-36*, 471–482. [CrossRef]
23. Pan, M.; Kitson, A.; Wan, H.; Prpa, M. ELLMA-T: An Embodied LLM-Agent for Supporting English Language Learning in Social VR. In Proceedings of the 2025 ACM Designing Interactive Systems Conference, New York, NY, USA, 5–9 July 2025; pp. 576–594. [CrossRef]
24. El Hajji, M.; Ait Baha, T.; Berka, A.; Ait Nacer, H.; El Aouifi, H.; Es-Saady, Y. An Architecture for Intelligent Tutoring in Virtual Reality: Integrating LLMs and Multimodal Interaction for Immersive Learning. *Information* **2025**, *16*, 556. [CrossRef]
25. Geris, A.; Alce, G. Real-Time Voice-Based LLM Integration for XR Tutoring: A Prototype Implementation. In *Proceedings of the 2026 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*; IEEE: New York, NY, USA, 2026; pp. 285–289. [CrossRef]
26. Mikkonen, K.; Butkeviciute, E.; Armalis, P.; Aluzaite, E.; Baranauskaite, R.; Paulauskas, L.; Spirgiene, L.; Subocius, A.; Blazauskas, T.; Riklikiene, O. Designing Safe Hybrid AI–XR Simulations for Healthcare Communication and Interaction Training. In *Digital Health and Wireless Solutions: Towards Trustworthy and Person-Centric Digital Health*; Sarestoniemi, M., Singh, D., Jarva, E., Reponen, J., Eds.; Springer: Cham, Switzerland, 2026; Communications in Computer and Information Science; Volume 3009, pp. 289–299. [CrossRef]
27. Su, Y. Voice2Action: Language Models as Agent for Efficient Real-Time Interaction in Virtual Reality. *arXiv* **2023**, arXiv:2310.00092. [CrossRef]
28. Hu, X.; Ma, D.; He, F.; Zhu, Z.; Hsia, S.K.; Zhu, C.; Liu, Z.; Ramani, K. GesPrompt: Leveraging Co-Speech Gestures to Augment LLM-Based Interaction in Virtual Reality. In Proceedings of the 2025 ACM Designing Interactive Systems Conference, New York, NY, USA, 5–9 July 2025; pp. 59–80. [CrossRef]
29. Nechesov, A.; Ruponen, J. INTELLECTUM: A Hybrid AR-VR Metaverse Framework for Smart Cities. *Appl. Syst. Innov.* **2026**, *9*, 61. [CrossRef]

30. World Health Organization; United Nations Children's Fund (UNICEF). *Protecting, Promoting and Supporting Breastfeeding in Facilities Providing Maternity and Newborn Services: Implementing the Revised Baby-Friendly Hospital Initiative 2018*; World Health Organization and United Nations Children's Fund: Geneva, Switzerland, 2018.
31. World Health Organization; United Nations Children's Fund (UNICEF). *Competency Verification Toolkit: Ensuring Competency of Direct Care Providers to Implement the Baby-Friendly Hospital Initiative*; World Health Organization and United Nations Children's Fund: Geneva, Switzerland, 2020.
32. World Health Organization. *Infant and Young Child Feeding Counselling: An Integrated Course: Trainer's Guide*, 2nd ed.; World Health Organization: Geneva, Switzerland, 2021.
33. Eppich, W.; Cheng, A. Promoting Excellence and Reflective Learning in Simulation (PEARLS): Development and Rationale for a Blended Approach to Health Care Simulation Debriefing. *Simul. Healthc.* **2015**, *10*, 106–115. [CrossRef] [PubMed]
34. Obeid, M.F.; Ewais, A.; Asia, M.R. NursingXR: Advancing Nursing Education Through Virtual Reality-Based Training. *Appl. Sci.* **2025**, *15*, 2949. [CrossRef]
35. Alvarado-Robles, B.; Rodriguez-Ramirez, A.G.; Luviano-Cruz, D.; Ortiz-Muñoz, D.; Alonso-Mendoza, V.M.; Garcia-Luna, F. Technology Assessment Models in Healthcare Education: An Integrative Review and Future Perspectives in the Era of AI and VR. *Appl. Sci.* **2026**, *16*, 1213. [CrossRef]
36. Kennedy, R.S.; Lane, N.E.; Berbaum, K.S.; Lilienthal, M.G. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *Int. J. Aviat. Psychol.* **1993**, *3*, 203–220. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.