TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: Maestría Profesional en Ingeniería del Software con Énfasis en Arquitectura y Diseño Soft**,** requisito para optar por el título de grado de Maestría Profesional en Ingeniería del Software con Énfasis en Arquitectura y Diseño Soft**,** el estudiante Orlando Alexander Gatica Valle.

DAVID GERARDO ALFARO VIQUEZ (FIRMA)
PERSONA FISICA, CPF-01-1337-0927.
Fecha declarada: 17/08/2026 09:06:02 PM
Es una representación gráfica,
verifique validez de la firma.

M.Sc. David Gerardo Alfaro Víquez
Tutor

CARMEN MARIA MOK ZHENG (FIRMA)
Digitally signed by CARMEN MARIA MOK ZHENG (FIRMA)
Date: 2026.08.17 22:03:52 -06'00'

M.Sc. Carmen María Mok Zheng
Lector 1

IGNACIO TREJOS ZELAYA (FIRMA)
Firmado digitalmente por IGNACIO TREJOS ZELAYA (FIRMA)
Fecha: 2026.08.18 18:15:52 -06'00'

M.Sc. Ignacio Trejos Zelaya
Lector 2

San José, Costa Rica, 17 de Agosto 2026

A Multimodal Software Architecture for Human Digital Twins in Human-Machine Interaction: Design and Proof-of-Concept Evaluation

Orlando Alexander Gatica Valle
Universidad CENFOTEC
San Jose, Costa Rica
ogaticav@ucenfotec.ac.cr

David Gerardo Alfaro Viquez
Universidad CENFOTEC
San Jose, Costa Rica
dalfarov1@ucenfotec.ac.cr

***Abstract*—Human Digital Twins require software architectures that can integrate heterogeneous data sources while preserving interoperability, traceability, synchronization, and controlled evolution. This paper proposes a multimodal software architecture for Human Digital Twins in human-machine interaction scenarios. The proposal was derived from a literature-based analysis and refined through Attribute Driven Design. The architecture organizes the solution into layers for multimodal ingestion, normalization and synchronization, integration and orchestration, Human Digital Twin services, persistence and traceability, and external consumption. A prototype was evaluated through six controlled experiments and a second evaluation stage with selected subsets of the WESAD and HA4M datasets. The controlled experiments covered nominal operation, high load, alarm behavior, video extension, degraded capture, and noisy physiological data. WESAD introduced real physiological signals while preserving the two-modality baseline of 160 persisted events, eight windows, eight snapshots, and no incidents. HA4M was evaluated in two stages: traceable references to real RGB frames and simple frame-derived indicators. Its final stage preserved the three-modality baseline of 240 events, eight windows, eight snapshots, and no incidents. The results indicate that the architecture can support multimodal integration, real-data source adaptation, localized extension, and operational continuity within the evaluated scope.**

***Index Terms*—Human Digital Twins, Software Architecture, Multimodal Data Integration, Human-Machine Interaction, Industry 5.0.**

I. Introduction

Digital twins are connected digital representations of physical systems, processes, or entities. In cyber-physical systems and manufacturing, they support monitoring, simulation, analysis, and data-driven decision support through data flows between the physical and digital environments [1], [2]. In recent years, this concept has been extended toward Human Digital Twins (HDTs), where the digital representation is centered on a human operator, worker, or user. In this context, the twin may integrate physiological signals, contextual information, visual data, operational events, and other indicators that help represent the human state during interaction with machines or software-intensive environments [3], [4].

HDTs fit the Industry 5.0 emphasis on human-centered systems, collaboration, resilience, and responsible automation [5], [6]. In human-machine interaction, an HDT can support adaptive interfaces, early detection of risky conditions, traceability of operational events, and feedback between the physical and digital environments. For that reason, constructing an HDT is not only a data science or sensing problem. It is also a software architecture problem.

The literature review conducted for this study found that many HDT proposals remain domain-specific, conceptual, or centered on a particular model or technology. Several works discuss layered structures, edge-cloud deployments, service-oriented approaches, or specific multimodal models [7]–[10]. Those contributions clarify important parts of the domain, but they do not fully resolve the architectural problem of integrating heterogeneous sources, synchronizing events, preserving traceability, exposing the HDT state, and allowing the system to grow without redesigning the core.

This paper addresses that gap by proposing and evaluating a multimodal software architecture for HDTs in human-machine interaction scenarios. The study does not aim to build a complete industrial HDT platform or an advanced artificial intelligence model. Its scope is narrower: to define an implementable architecture for multimodal ingestion, normalization, synchronization, persistence, traceability, and consumption, and to evaluate that architecture through a controlled prototype.

The main contributions of this research are:

1) A multimodal software architecture for Human Digital Twin development that addresses data heterogeneity and source extensibility through a layered, service-oriented design with a canonical event contract and localized source adapters.
2) An architecture-driven design process using Attribute Driven Design, in which interoperability, synchronization, extensibility, traceability, and maintainability are treated as explicit quality drivers and connected to implementable structural decisions.
3) A prototype evaluation combining six controlled experiments with WESAD physiological input and a two-stage HA4M visual integration, providing evidence that the architecture can preserve traceability and incorporate real-data sources with localized structural impact.

The remainder of this paper is organized as follows. Section II summarizes the related work derived from the literature review. Section III presents the proposed framework and its main architectural decisions. Section IV describes the controlled and real-dataset evaluation, reports the results, and discusses their meaning. Section V concludes the paper and outlines future work.

II. Related Work

This section summarizes the literature basis used to define the architectural problem. It does not present a full systematic literature review; instead, it identifies the concepts, architectural tendencies, and limitations that justify the proposed framework.

A. Review Process

The related work was derived from the literature review conducted for this study, following general guidance for systematic reviews in software engineering [11]. The review combined an expert-provided list with Google Scholar searches. Google Scholar was used as the entry point because it indexes results from multiple publishers and repositories, while the expert list reduced the risk of omitting relevant work from the HDT and human-robot collaboration domain. The protocol considered studies in English and Spanish, with a recommended temporal focus on the last four years and justified exceptions for critical or high-impact studies needed to establish the architectural basis.

The search focused on terms such as *Digital Twin*, *Human Digital Twin*, *multimodal data*, *software architecture*, *human-machine interaction*, *human-robot collaboration*, and *Industry 5.0*. Search strings combined "human digital twin" or "human-centric digital twin" with terms such as "software architecture", "reference architecture", "framework", "multimodal", "sensor fusion", "heterogeneous data", "data integration", "interoperability", "synchronization", "data quality", "edge computing", "cloud-native", "microservices", and "event-driven". The objective was to recover studies that discussed architectural structures, multimodal integration, quality attributes, or validation mechanisms.

The filtering process was applied in stages: title and keyword screening, abstract screening, full-text reading, and final quality assessment. Studies were included when they addressed HDT or an equivalent human-centered digital twin concept, described architecture or system-level integration, considered multimodal data or related integration concerns, and provided enough technical detail for extraction. Papers were excluded when they addressed digital twins without a human-centered or human-machine interaction scope, were purely algorithmic without architectural contribution, were not peer reviewed, were duplicates, or did not provide sufficient detail. The selection process started from 28 expert-list studies and 12 Google Scholar studies. After filtering, 14 expert-list studies and 5 Google Scholar studies were retained, for a final set of 19 studies. These studies were used to extract recurring data flows, architectural styles, quality attributes, validation practices, and limitations; the review itself is therefore an input to the design rather than the main research contribution.

B. Digital Twins

Digital twins are digital representations connected to physical entities through data flows. In manufacturing and cyber-physical systems, they are commonly used for monitoring, simulation, predictive maintenance, process analysis, and decision support [1], [2]. A digital twin is therefore not only a model, but also a software-intensive mechanism that connects data capture, representation, persistence, analysis, and feedback.

Business ecosystem studies also show that digital twins can support coordination across actors, data sources, and operational processes [12]. This broader view is relevant for software architecture because the twin normally depends on multiple components with different responsibilities. The architecture must define how data enters the system, how it is transformed, where it is persisted, and how the resulting state is exposed to consumers.

C. Human Digital Twins

Human Digital Twins specialize the digital twin idea around human-related data and human-centered behavior. Recent work places HDTs in the context of Industry 5.0, human factors, digital human modeling, collaborative work cells, and operator support [3], [4], [13]. In these cases, the digital representation may include physiological signals, motion data, visual information, context, cognitive indicators, or operational events.

HDTs are especially relevant in human-machine interaction because the human state is dynamic and context-dependent. A useful HDT must integrate heterogeneous sources and update a representation that other systems can consume. This makes HDT development an architectural problem as well as a modeling problem. Without a clear architecture, each modality can remain isolated in a specific adapter, model, or experiment, reducing traceability and making later extension difficult.

D. Industry 5.0 Context

Industry 5.0 emphasizes human-centered, collaborative, and resilient systems. The reviewed literature connects this vision with digital twins, collaborative robotics, operator support, human-centric ecosystems, and skilled-operator assistance [5], [6], [14], [15]. From this perspective, HDTs can support systems that adapt to the operator, improve awareness of human state, and provide better feedback between humans and machines.

The architecture proposed in this paper is intentionally general. It is not limited to a clinical, manufacturing, or robotics-specific implementation. This generality is aligned with the research objective: to design a software architecture that can serve as a base for multimodal HDT development in human-machine interaction scenarios. Industry 5.0 provides the motivation for this direction, but the architectural concern is broader: how to integrate and expose human-related multimodal data in a controlled and evolvable way.

E. Multimodal Integration

HDT scenarios may combine physiological signals, contextual information, visual data, interaction events, and operational telemetry. These modalities differ in structure, frequency, quality, reliability, and privacy sensitivity. For that reason, multimodal integration is a central challenge.

Several studies report the use of video, physiological data, motion data, cognitive indicators, and operational context. Chand et al. proposed a fatigue-sensitive HDT for human-robot collaboration using visual and physiological information [9]. Zhong et al. proposed a multimodal HDT model for locomotion mode identification [10]. You et al. addressed fatigue-aware task planning in collaborative assembly [16]. These works show the value of multimodal data, but they also reinforce the need to manage heterogeneous sources in a structured way.

Across the reviewed studies, the most common data flow can be summarized as capture, preprocessing or normalization, integration or feature extraction, twin update, analysis or decision, and feedback to the environment. This pattern appears in human-robot collaboration, additive manufacturing, predictive maintenance, and human-centered digital twin frameworks [2], [4], [7], [8], [13]. For a software architecture, this means that the pipeline must define where each responsibility belongs and how data can be traced across transformations.

F. Architectural Approaches

The reviewed literature shows recurring architectural directions. Layered frameworks separate the physical world, the digital representation, and the linking or service mechanisms [3], [4]. Service-oriented architectures are used to improve modularity, reuse, and interoperability, especially in manufacturing contexts [8]. Edge-cloud architectures are often used when latency and proximity to the source are important [7]. Other works explore cognitive models, multi-agent structures, or the use of large language models for temporal feature learning and interpretation [17], [18].

Table I summarizes the main architectural directions identified in the review and how they relate to the gap addressed by this paper.

These approaches provide useful foundations, but they are frequently presented as conceptual frameworks, domain-specific solutions, or technology-centered implementations. The literature also reports recurring limitations: lack of standard platforms, complex integration of heterogeneous data, synchronization challenges, limited availability of multimodal datasets, and limited industrial-scale validation [6], [19], [20].

The gap left by each direction is different. Layered frameworks provide useful separation of concerns, but they often do not define implementable event contracts or evidence mechanisms; this paper addresses that gap through a canonical event structure and persistent traces. Service-oriented approaches expose functionality through explicit interfaces, but they still require decisions about source adaptation, synchronization, and internal event flow; the proposed architecture combines service exposure with adapters and temporal windows. Edge-cloud architectures help with latency and proximity to sources, but they do not by themselves define how heterogeneous human-related events become a coherent HDT state; the proposed framework keeps edge-cloud deployment as a possible evolution while focusing the first implementation on integration and traceability. Cognitive or model-driven approaches improve fidelity in specific tasks, but they may depend on one model, dataset, or domain; this paper separates the architectural pipeline from any single model. Finally, LLM and multi-agent extensions add interpretation capabilities, but they do not remove the need for a stable multimodal integration architecture; the proposed framework treats those mechanisms as future consumers or services rather than as the architectural foundation.

G. Quality Attributes and Research Gap

The most relevant quality attributes identified in the review are interoperability, latency or synchronization, scalability, security, modularity, maintainability, traceability, and fidelity of the representation. Interoperability and safety are especially relevant in human-robot collaboration and human-centered ecosystems [3], [19]. Latency and scalability are emphasized in edge-cloud approaches [7]. Reusability and modularity are common motivations for service-oriented designs [8]. Traceability and interpretability become important when digital twins are used to explain or reconstruct decisions [18].

Given that HDT concepts, sensing technologies and ad-hoc implementation approaches were found to be well established, the gap addressed by this paper is the need for an architecture-level proposal that connects the main quality attributes with implementable components, canonical data contracts, multimodal flow, traceability mechanisms, and validation evidence. This motivates the proposed framework described in the next section.

III. Proposed Framework

The proposed framework is a multimodal software architecture for HDT development in human-machine interaction scenarios. It was designed as an applied software engineering artifact and refined through Attribute Driven Design (ADD) [21]. The design also follows an applied informatics orientation and the general logic of Design Science Research, where a problem is analyzed, an artifact is built, and the artifact is evaluated in context [22], [23]. During the design process, the main structural choices were documented through C4 views, Architecture Decision Records, and a canonical event contract for internal multimodal events.

A. Architectural Drivers

The design was guided by functional and quality drivers. In this context, a driver is a design force that must influence the structure of the architecture, not only a desirable property of the final system. The drivers were selected from four inputs. First, the related work showed recurring gaps around heterogeneous data integration, explicit architectural structure, and evaluation evidence for HDT systems. Second, the research objective

TABLE I
ARCHITECTURAL DIRECTIONS IDENTIFIED IN THE REVIEWED LITERATURE.

Direction	Main characteristics	Strengths	Limitations for multimodal HDT architecture
Layered frameworks	Separate physical entities, digital representation, and linking or service mechanisms.	Provide a clear separation of responsibilities and are easy to explain at architecture level.	Often remain conceptual and do not always specify implementable contracts, traceability, or validation mechanisms.
Service-oriented architectures	Organize functionality through services, interfaces, and reusable components.	Support modularity, reuse, and interoperability through explicit interfaces.	Require additional decisions for synchronization, event flow, and source-specific adaptation.
Edge-cloud architectures	Place part of the processing close to the source and part in cloud or centralized services.	Help address latency, scalability, and proximity to data sources.	Increase deployment complexity and may be excessive for an initial controlled prototype.
Cognitive or model-driven approaches	Focus on human behavior, cognitive state, fatigue, or domain-specific models.	Improve fidelity of the human representation in specific scenarios.	Can depend strongly on a particular model or dataset, reducing architectural generality.
LLM or multi-agent extensions	Use language models or agent structures for interpretation, temporal learning, or richer analysis.	Provide promising mechanisms for semantic interpretation and decision support.	Add complexity and do not remove the need for a stable multimodal integration architecture.

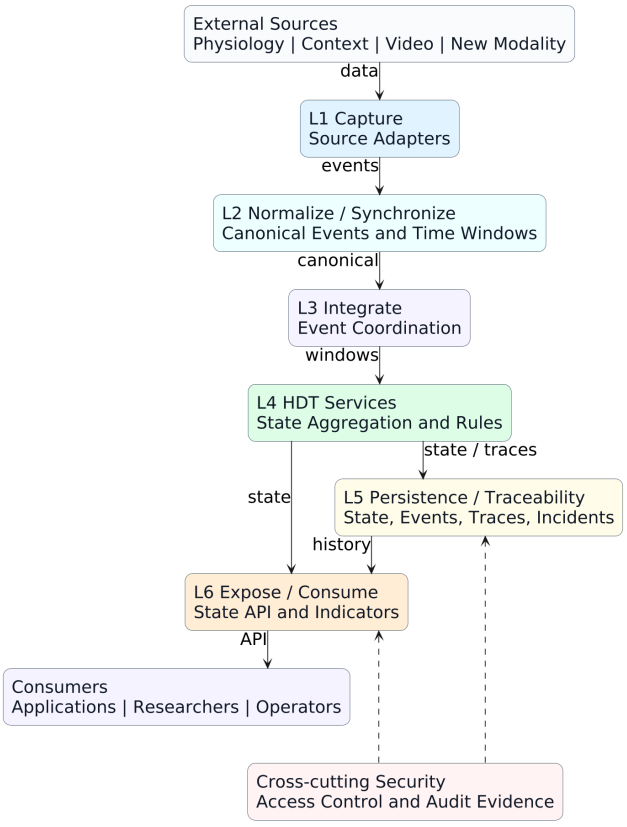

Fig. 1. Overview of the proposed multimodal HDT software architecture.

required an architecture oriented to interoperability and evolvable integration in human-machine interaction scenarios. Third, the controlled evaluation scope required drivers that could be observed through implementation evidence, such as event processing, persistence, traces, and localized extension. Fourth, the initial list was refined through ADD so that each driver could be connected to concrete architectural decisions.

The main functional drivers are multimodal capture, normalization of heterogeneous data, temporal synchronization, HDT state update, exposure of the consolidated state, and incorporation of new sources. The main quality drivers are interoperability, scalability and extensibility, latency and synchronization, security, maintainability and modularity, functional fidelity, and traceability. Table II summarizes how each quality driver influenced the design.

Interoperability and scalability were treated as primary drivers because they are directly related to multimodal HDT construction. Latency and synchronization were also considered high priority because an HDT that combines modalities without temporal coherence can produce misleading state updates. Security and maintainability were included as high-priority engineering concerns because the system handles human-related data and is expected to evolve. Functional fidelity and traceability kept the design connected to verifiable behavior: the system should preserve a coherent relationship between incoming observations and the represented HDT state, and it should retain enough execution records to review that behavior.

Following the structure used by Bass et al. for quality attribute scenarios [21], the ADD process formalized priority drivers as stimulus–environment–response measure scenarios. Table III summarizes the six scenarios defined during design; this table connects the design scenarios with the experimental evidence reported in Section IV.

B. Architectural Layers

The framework is organized into six main layers and one cross-cutting security concern. Figure 1 shows the architectural layers and the main flow between them. The choice is grounded in the literature review: prior HDT and digital twin works commonly separate physical sources, digital representation,

TABLE II
QUALITY DRIVERS USED IN THE PROPOSED FRAMEWORK.

Driver	Why it matters for HDT	Design response	Priority
Interoperability	HDT data may come from physiological, contextual, visual, and operational sources with different formats and semantics.	Source adapters and a canonical event contract reduce source-specific coupling after ingestion.	High
Scalability / extensibility	The architecture must support new modalities and services without forcing a complete redesign.	Localized adapters, separated layers, and service-oriented exposure limit the impact of changes.	High
Latency / synchronization	The HDT state is useful only if events from different modalities can be aligned in time.	Timestamped events, temporal windows, and asynchronous processing support consistent state updates.	High
Security	Human-related data and derived indicators require controlled exposure.	Access control and service-based interfaces avoid direct access to internal storage and processing components.	High
Maintainability / modularity	Multimodal systems evolve through changes in sources, rules, storage, and consumers.	The design separates capture, processing, services, persistence, and consumption responsibilities.	High
Functional fidelity	The consolidated state must remain coherent with the evidence received from the active sources.	Validation, state aggregation rules, and persisted traces connect source events with state snapshots.	Medium
Traceability / observability	Evaluation and troubleshooting require reconstruction of how events move through the pipeline.	Events, snapshots, incidents, and trace identifiers are persisted as first-class evidence.	Medium

TABLE III
PRIORITY QUALITY ATTRIBUTE SCENARIOS USED DURING ADD.

Scenario	Stimulus and environment	Expected response	Response measure / validation link
QC-01: new source integration	A new multimodal source with a different format or protocol must be incorporated during a prototype extension.	The architecture integrates the source through a localized adapter and mapping rules without modifying the HDT core or consumer-facing contracts.	The change remains confined to the adapter, source registration, and integration configuration; the public HDT interface is unchanged. This was exercised by the simulated video extension in experiment D (Tables VII and IX).
QC-02: temporal synchronization	Temporally misaligned events are received from different modalities during normal prototype operation.	The system normalizes, timestamps, and synchronizes the information before consolidating the HDT state.	Each event preserves source identity and timestamp metadata, and the alignment criterion remains verifiable in tests. The controlled runs produced matching windows and snapshots, with eight to nine closed windows depending on the scenario (Table VII).
QC-03: partial capture failure	A capture source stops emitting data or fails intermittently during degraded prototype operation.	The system isolates the failure, registers the incident, and continues operating with the available sources.	The failure does not stop the HDT core, and diagnostic evidence is retained. Experiments E and F continued producing snapshots while registering four and eight incidents, respectively (Table VII).
QC-04: authorized state query	An authorized consumer queries the current operator state or a derived indicator during normal prototype operation.	The system returns a stable and consistent state view while keeping the consumer decoupled from internal pipeline details.	The interface returns consolidated HDT state and relevant metadata without requiring direct access to internal storage; protected API checks are summarized in Table VIII.
QC-05: basic protection of human data	An unauthorized user or component attempts to access restricted data or services during normal operation or a basic security test.	The system denies access and avoids exposing protected HDT state through the service interface.	The prototype validation covered rejection without an API key and successful access with a valid key; the evidence scope remains a basic API-key check (Table VIII).
QC-06: pipeline observability	A researcher or developer needs to reconstruct the path of a datum from capture to consumption during test execution or incident analysis.	The system allows the event, its transformations, and its effect on the HDT state to be traced through persisted operational evidence.	Identifiers, timestamps, events, windows, snapshots, traces, incidents, logs, and SQLite exports provide verifiable evidence; the traceability evaluation is summarized in Table VIII.

and linking or service mechanisms [3], [4], while multimodal flows recurrently move through capture, preprocessing or normalization, integration or feature extraction, twin update, analysis or decision, and feedback or consumption [2], [4], [7], [8], [13]. The proposed framework translates those recurring stages into implementable software responsibilities for the scope of this research; it does not present the six layers as a universal standard. The layered structure keeps source-specific concerns away from HDT services and external consumers, so a new sensor, file format, protocol, or semantic enrichment process should mainly affect adapter and normalization logic rather than the complete system.

Additionally, the six-layer structure was selected because each layer represents a responsibility with a different reason to change. Capture and ingestion changes when a source, protocol, or device changes. Normalization and synchronization changes when the canonical contract, data quality checks, or temporal alignment rules change. Integration and orchestration changes when the internal flow or routing policy changes. HDT services change when state aggregation or domain rules change. Persistence and traceability changes when the evidence model or audit needs change. Exposure and consumption changes when consumers, interfaces, or access patterns change. Merging these responsibilities would reduce traceability and increase coupling between source-specific code, state logic, storage, and external interfaces. Splitting them into more layers was not necessary for the controlled prototype because the identified drivers could already be mapped to these responsibilities without adding extra coordination mechanisms.

Table IV summarizes the responsibility, data boundary, and quality drivers associated with each layer and with the cross-cutting security concern.

Figure 2 details the expected data flow across these layers, from source-specific capture to the generation, persistence, and protected consumption of the HDT state.

Security is shown as a cross-cutting concern because it protects exposed interfaces and constrains access to persisted human-related data. For that reason, it is not counted as an additional processing layer or modeled as only one isolated component. The prototype applies this concern in a limited way, but the architecture leaves a clear place for stronger authentication, authorization, and audit mechanisms in later versions.

C. Interoperability, Extensibility, and Scalability

Interoperability is supported through source adapters, canonical event contracts, and separation between ingestion logic and HDT services. The canonical event contract includes source identity, modality type, timestamp, payload, quality metadata, and trace identifiers. This contract reduces source-specific coupling and allows the pipeline to process events through common mechanisms after normalization. Future semantic extensions could also use ontology-based mechanisms for more portable definitions of human state and contextual concepts [24].

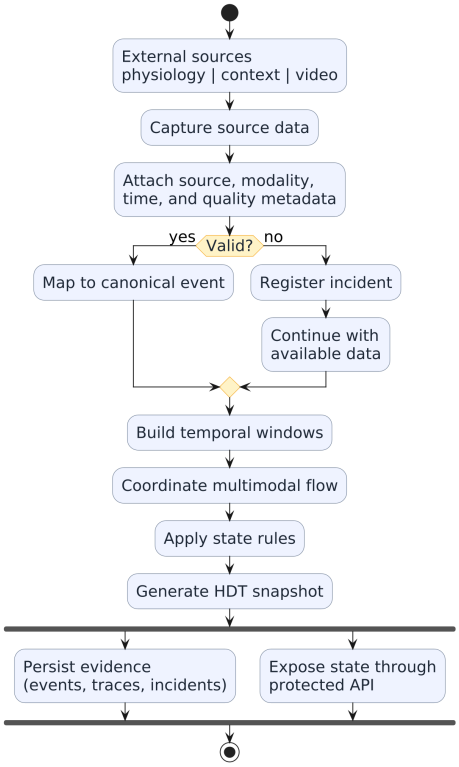

Fig. 2. Multimodal data flow from capture to HDT state consumption.

The prototype evaluates source extensibility by adding modalities through localized adapters and mappings. It does not evaluate throughput, horizontal scaling, or distributed deployment. Scalability is therefore treated as an architectural evolution concern: the separation of capture, integration, persistence, and exposure leaves boundaries that future versions could move toward queue-based ingestion, distributed services, or edge-cloud partitioning. This direction is consistent with the reviewed literature, where edge-cloud structures are useful when latency and proximity to data sources become critical [7].

D. Architectural Decisions

The main decisions are summarized in Table V. The ADRs covered the choice of a hybrid layered and service-oriented architecture, localized source adapters, a canonical event contract, event-based internal exchange, and the separation between captured events, consolidated HDT state, and consumer-facing services. The records kept each decision traceable to a quality driver and to an implementable structure in the controlled prototype.

E. Proof-of-Concept Implementation

The prototype was implemented as a controlled software system, and its source code is available in the repository used for the evaluation.[1] Python 3.12 was selected as the main

[1]https://github.com/agvor/poc-hdt-multimodal

TABLE IV
LAYER RESPONSIBILITIES IN THE PROPOSED FRAMEWORK.

Layer	Main responsibility	Input / output	Supported drivers
Capture and ingestion	Encapsulate source-specific access and convert external observations into internal events with provenance metadata.	Input: physiological, contextual, visual, and operational data. Output: adapter events.	Interoperability, extensibility
Normalization and synchronization	Validate payloads, map them to a common event contract, and align events into temporal windows.	Input: adapter events. Output: canonical events and synchronized windows.	Interoperability, synchronization, fidelity
Integration and orchestration	Coordinate the multimodal flow and route events to HDT services and evidence storage without binding sources to consumers.	Input: canonical events or windows. Output: integrated windows, traces, and incidents.	Scalability, traceability, maintainability
HDT services	Apply state aggregation and rules to build the current and historical representation of the human digital twin.	Input: integrated windows and state rules. Output: state snapshots and derived indicators.	Functional fidelity, modularity
Persistence and traceability	Store state, events, incidents, and logs required to reconstruct pipeline behavior.	Input: snapshots, events, traces, and incidents. Output: queryable history and evidence.	Traceability, observability, security
Exposure and consumption	Provide controlled APIs or interfaces for applications, researchers, or interaction systems that need HDT state.	Input: authorized state queries. Output: state views, indicators, and evidence responses.	Interoperability, security
Security	Constrain access to the exposed interfaces and reduce unnecessary exposure of human-related data.	Input: access requests and identity context. Output: authorized or rejected interactions.	Security, traceability

TABLE V
MAIN ARCHITECTURAL DECISIONS.

Decision	Rationale	Supported drivers
Layered architecture	Separate capture, normalization, integration, HDT services, persistence, and exposure responsibilities.	Modularity, maintainability, traceability
Service-oriented exposure	Expose the HDT state through controlled interfaces instead of direct database or component access.	Interoperability, security, evolvability
Canonical event contract	Normalize heterogeneous payloads into a common internal event structure.	Interoperability, extensibility, traceability
Temporal windowing	Consolidate events using time windows to support functional synchronization.	Synchronization, fidelity
Persistent traces and incidents	Store enough evidence to reconstruct the movement of data and degraded behavior.	Traceability, observability
Localized adapters	Add modalities through source-specific adapters and mapping rules.	Extensibility, maintainability

language. FastAPI and Uvicorn were used for service exposure, Pydantic for data validation, SQLite and SQLAlchemy for persistence, and asynchronous queues for the initial internal event exchange. The implementation also included tests and measurement scripts to support repeatable execution. Selected WESAD and HA4M subsets were later converted offline into continuous scenarios and processed through dataset-specific capture adapters. Dataset knowledge remained in conversion and capture, while the existing normalization, windowing, HDT services, persistence, API, and dashboard flow remained in use.

Figure 3 presents the runtime view of this implementation in two parts: the controlled processing pipeline and the supporting services used to collect evidence and expose the HDT state.

IV. RESULTS AND DISCUSSION

This section reports the controlled and real-dataset evaluation of the proposed framework. The intent was to collect technical evidence about feasibility, not to claim industrial validation.

A. Experimental Setup

The controlled implementation evaluates whether the architecture can execute a complete multimodal flow from data capture to consolidated HDT state consumption. It first integrates physiological and contextual data, and later incorporates a simulated video modality to observe the structural impact of extension. The evaluation was designed as a controlled technical experiment for architectural feasibility, traceability, and localized extensibility under repeatable conditions. This

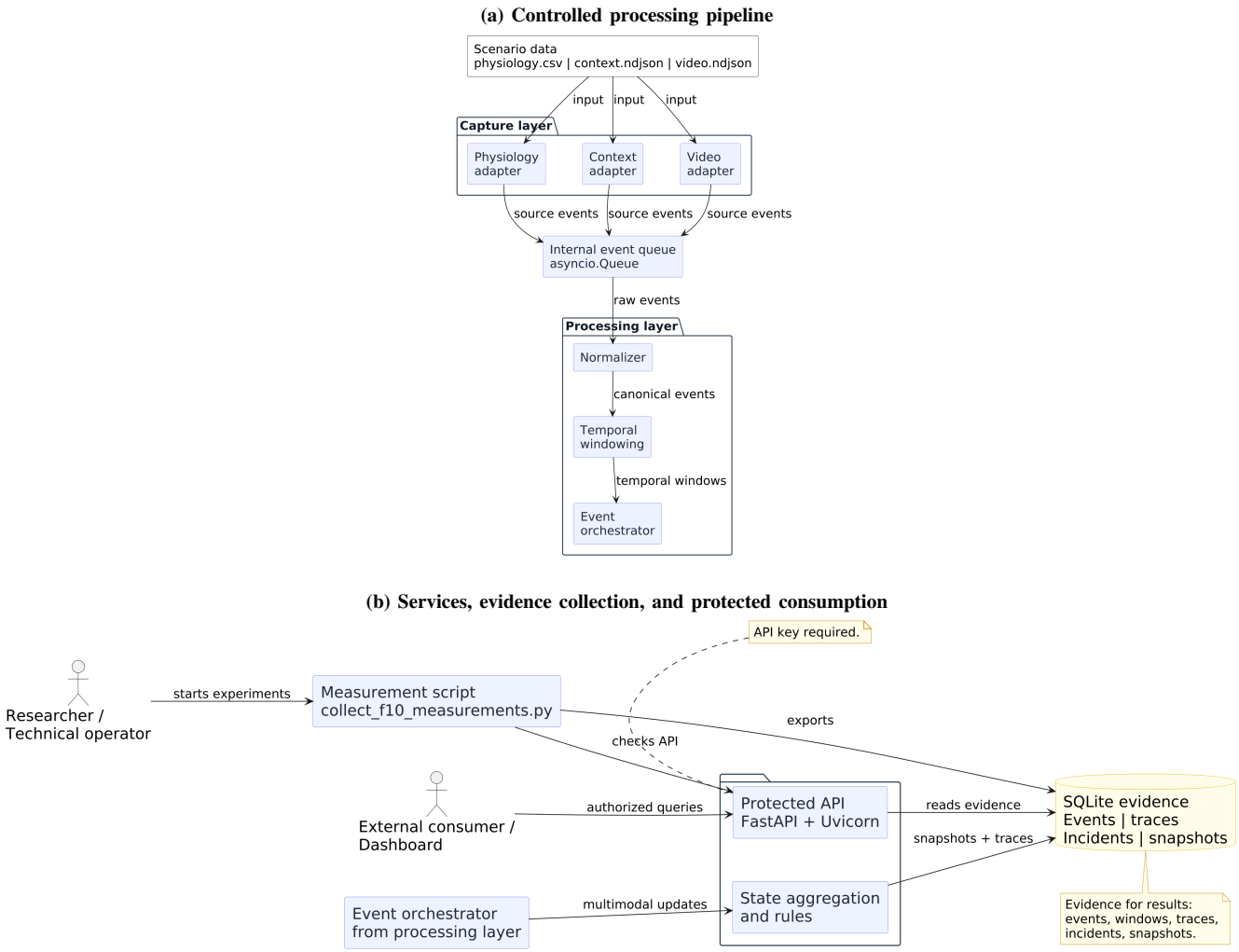

Fig. 3. Runtime view of the prototype implementation.

scenario-based evaluation is consistent with the reviewed HDT literature, where early validation frequently appears as proof-of-concept implementations, controlled experiments, or bounded case studies rather than large-scale industrial deployments [7], [9], [10], [13].

Six controlled experiments were executed. Each experiment used a predefined scenario, active sources, execution interval, and number of iterations. The same architectural pipeline was used in all experiments: source adapters produced events, the system normalized and synchronized those events, the integration layer built windows, HDT services generated state snapshots, and persistence mechanisms stored evidence for later review. The interval was fixed at one second, and each scenario used 40 iterations. This duration generated multiple closed windows while keeping the evaluation repeatable during development and review. Table VI summarizes the configuration.

The experiments were selected to exercise different architectural concerns rather than to represent a statistical sample of real operating conditions. Experiments A–C kept the same two modalities and changed the scenario behavior: A established nominal behavior, B increased physiological and operational load, and C introduced an alarm episode. Experiment D added simulated video to observe extensibility with a third modality. Experiments E and F introduced controlled degradation: E represented partial capture failure, while F introduced invalid physiological data. The one-second interval and 40 iterations allowed repeated event ingestion, window closure, snapshot persistence, and incident registration to be observed without treating the evaluation as a performance benchmark.

The fixed variables were the architecture, implementation baseline, execution interval, measurement procedure, persistence mechanisms, trace collection, protected API checks, and evidence export process. The changing variables were the scenario, the active sources, the presence or absence of video, the existence of a degraded source, the injection of noisy or invalid physiological data, and the presence of an alarm condition. This separation allowed each experiment to exercise a specific architectural concern without changing the complete evaluation procedure.

TABLE VI
CONTROLLED EXPERIMENTS USED IN THE PROTOTYPE EVALUATION.

Exp.	Scenario	Active sources	Interval	Iterations	Purpose
A	normal-day	physiology + context	1 s	40	Establish nominal baseline behavior.
B	high-load	physiology + context	1 s	40	Observe increased physiological and operational load.
C	alarm-episode	physiology + context	1 s	40	Verify state response under an alert condition.
D	normal-day-video	physiology + context + video	1 s	40	Evaluate extension by adding a third modality.
E	degraded-source	physiology + context	1 s	40	Observe handling of partial capture failures.
F	noisy-physiology	physiology + context	1 s	40	Observe rejection of invalid data and controlled degradation.

For each experiment, the complete pipeline was executed and evidence was collected from the runtime state, iteration records, closed windows, persisted events, generated snapshots, traces, incidents, logs, protected API responses, and SQLite database. The measurement procedure was implemented by the repository script `scripts/collect_f10_measurements.py`. The collected evidence was used to check whether the system completed the multimodal flow, whether degraded cases were registered as incidents, whether snapshots were persisted, and whether the protected API exposed the consolidated HDT state.

B. Initial Simulated Video Modality

The initial video modality was simulated and semantically enriched. It was not based on an external video dataset, and at this stage the prototype did not process real frames or implement computer vision models. Instead, the scenario represented video observations as structured events in `video.ndjson`. These events provided frame-level information and semantic indicators that could be consumed by the HDT pipeline in the same way as other normalized modality events.

This bounded design kept the evaluation focused on architectural extension. The objective was to evaluate whether the architecture could incorporate a new modality with localized changes, not to evaluate visual perception accuracy, frame processing performance, or model quality. In the video scenario, the HDT state included indicators such as `latest_frame_uri`, `latest_frame_index`, and `latest_attention_state`. These indicators verified that video-derived information entered the pipeline, reached the consolidated state, and remained available through the existing consumption mechanisms.

C. Operational Results

The controlled implementation sustained a complete multimodal flow from source capture to HDT state consumption. In the base configuration, the system integrated physiological and contextual sources into temporal windows and generated persisted snapshots. The video experiment added a third modality and continued to produce windows, snapshots, and API-visible state without changing the protected API.

Table VII presents the consolidated operational results. Experiments A and B generated 160 persisted events and eight snapshots without incidents. Experiment C also generated

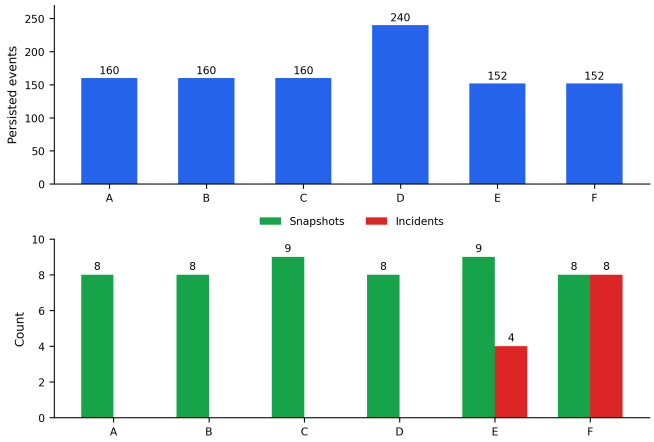

Fig. 4. Operational results by controlled experiment.

160 persisted events and produced nine snapshots. Experiment D generated 240 events because the video scenario added a third active modality while keeping the same number of iterations. Experiments E and F generated 152 persisted events. This lower count is consistent with the controlled degradation introduced in those scenarios: part of the expected input was unavailable, rejected, or treated as invalid, but the pipeline continued operating and recorded incidents for later review.

Figure 4 summarizes the operational trend across the controlled scenarios. Experiments A–C preserve the same two-modality event volume across nominal, high-load, and alarm conditions. Experiment D shows that adding video increases event volume without increasing incidents. Experiments E and F show that degraded behavior is not hidden: incidents are recorded while snapshots continue to be generated.

The numeric pattern also helps interpret the quality attributes. The 160-event result in experiments A–C shows that the same two-modality pipeline processed nominal, high-load, and alarm scenarios. The number of closed windows varied from eight to nine across the selected runs, while each closed window produced a matching snapshot. Experiment D increased the persisted event count to 240 and still produced eight snapshots with no incidents; this supports the narrower claim that the third modality entered the pipeline without disrupting the existing flow. Experiments E and F produced 152 persisted events, eight fewer than the two-modality baseline, but they still generated

TABLE VII
OPERATIONAL RESULTS OF THE CONTROLLED PROTOTYPE.

Exp.	Windows	Events	Snapshots	Incidents	Modalities	Observation
A	8	160	8	0	context, physiology	Nominal operation without incidents.
B	8	160	8	0	context, physiology	Sustained increase in physiological load.
C	9	160	9	0	context, physiology	Alarm activation and warning context.
D	8	240	8	0	context, physiology, video	Correct integration of three modalities.
E	9	152	9	4	context, physiology	Partial capture failure with operational continuity.
F	8	152	8	8	context, physiology	Invalid data rejected with controlled degradation.

snapshots and registered four and eight incidents, respectively. These incidents show that degraded behavior was observable and traceable, not that failure itself was successful behavior.

D. Results by Quality Attribute

The results were also interpreted against the quality drivers defined in the design. Table VIII connects each attribute to observable evidence from Table VII, the protected API checks, and the persisted SQLite evidence. The final column states the scope of that evidence rather than declaring an attribute fully satisfied.

Experiment D provides the main interoperability and extensibility evidence. Simulated video increased the persisted event count from 160 to 240 while the system continued to produce closed windows and snapshots through the same downstream flow. The change affected expected implementation points but preserved integration, persistence, and protected API routes. This is evidence of localized extension, not of cost-free change or throughput scalability.

Synchronization evidence is limited to functional temporal windowing. Across the selected 40-iteration runs, each closed window produced a matching snapshot, but the evaluation did not measure strict real-time latency. Traceability is more directly observable: experiments E and F persisted incidents and state evidence even when capture was incomplete or physiological data were rejected, making degraded behavior reviewable after execution.

Security and functional fidelity were evaluated narrowly. The API-key check distinguished unauthorized from authorized requests, but it is not a complete security model. Scenario changes also appeared in the consolidated state as higher load, alarm behavior, visual indicators, capture incidents, and rejected invalid data. This establishes scenario-to-state correspondence within the controlled implementation, not physiological or clinical accuracy.

E. Impact of Adding the Simulated Video Modality

The addition of simulated video was the most relevant extension test. It allowed the evaluation to observe whether the architecture could absorb a new modality without redesigning the system. In this evaluation, impact means the structural effect of the change on the implementation: which files were added or modified, which architectural layers were touched, and whether core contracts, protected routes, or the general architecture had to change.

The change introduced a controlled video adapter and new scenario data for the video-enabled experiment. It also required localized modifications in expected areas: capture, normalization, HDT state aggregation, API wiring, tests, documentation, and scenario data. The claim is narrow: adding a modality still requires code, but it should not force changes to the established event contracts, protected API routes, persistence strategy, or overall architectural structure.

Table IX summarizes the concrete implementation impact, and Figure 5 presents the same impact visually. The result supports the architectural decision to isolate source-specific concerns in capture and normalization while keeping integration, persistence, and external consumption stable. The count of modified files should not be interpreted as a universal measure of implementation cost or developer productivity. In this prototype, the count is only a way to observe whether the change was concentrated in expected parts of the system.

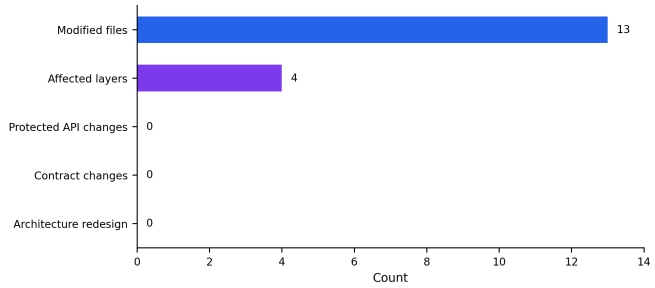

Fig. 5. Localized implementation impact of adding the simulated video modality.

The same result also defines its limits. The video modality was simulated and semantically enriched, so the experiment does not evaluate computer vision accuracy, frame processing latency, storage pressure, or privacy mechanisms for real video streams. A full real-time video pipeline could require additional design decisions for streaming, storage retention, edge processing, privacy controls, and higher-volume evidence management. Experiment D supports structural extensibility: a third modality was integrated while contracts and protected consumption remained stable.

F. Evaluation with Real Datasets

A second evaluation stage used selected subsets of WESAD and HA4M. Both datasets were converted offline into con-

TABLE VIII
EVALUATION BY QUALITY ATTRIBUTE.

Attribute	Observed evidence	Evidence scope
Interoperability	The experiments generated valid windows and snapshots with context and physiology; experiment D also included video and produced 240 persisted events.	Three modalities in a controlled flow
Latency / synchronization	Forty iterations produced between eight and nine closed windows and matching snapshots depending on the scenario, without observable fragmentation.	Functional windowing; no latency benchmark
Extensibility	Experiment D operated with three modalities, generated eight snapshots and 240 persisted events, and the change remained localized.	One simulated modality extension
Security	Protected services returned unauthorized access without an API key and successful access with a valid key.	Basic API-key check only
Maintainability / modularity	Video integration affected capture, normalization, HDT services, API wiring, tests, documentation, and scenario data, but did not require changes to integration, persistence, or protected API routes.	Change-localization evidence
Traceability / observability	Each experiment preserved state, iterations, windows, traces, incidents, logs, and the complete SQLite database.	Persisted execution evidence
Functional fidelity	Scenario variations were reflected in the consolidated state: higher load in B, alarm behavior in C, visual indicators in D, capture incidents in E, and rejection of invalid data in F.	Scenario-to-state correspondence

TABLE IX
IMPACT OF INCORPORATING THE SIMULATED VIDEO MODALITY.

Aspect	Observed result
New modality	Simulated and semantically enriched video.
New files	`src/app/capture/controlled_video_adapter.py` and scenario files under `data/scenarios/normal-day-video/`.
Modified files	13 files, including documentation and tests.
Affected areas	capture, normalize, hdt_services, API wiring, tests, documentation, and scenario data.
Protected API routes modified	No.
Contracts modified	No.
Architectural redesign required	No.
Interpretation	The new modality was integrated with localized impact, consistent with the separation of responsibilities defined by the architecture.

tinuous scenarios and processed with the same one-second interval and 40-iteration procedure used in the controlled experiments. They are presented separately because they exercise different extension paths: WESAD tests the adaptation of real physiological signals, while HA4M tests real-frame provenance and simple visual processing.

1) WESAD: Real Physiological Input: WESAD is a multimodal dataset for wearable stress and affect detection that includes synchronized physiological signals [25]. The prototype used subject S2 and preserved raw signal slices in a sidecar. `WesadPhysiologyAdapter` projected BVP or ECG, temperature, EDA, respiration, and acceleration into the canonical physiological payload. The derivations were intentionally simple and were used for architectural validation rather than clinical interpretation.

Table X separates the two-modality WESAD scenario from its optional three-modality profile. The base and WESAD scenarios both produced 160 persisted events, eight windows, eight snapshots, and no incidents. WESAD increased the HDT state from 13 to 16 indicators by adding EDA, respiration rate, and motion magnitude. Their final measured values were 1.3068 μS, 29 bpm, and 63.286, respectively.

The protocol profile added 80 events, corresponding to raw and normalized events from the third source, without adding snapshots or incidents. It was used to verify three-modality execution only; its visual metadata came from WESAD protocol labels and did not represent real frames.

The architectural impact of WESAD was low, while its implementation impact was medium and localized. The work added an offline converter, external scenario configuration, `WesadPhysiologyAdapter`, runtime source selection, optional physiological fields in normalization, and three HDT indicators. It also required measurement scripts and focused tests. No container, channel, endpoint, persistence table, C4 element, or cross-cutting core responsibility was added. Raw WESAD signals remained outside the normalized state, and the protected API and dashboard reused their existing interfaces.

2) HA4M: Two-Stage Real-Frame Evaluation: HA4M contains multimodal recordings of a manufacturing assembly task [26]. The prototype used RGB frames and action labels from sequence `IDU004V006`. HA4M does not provide the physiological stream used by the prototype, so the HA4M scenarios retained controlled auxiliary physiology. The visual integration was evaluated in two stages, HA4M-A and HA4M-B, to distinguish frame provenance from frame-derived processing.

HA4M-A was the first stage. An offline converter sampled and copied real RGB frames, generated context from `Labels.txt`, and stored traceable `frame_uri` and `frame_index` values in the existing visual event format. The runtime reused `ControlledVideoAdapter`; the canonical contract, normalization, HDT rules, persistence schema, API, and dashboard did not change. Its implementation changes were

TABLE X
WESAD EXECUTION RESULTS.

Case	Data contribution	Modalities	Windows	Events	Snapshots	Incidents	HDT evidence
Controlled base	Simulated physiology and context.	context, physiology	8	160	8	0	13 indicators.
WESAD	Physiological signals from subject S2.	context, physiology	8	160	8	0	16 indicators; EDA, respiration, and motion added.
WESAD + protocol profile	WESAD physiology with protocol-derived visual metadata.	context, video, physiology	8	240	8	0	24 indicators; no real video frames.

limited to the converter, measurement script, converter test, and supporting documentation. This stage had low architectural impact.

HA4M-B extended HA4M-A with simple frame processing in capture. `Ha4mVideoAdapter` read the PNG referenced by `frame_uri` and derived luminance mean, luminance standard deviation, and a SHA-256 checksum. This required optional visual contract fields, runtime adapter selection, normalization support, three HDT indicators, and focused tests. It did not add a service, channel, endpoint, container, database migration, or visual-processing responsibility to the HDT core. Its architectural impact was therefore medium but localized.

Table XI contrasts the two stages. Both processed 40 iterations and persisted 240 events with no incidents. HA4M-A produced nine windows and snapshots because the runtime materialized one partial window during shutdown. This is reported as execution-specific closure behavior, not as an improvement or regression. HA4M-B matched the three-modality baseline with eight windows and snapshots and increased the final state from 21 to 24 indicators. The recorded luminance mean and standard deviation were 121.578 and 73.722, respectively, and the checksum identified the final referenced frame.[2]

G. Discussion

The results are easier to interpret in stages. The controlled experiments first established that the prototype could execute the full HDT flow under repeatable conditions. Experiments A–C exercised nominal, high-load, and alarm behavior with two modalities. Experiment D added simulated video and verified that a third modality could increase event volume without changing the downstream consumption path. Experiments E and F then introduced degraded capture and invalid physiological data to verify that the system continued operating while preserving incidents and reviewable records. This controlled baseline defines the expected architectural behavior before introducing real datasets.

Figure 6 compares event volume across the controlled baselines and the real-dataset scenarios. The result mainly reflects the number of active sources: two modalities produced 160 persisted events, while three modalities produced 240. Figure 7 separates the state view: indicator count reflects state

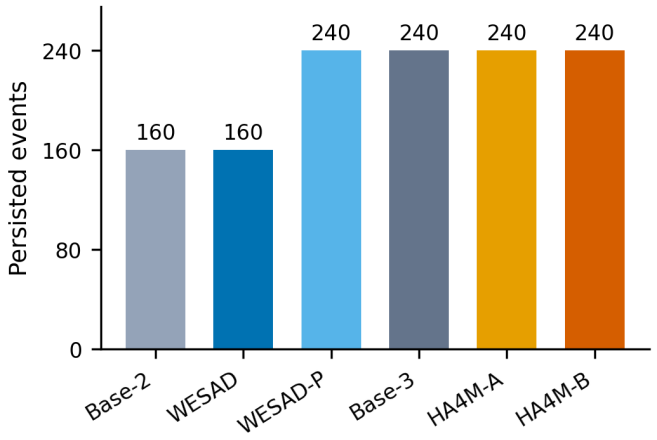

Fig. 6. Persisted events across controlled baselines and real-dataset scenarios. Event volume follows the number of active sources.

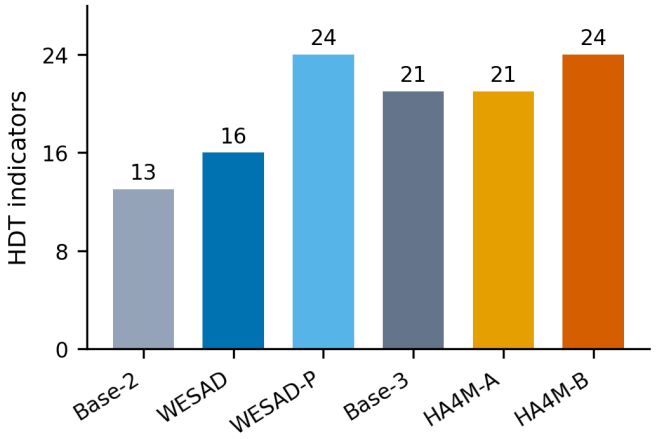

Fig. 7. HDT indicator counts across controlled baselines and real-dataset scenarios. Indicator count describes state expressiveness, not semantic correctness.

expressiveness, not indicator quality or accuracy. WESAD added three physiological indicators, and HA4M-B added three frame-derived indicators. Tables X and XI report window and snapshot closure; both remained at eight except for the documented HA4M-A closure artifact. The real-dataset runs recorded zero incidents.

Figure 8 compares WESAD with the two-source baseline,

[2]HA4M-A evidence was collected at repository state `102d19a`; HA4M-B is reproducible from `2d1336f`.

TABLE XI
HA4M-A AND HA4M-B EXECUTION RESULTS.

Case	Visual capability	Windows	Events	Snapshots	Incidents	Indicators	Interpretation
Three-modality base	Simulated visual events.	8	240	8	0	21	Operational comparison point.
HA4M-A	Real frame references and manufacturing action labels.	9	240	9	0	21	Provenance and traceability; one partial closing window.
HA4M-B	HA4M-A plus three features derived from referenced frames.	8	240	8	0	24	Minimal real-frame processing in capture.

HA4M-A with the existing three-source pipeline, and HA4M-B with HA4M-A. A changed cell identifies an implementation area modified at that stage; a reused cell identifies a preserved boundary. Figure 9 counts the functional files listed in the prototype impact analyses under one consistent rule: runtime source, conversion or measurement scripts, and tests are included, while documentation-only changes are excluded. WESAD affected 12 functional files, HA4M-A affected three, and HA4M-B added nine relative to HA4M-A. These counts describe change localization, not implementation effort or productivity.

The controlled and real-dataset results lead to three architectural observations. Source-specific formats were absorbed through offline conversion and capture adapters while the downstream runtime remained stable. Raw WESAD signals and RGB frames remained outside the consolidated HDT state and event payloads. Persisted events, snapshots, traces, and source references preserved a reviewable record of how each state was produced. These observations support interoperability, traceability, and extensibility within the evaluated scope; they do not demonstrate advanced multimodal fusion or production readiness.

The three real-data integrations show different levels of reuse. WESAD required a dedicated physiology adapter, optional physiological fields in normalization, and new HDT indicators because it introduced signal types absent from the controlled baseline. HA4M-A mostly reused the existing visual path and added provenance through frame references. HA4M-B crossed additional extension points in capture, the visual contract, normalization, and HDT indicators to derive frame features. In all three cases, runtime topology, persistence schema, and API endpoints remained unchanged. The affected-file counts in Figure 9 show where change was localized, not implementation effort.

The degraded scenarios provide a separate check on operational continuity. Experiments E and F did not eliminate all errors; that was not the goal. Instead, they showed that failures and invalid data can be recorded while the pipeline continues operating with available sources. This behavior matters for HDT systems because human-machine interaction environments are likely to involve noisy, partial, or temporarily unavailable data.

Taken together, the results support the feasibility of adapting new source formats and preserving traceable state updates in a controlled environment. They do not establish throughput scalability, strict real-time behavior, clinical validity, action-recognition accuracy, or advanced multimodal fusion. WESAD and HA4M also came from separate selected subsets, so the evidence concerns architectural adaptation rather than a synchronized representation of one participant.

The repository contains the measurement script, scenario definitions, and export procedure used in this evaluation.

H. Threats to Validity

The evaluation is limited by the scope of the prototype. Its purpose was to collect initial technical evidence for the proposed architecture, not to demonstrate production readiness or industrial effectiveness.

Internal validity is limited by the use of predefined scenarios and a controlled implementation. The observed behavior may depend on the scenario data, adapter and rule implementations, measurement scripts, and execution timing. HA4M-A and HA4M-B were also measured at different repository states, and the ninth HA4M-A snapshot resulted from partial-window materialization during shutdown. The two stages are therefore reported separately, and that difference is not interpreted as a performance result. To reduce this threat, the evaluation kept the interval, measurement procedure, persistence mechanism, trace collection, API checks, and evidence export process fixed. Results were backed by events, snapshots, windows, traces, incidents, logs, and SQLite exports rather than by manual observation only.

External validity remains limited because the prototype was not deployed in a real industrial human-machine interaction scenario. The evaluation incorporated real WESAD physiological signals and real HA4M RGB frames, but only selected subsets were used: one WESAD subject and one HA4M sequence. The real physiological and visual sources also came from separate datasets rather than the same participant or synchronized field session. In addition, the WESAD visual profile was protocol-derived metadata, and HA4M physiology was controlled auxiliary data. For these reasons, the results should not be generalized directly to production HDT systems.

Construct validity is limited because the measured evidence is a proxy for architectural qualities. Event count mainly reflects the number of active sources, while indicator count reflects state expressiveness rather than semantic correctness or model accuracy. File counts show where changes occurred, not implementation effort, complexity, or productivity. Windows,

	WESAD	HA4M-A	HA4M-B
Offline conversion	**Change**	**Change**	Reuse
Runtime source selection	**Change**	Reuse	**Change**
Capture adapter	**Change**	Reuse	**Change**
Canonical contract	**Change**	Reuse	**Change**
Normalization	**Change**	Reuse	**Change**
HDT indicators	**Change**	Reuse	**Change**
Runtime topology / services	Reuse	Reuse	Reuse
Persistence schema	Reuse	Reuse	Reuse
API endpoints	Reuse	Reuse	Reuse

Fig. 8. Architectural impact of real-dataset support. Changed cells identify localized implementation changes; reused cells identify preserved architectural boundaries.

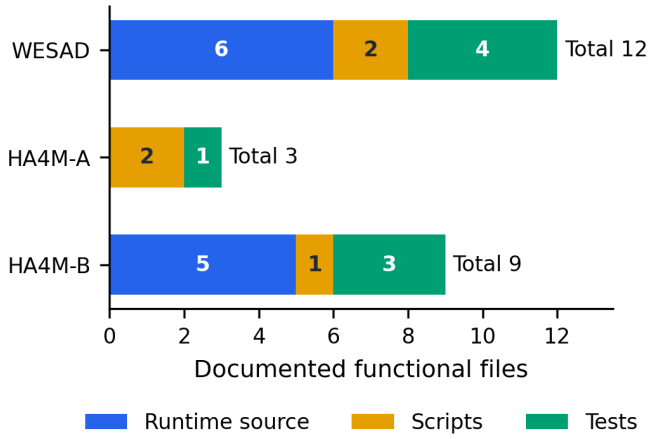

Fig. 9. Documented functional files affected by real-dataset support. Documentation-only changes are excluded, and counts are not measures of effort or productivity.

snapshots, incidents, affected areas, and API responses help evaluate synchronization, extensibility, traceability, and basic security, but they do not fully measure those qualities. WESAD derivations are not clinical measurements, and HA4M-B computes descriptive frame features rather than action recognition. The evaluation also checks closed windows rather than strict real-time latency, basic API protection rather than a complete security model, and localized integration rather than advanced multimodal fusion.

Conclusion validity is limited by the bounded number of scenarios, selected dataset subsets, and absence of a baseline architecture for comparison. The reported values are descriptive results from 40-iteration executions, not statistical estimates. They show that the prototype completed the intended flow, incorporated real physiological and visual sources with localized impact, and preserved evidence under degraded scenarios. They do not prove superiority over alternative styles or equivalent behavior at industrial scale. The conclusions are limited to technical feasibility and architectural plausibility for the evaluated scope.

V. Conclusions and Future Work

This paper presented a multimodal software architecture for Human Digital Twins in human-machine interaction scenarios. The architecture organizes the HDT data flow into layers for ingestion, normalization and synchronization, integration, HDT services, persistence, traceability, and consumption. The proposal was designed with explicit attention to interoperability, extensibility, synchronization, traceability, maintainability, and future scalability.

The implementation evidence indicates that the architecture can integrate heterogeneous sources, preserve traceability, and incorporate additional data sources with localized impact. The six controlled experiments generated windows and snapshots, registered degraded cases, and preserved evidence of pipeline behavior. WESAD added real physiological signals and three indicators with low architectural impact and medium localized implementation change. HA4M-A established traceable access to real RGB frames by reusing the visual contract, while HA4M-B added three frame-derived indicators through localized changes in capture, normalization, and HDT state aggregation. Across these integrations, runtime topology, persistence schema, and API endpoints remained unchanged. Within this scope, the study met its objective of designing and technically evaluating a multimodal HDT architecture. The results support technical

feasibility, but not clinical validity, visual-model accuracy, or synchronized field operation.

Future work should address six directions. First, the architecture should be evaluated with synchronized real multimodal data from the same participants in deployed human-machine interaction scenarios. Second, the system should be tested with larger data volumes and stricter latency requirements. Third, privacy, consent, data minimization, and security controls should be strengthened because HDTs may process sensitive human-related data. Fourth, the architecture should be extended toward edge-cloud deployment to evaluate scalability and near-real-time behavior. Fifth, future iterations should incorporate multimodal fusion mechanisms, since the current version focuses on integration, synchronization, traceability, and state consolidation rather than advanced fusion. Finally, validation should include longer experiments and comparison with alternative architectural styles.

REFERENCES

[1] I. Onaji, D. Tiwari, P. Soulatiantork, B. Song, and A. Tiwari, "Digital twin in manufacturing: conceptual framework and case studies," *International Journal of Computer Integrated Manufacturing*, vol. 35, no. 8, pp. 831–858, 2022.

[2] D. Zhong, Z. Xia, Y. Zhu, and J. Duan, "Overview of predictive maintenance based on digital twin technology," *Heliyon*, vol. 9, no. 4, p. e14534, 2023.

[3] B. Wang, H. Zhou, X. Li, G. Yang, P. Zheng, C. Song *et al.*, "Human digital twin in the context of industry 5.0," *Robotics and Computer-Integrated Manufacturing*, vol. 85, p. 102626, 2024.

[4] Q. He, L. Li, D. Li, T. Peng, X. Zhang, Y. Cai *et al.*, "From digital human modeling to human digital twin: Framework and perspectives in human factors," *Chinese Journal of Mechanical Engineering*, vol. 37, p. 9, 2024.

[5] U. Asad, M. Khan, A. Khalid, and W. Lughmani, "Human-centric digital twins in industry: A comprehensive review of enabling technologies and implementation strategies," *Sensors*, vol. 23, no. 8, p. 3938, 2023.

[6] M. Zafar, E. Langås, and F. Sanfilippo, "Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review," *Robotics and Computer-Integrated Manufacturing*, vol. 89, p. 102769, 2024.

[7] Y. Shi, W. Shen, L. Wang, F. Longo, L. Nicoletti, and A. Padovano, "A cognitive digital twins framework for human-robot collaboration," in *Procedia Computer Science*, vol. 200, 2022, pp. 1867–1874.

[8] Z. Chen, K. Surendraarcharyagie, K. Granland, C. Chen, X. Xu, Y. Xiong *et al.*, "Service oriented digital twin for additive manufacturing process," *Journal of Manufacturing Systems*, vol. 74, pp. 762–776, 2024.

[9] S. Chand, H. Zheng, and Y. Lu, "A vision-enabled fatigue-sensitive human digital twin towards human-centric human-robot collaboration," *Journal of Manufacturing Systems*, vol. 77, pp. 432–445, 2024.

[10] R. Zhong, B. Hu, Y. Feng, H. Zheng, Z. Hong, S. Lou, and J. Tan, "Construction of human digital twin model based on multimodal data and its application in locomotion mode identification," *Chinese Journal of Mechanical Engineering*, vol. 36, p. 126, 2023.

[11] J. Biolchini, P. Mian, A. Natali, and G. Travassos, "Systematic review in software engineering," COPPE/UFRJ, Systems Engineering and Computer Science Department, Rio de Janeiro, Brazil, Tech. Rep., 2005.

[12] K. Kokkonen, L. Hannola, T. Rantala, J. Ukko, M. Saunila, and T. Rantala, "Preconditions and benefits of digital twin-based business ecosystems in manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 36, no. 5, pp. 789–806, 2023.

[13] T. Wang, Z. Liu, L. Wang, M. Li, and X. Wang, "A design framework for high-fidelity human-centric digital twin of collaborative work cell in industry 5.0," *Journal of Manufacturing Systems*, vol. 80, pp. 140–156, 2025.

[14] V. Villani, M. Picone, M. Mamei, and L. Sabattini, "A digital twin driven human-centric ecosystem for industry 5.0," *IEEE Transactions on Automation Science and Engineering*, vol. 22, 2025.

[15] J. de Marchi and E. Baalbergen, "Towards a human-centric digital twin architecture for industry 5.0: Aiding skilled operators with composites production automation," in *Journal of Physics: Conference Series*, vol. 2526, 2023, p. 012047.

[16] Y. You, B. Cai, D. Pham, Y. Liu, and Z. Ji, "A human digital twin approach for fatigue-aware task planning in human-robot collaborative assembly," *Computers & Industrial Engineering*, p. 110774, 2025.

[17] B. Balaji, M. Shahab, B. Srinivasan, and R. Srinivasan, "ACT-R based human digital twin to enhance operators' performance in process industries," *Frontiers in Human Neuroscience*, vol. 17, 2023.

[18] Y. Sun, Q. Zhang, J. Bao, Y. Lu, and S. Liu, "Empowering digital twins with large language models for global temporal feature learning," *Journal of Manufacturing Systems*, vol. 74, pp. 83–99, 2024.

[19] A. Ramasubramanian, R. Mathew, M. Kelly, V. Hargaden, and N. Papakostas, "Digital twin for human–robot collaboration in manufacturing: Review and outlook," *Applied Sciences*, vol. 12, no. 10, p. 4811, 2022.

[20] Y. Liu, J. Feng, J. Lu, and S. Zhou, "A review of digital twin capabilities, technologies, and applications based on the maturity model," *Advanced Engineering Informatics*, vol. 62, p. 102592, 2024.

[21] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 4th ed. Boston, MA, USA: Addison-Wesley Professional, 2021.

[22] L. Naranjo-Zeledón, "Investigación en informática: el enfoque alternativo," *Technology Inside*, vol. 5, no. 5, pp. 1–15, 2020. [Online]. Available: https://cpic-sistemas.or.cr/revista/index.php/technology-inside/article/view/35

[23] A. Hevner, "A three cycle view of design science research," *Scandinavian Journal of Information Systems*, vol. 19, no. 2, pp. 87–92, 2007.

[24] T. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[25] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. V. Laerhoven, "Introducing WESAD, a multimodal dataset for wearable stress and affect detection," in *Proc. 20th ACM Int. Conf. Multimodal Interaction*, 2018, pp. 400–408.

[26] G. Cicirelli, R. Marani, L. Romeo, M. G. Domínguez, J. Heras, A. G. Perri, and T. D'Orazio, "The HA4M dataset: Multi-modal monitoring of an assembly task for human action recognition in manufacturing," *Scientific Data*, vol. 9, no. 1, p. 745, 2022.