International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V7I1P124
Eureka Vision Publication | Volume 7, Issue 1, 157 -166, 2026

Original Article

Clinical Event Architecture: Perspective Conflict Patterns in Healthcare Information Systems

Deepanjan Mukherjee
Independent Researcher, Austin, TX USA.

Received On: 30/12/2025 ***Revised On: 29/01/2026*** ***Accepted On: 09/02/2026*** ***Published On: 16/02/2026***

Abstract – Healthcare information systems force diverse stakeholders into single data models, creating fundamental conflicts when the same clinical events require different representations for billing, safety monitoring, regulatory compliance, and clinical workflows. Current approaches rely on complex transformation layers, data duplication, or forcing stakeholders into misfit models, resulting in data quality issues, integration complexity, and compromised decision support. This paper presents a systematic catalog of perspective conflict patterns observed in clinical event systems, documenting four fundamental conflict types: Safety versus Workflow Granularity, Billing versus Clinical Timing, Identity Context Dependence, and Compliance Audit versus Performance Optimization. For each pattern, the paper provides clinical context, competing stakeholder requirements, architectural solutions grounded in event-driven design principles, and trade-off analysis. The paper proposes a reference architecture employing event sourcing with perspective-specific projections that enables multiple valid representations of the same clinical events without transformation complexity. This pattern catalog provides healthcare architects and informaticists with a systematic framework for diagnosing and resolving multi-perspective conflicts in electronic health record systems and clinical decision support platforms.

Keywords – Healthcare Information Systems, Electronic Health Records (Ehrs), Event-Driven Architecture, Clinical Decision Support, Domain-Driven Design, Interoperability, Pattern Language, Event Sourcing, Command Query Responsibility Segregation (CQRS).

1. Introduction

In modern healthcare delivery, there are multiple entities who have drastically different perspectives of the same clinical events. A single administration of medication is a billable service to revenue cycle staff; a safety-critical event requiring detailed documentation for clinical pharmacists; a workflow completion task for nursing staff; and a documentation for auditable compliance for quality officers. Electronic health record (EHR) systems must serve all these perspectives simultaneously, but existing architectures usually prioritize one stakeholder and often lead to ill-fitting data models for others. The negative implications of this architectural dissonance are well established. Clinicians are finding themselves spending more time navigating interfaces designed primarily for billing documentation than for clinical workflows, contributing to physician burnout and reduced time with patients [1]. By failing to differentiate between clinically useful and administratively necessary warnings, clinical decision support systems raise alerts so high that override rates for particular kinds of alerts exceeding 90% for certain alert types often take place [2]. In situations where stakeholders choose to enter information, but not all fields match their own mental model, data quality suffers and workarounds are devised, copy-pasted, and missing important information occurs [3].

The present methods to resolve such conflicts are divided into three different categories. Each category has considerable shortcomings. Transformation layer approaches attempt to transfer information from one point of view to another during a query, thus bringing latency, semantic loss and maintenance difficulty along the way, as the number of different points of view increases. Data duplication strategies have to maintain individual databases for separate members, leading to synchronization difficulties and more chances of inconsistency. Forced unification methods adopt a stakeholder's system as standard and demand adjustment by every other stakeholder which will inevitably result in friction and an attempt at workarounds.

Recent advances in event-driven architecture and domain-driven design provide alternative approaches. Event sourcing architectures maintain immutable records of clinical events and allow for multiple perspective-specific projections, which are tailored to the specific needs of any given stakeholder group [4]. Domain-driven design principles accept that various bounded contexts require different models of the same domain entities with justification [5]. Healthcare literature, however, does little to systematically document the perspective of conflicts in clinical systems, and the architectural pattern for that conflict resolution. There are three contributions to this paper. First, a taxonomy is proposed that divides perspective conflicts into four basic classes, depending on what mismatches between stakeholder requirements exist. Second, four specific patterns are presented that represent the most frequently observed conflicts in clinical event systems—clinical context, competing forces, solution approaches, and trade-offs. Third, a reference architecture is proposed proving how event sourcing with perspective-specific materialized views can

resolve these conflicts without the complexity of transformation layers or risks of data duplication.

Our pattern catalog relies on established software architecture patterns, published clinical workflow studies, and healthcare information technology case studies. Each is based on well-documented clinical cases and accompanied by relevant citations to peer review documents. The reference architecture includes proven methods taken from event driven systems, in a healthcare context while meeting the regulatory requirements, including HIPAA compliance and audit trail maintenance. The rest of this paper will proceed as follows. Section II is a review of relevant background information about clinical event systems. This is based upon the current state data model approaches, architectural trends in software engineering, and architectural concepts taken from the literature. A taxonomy of perspective conflict types is presented in Section III. Section IV details four detailed patterns that are supported by clinical examples and architectural solutions. Section V provides the reference architecture to apply these patterns. Section VI introduces the implications, boundaries, and scope, and Section VII concludes by drawing prospects for future work.

2. Background

2.1. Clinical Event Systems and Stakeholder Standpoints

Thousands of discrete clinical events are generated daily in healthcare delivery, from administering medications and laboratory tests to vital sign measurements and clinical assessments. Every event is relevant to multiple stakeholder groups with distinct information needs and decision-making requirements.

Patient state information is needed to help clinical staff make informed treatment decisions, and this is something care should have in real time. Nurses need to have task lists that outline pending medication administrations and required documentation. Physicians need a longitudinal patient history sorted by clinical problems or body system [6]. In terms of medication safety, pharmacists pay close attention to medication interaction checking, confirmation of allergies, and medication avoidance to guarantee that drug safety is provided. The revenue cycle functions view the same events as billable services. Billing specialists need to charge capture for all services provided, mapped to relevant CPT and ICD-10 codes for reimbursement [7]. Case managers also need documents to authorize treatment based on medical necessity assessments for payers [8]. Aggregated utilization data are useful for financial analysts to budget and forecast.

More requirements are introduced for quality and compliance functions. Quality improvement staff require performance criteria computed from a defined clinical quality measure [9]. In risk management, full audit trails are needed for malpractice defense and root cause analysis [10]. Data formats and regulations require that it is mandated by CMS, Joint Commission, and state-specific authorities [11].

A perspective based on research and population health uses data from individual events for cohort analysis and outcomes of research. Clinical investigators require de-identified patient information consistent with privacy laws and allowing for statistical validity [12]. Focused population health management necessitates risk stratification and care gap identification across patient panels. These different requirements present fundamental tensions. Real-time clinical workflows provide speed and simplicity, but compliance requirements have demanded documentation. While billing perspectives are service-level granularity, clinical quality measures operate with episode or population scales. Longitudinal data aggregation is required for research applications whereas operational dashboards are the current-state snapshots.

2.2. Current EHR Data Modeling approaches

Electronic health record systems practice various models and employ multiple data modeling techniques that accommodate varying levels of optimization for specific stakeholders and create new challenges for others. It offers a model for data organization, structured around the current state of patients; transactional database models, such as transactional database models, optimize for current patient state and clinical workflow, for which the workflows dictate timely retrieval of up-to-date information. These models usually use normalized relational plans, with tables for patients, encounters, orders, results, and documentation. This architecture is best for point-in-time queries in data handling and is efficient, but it can make historical analysis and audit trail reconstruction more challenging.

Data sources from diverse and disparate healthcare systems are linked to clinical data repositories (CDR) where these data-driven transformations are performed to form unified data models for reporting systems and clinical analytics [13]. These repositories usually retrieve data either nightly or weekly and introduce latency that constrains real-time decision support. The source systems, as well as the target analytical perspectives, increase exponentially with the number of transformation systems available.

Clinical information can be presented as semi-structured documents containing defined sections, which is the hallmark of document-centric model as represented in Clinical Document Architecture [14]. While appropriate for clinical workflows where providers collate on narrative notes, this model creates obstacles to systematic data extraction required for decision support systems and quality reporting.

The HL7 FHIR resources establish standardized representations for a unified clinical information exchange across the clinical data exchange, with separate definitions (resources) that include patients, observations, medications, and procedures [15]. FHIR facilitates inter-organizational interoperability but does not dictate the internal data architecture. There is still a gap between FHIR representations and internal operational databases.

They each reflect design choices that favor various stakeholder views. Transactional databases help optimize clinical operational needs. Data warehouses provide data for

analysis and reporting. Documenting is consistent with provider documentation workflows. FHIR facilitates the exchange of external data. No single model can comprehensively address all views at the same time.

2.3. Patterns of Event-Driven Architecture

The architectural patterns of software engineering have served multiple perspectives with different requirements by designing systems for each perspective. Event sourcing keeps the state of the system a single immutable sequence of events, instead of current state records that can be changed [4]. Applications replay events to reconstruct state at any point in time, allowing them to create temporal queries and audit trails. Event sourcing naturally separates event capture from state interpretation, allowing different consumers to project events into perspective-specific views.

Command Query Responsibility Segregation (CQRS) distinctively separates write models optimized for transactional consistency from read models optimized for query performance. Write models ensure business rule enforcement and data integrity. Read models convert data into query-optimized structures and possibly can use multiple read models depending on the query patterns. CQRS recognizes that appropriate write and read representations are fundamentally different.

Domain-driven design acknowledges that complex domains have many bounded contexts, each with its own model and common language [5]. Bounded contexts define distinct boundaries in which certain models can be employed with clear integration patterns for cross-context communication. Various stakeholder groups represent different bounded contexts that need to use different models of the same domain entities.

Materialized views preserve precomputed query results, sacrificing storage for query performance [16]. Instead of repeating complex queries, systems calculate results once and gradually update them as changes to underlying data happen. Several materialized views may co-exist together; each being optimized for distinct query patterns. Despite their importance in multi-perspective problems, these patterns do not have wide acceptance in healthcare systems. Despite EHR implementation, almost all of its systems still feature transactional databases with nightly batch processes for the analytical views that miss opportunities for real-time perspective-specific projections.

2.4. Research Gaps in the Current Health IT Literature

In fact, many studies in the healthcare informatics literature discuss stakeholder requirements, usability issues, or interoperability problems, but little more systematic catalogs of architectural patterns exist for identifying multi-perspective conflicts. Clinical workflow studies identify points of friction when system designs do not align with clinical needs [1], [3], and do not propose architectural remedies in their review. The field of interoperability studies primarily has been on data exchange standards like FHIR [15] with little discussion on internal system architecture that supports different local viewpoints.

Software engineering pattern languages systematically record repeatable design difficulties and mitigations, but few catalogs consider constraints unique to healthcare like regulatory requirements, patient safety criticality, and the health user ecosystem in the context of clinical regulatory settings. The use of event sourcing and CQRS patterns has been previously performed in financial systems, e-commerce, and other domains [4]. However, no published literature on healthcare was made available; and these features are underrepresented in applications today.

This gap is what motivates the pattern catalog. Using recognized architectural patterns for documented clinical scenarios we will develop reusable solutions in the context of perspective conflicts occurring in healthcare information systems. The impact connects clinical informatics and software architecture through proven patterns and is translated to clinical use by healthcare technologists.

3. Taxonomy of Perspective Conflicts

Before documenting specific patterns, a taxonomy is established classifying perspective conflicts by the fundamental nature of stakeholder requirement mismatches. This taxonomy helps architects diagnose which patterns apply to situations and understand relationships between patterns.

3.1. Temporal Conflicts

Temporal conflicts arise when different perspectives require different temporal representations of the same events. Clinical documentation may record when events occurred, while billing systems care when charges should be posted. Quality measures may calculate metrics based on calendar dates, while clinical episodes span variable timeframes crossing calendar boundaries. Audit systems need precise timestamps for all state changes, while operational dashboards aggregate to hourly or daily granularities.

These conflicts manifest as disagreements about event timestamps, sequencing, and temporal boundaries for aggregation. Resolution strategies typically involve maintaining multiple timeline representations or temporal projections of the same underlying events.

3.2. Semantic Conflicts

Semantic conflicts occur when the same data elements hold different meanings across perspectives. A medication order represents a clinical decision to a physician, a task to a nurse, a billable charge to revenue cycle staff, and a compliance requirement to quality officers. Laboratory results indicate diagnostic findings to clinicians but represent workflow completions to laboratory operations.

The challenge extends beyond simple labeling differences to fundamental semantic interpretation. What constitutes a "completed" event varies by perspective. Clinical completion may precede documentation completion,

which may precede billing completion. Resolution requires recognizing that multiple semantic interpretations are simultaneously valid.

3.3. Granularity Conflicts

Granularity conflicts arise when perspectives require different levels of detail about the same events. Patient safety monitoring may need every detail of medication preparation and administration, while workflow management needs only task completion status. Financial forecasting aggregates monthly service volumes, while operational staffing requires hourly census projections.

These conflicts cannot be resolved by simple aggregation or disaggregation. Increasing granularity requires collecting data not needed by coarser perspectives. Decreasing granularity loses information required by finer perspectives. Different perspectives legitimately need different granularities simultaneously.

3.4. Identity and Context Conflicts

Identity conflicts occur when entities are identified differently across perspectives. Patients have medical record numbers for clinical purposes, member identifiers for insurance, account numbers for billing, and anonymous research identifiers for de-identified datasets. The same medication may be identified by brand name, generic name, NDC code, or therapeutic class depending on perspective.

Context conflicts arise when the same entity plays different roles in different bounded contexts. A clinical encounter represents a care delivery event in the clinical context, a billing claim in the revenue context, and a quality measurement opportunity in the compliance context. These are not simply different views of the same thing but legitimately different conceptual entities that happen to correspond to the same real-world event.

This taxonomy provides a framework for pattern identification. The four patterns documented in Section IV instantiate these conflict types in specific clinical scenarios.

4. Perspective Conflict Patterns

This section presents four detailed patterns representing commonly encountered perspective conflicts in clinical event systems (Fig. 1). Each pattern follows a standard structure: Context, Problem, Forces, Solution, Implementation, Consequences, and Related Patterns.

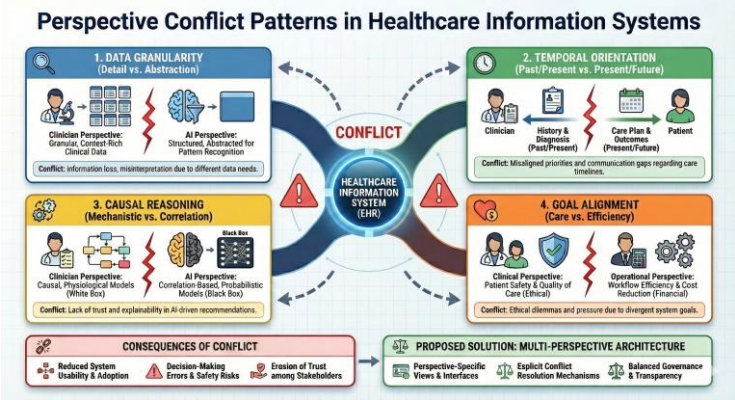

Figure 1. Perspective Conflict Patterns in Healthcare Information Systems

Table 1 provides a concise overview of the four patterns identified in this catalog. Each pattern addresses a specific type of conflict arising when multiple stakeholders require different representations of the same clinical events. The patterns are described in detail in the following subsections, including clinical context, competing forces, implementation considerations, and trade-offs.

Table 1. Architectural Conflict Patterns and Event-Driven Resolution Strategies

Pattern	Conflict Type	Solution
Safety versus Workflow Granularity	Fine-grained safety detail versus coarse-grained workflow status	Separate projections for safety history and workflow completion
Billing versus Clinical Timing	Multiple valid timestamps for single event	Event store captures all timestamps; projections expose relevant timing per context
Identity Context Dependence	Patient identifiers differ across organizational boundaries	Context-specific identifiers with authorized mapping services
Compliance Audit versus Performance Optimization	Audit completeness versus query performance	Event store for audit; materialized views for operational queries

4.1. Pattern 1: Safety versus Workflow Granularity

Context: Clinical workflows require task completion tracking while patient safety monitoring demands detailed event documentation. These perspectives need fundamentally different granularities for the same activities.

Problem: Medication administration appears as a binary task from workflow perspectives: administered or not administered. Workflow systems show nurses pending tasks and accept completion confirmations with minimal documentation burden.

Patient safety requires comprehensive detail: exact dose, route, administration time, witnessing staff for high-risk medications, patient response, and deviations from ordered parameters [17]. Error analysis depends on reconstructing precise event sequences. Regulatory requirements mandate detailed documentation for controlled substances.

Forcing workflow systems to capture safety-level detail creates documentation friction and workarounds [3]. Presenting only completion status to safety systems loses critical information for adverse event detection.

Forces: Workflow efficiency requires minimal documentation burden. Patient safety demands comprehensive detail for error detection. Cognitive load management requires simple interfaces during time-pressured activities. Regulatory compliance mandates detailed audit trails for specific medications. Error analysis requires reconstructing exact sequences.

Solution: Implement hierarchical event structures where workflow events represent task completions while detailed safety events capture comprehensive parameters. Workflow systems subscribe to completion events only. Safety monitoring systems subscribe to detailed events with all required parameters.

Clinical applications publish detailed medication administration events containing dose, route, time, witnessing staff, patient response, and deviations. Workflow projections extract completion status. Safety projections maintain full event detail for interaction checking and compliance verification.

Implementation:
Event schema captures comprehensive administration data:
MedicationAdministrationEvent {
eventId, timestamp, patientId, medicationOrderId
administeredDrug: { code, displayName, dose }
route, site, administeredBy, witnessedBy
patientResponse, deviations: [{ parameter, ordered, actual, reason }]
}
Workflow projection maintains simplified state:
WorkflowTaskProjection {
taskId, patientId, taskType, status, completedBy, completedAt
}

Safety projection retains full event history with interaction warnings and compliance flags.

Consequences: Clinical staff interact with simplified workflow interfaces. Safety systems receive comprehensive details for adverse event detection. Single event capture supports multiple perspectives without duplicate entry. Event immutability provides audit trails.

Trade-offs include increased system complexity from multiple projections, higher storage requirements, and potential eventual consistency windows between event publication and projection updates.

Related Patterns: Exemplifies Granularity Conflicts. Relates to Compliance Audit versus Performance Optimization (Pattern 4) in maintaining detailed events for compliance while providing optimized views. Shares event sourcing architecture with Billing versus Clinical Timing (Pattern 2).

4.2. Pattern 2: Billing versus Clinical Timing

Context: Healthcare services generate clinical events requiring documentation and financial events requiring charge of capture. These have different timing requirements reflecting distinct business processes.

Problem: Consider diagnostic imaging procedures. Clinically relevant timestamps include when ordered, scheduled, performed, interpreted, and results communicated [18]. Clinical workflows and quality measures reference actual performance times.

Billing perspectives recognize chargeable events when orders are placed or services scheduled, not when performed. Revenue recognition rules and payer contracts determine charge posting timing independent of clinical timing. Bundled payments may map multiple clinical services to single billing events at episode level.

Using billing timestamps for clinical purposes misrepresents care delivery timing and corrupts quality measures. Forcing billing timestamps to match clinical creates revenue recognition issues.

Forces: Revenue recognition rules specify charge posting timing for financial reporting. Clinical documentation standards require accurate care timing. Quality measures calculate metrics from clinically relevant timestamps. Payer contracts define billing rules misaligned with clinical timing. Financial forecasting aggregates by posting date while clinical analytics aggregate by performance date.

Solution: Maintain separate clinical and billing event streams with explicit relationships. Clinical events record actual care delivery timestamps. Billing events record charge posting per revenue cycle rules. Relationships link billing events to corresponding clinical events.

Clinical systems subscribe to clinical streams. Billing systems subscribe to billing streams. Analytics requiring reconciliation use relationship links for correlation.

Implementation
Clinical events capture service delivery timing:
DiagnosticImagingPerformedEvent {eventId, performedTimestamp, patientId, orderingProviderId performingTechnologistId, procedureCode, modalityType, studyInstanceUID}

Billing events capture charge posting:
ServiceChargeEvent {eventId, chargeTimestamp, patientAccountId, serviceDate
chargeCode, chargeAmount, payerContractId
relatedClinicalEventId (reference)}

Clinical analytics query clinical timestamps. Financial reporting queries charge timestamps. Reconciliation analysis joins on relatedClinicalEventId when needed.

Consequences: Clinical documentation maintains accurate delivery timestamps without financial constraints. Billing applies to appropriate revenue recognition rules. Quality measures are calculated correctly using clinical timestamps. Financial reporting follows accounting requirements. Relationship links enable reconciliation.

Trade-offs include managing separate event streams and requiring join operations for correlated analytics. Data quality depends on maintaining accurate relationship links.

Related Patterns: Represents Temporal Conflicts. Shares architectural approaches with Safety versus Workflow Granularity (Pattern 1) using separate event streams and projections. Relates to Identity Context Dependence (Pattern 3) in recognizing different bounded contexts requiring different representations.

4.3. Pattern 3: Identity Context Dependence

Context: Healthcare involves multiple organizational contexts identifying the same entities differently for legitimate operational reasons. Clinical operations use medical record numbers. Insurance uses member identifiers. Devices use encounter-specific identifiers. Research requires de-identified pseudonyms.

Problem: Patients receiving care at multi-facility health systems have separate medical record numbers at each facility reflecting independent registration [19]. Insurance assigns member identifiers unrelated to medical record numbers. Bedside monitoring devices identify patients using admission numbers. Research studies use pseudonyms to protect privacy through de-identification.

Master patient indexes attempt mapping all identifiers to canonical identifiers. However, different contexts require context-specific identifiers for legitimate reasons. Clinical systems cannot use insurance identifiers because members change plans while maintaining clinical records. Research pseudonyms must not deterministically link to clinical identities per IRB protocols [12].

Forcing unified identifiers creates operational friction and privacy risks. Medical record numbers lack structure for cross-organizational use. Using insurance identifiers for clinical purposes breaks when patients change coverage.

Forces: Clinical operations require facility-unique identifiers for patient safety. Billing requires identifiers aligned with payer systems. Device integration requires encounter-scoped temporary identifiers. Research requires non-reversible pseudonyms. Patient matching across organizations requires robust linkage. Privacy regulations mandate controlled linkage with audit trails.

Solution: Implement context-bounded identifiers where each bounded context maintains its own namespace with explicit, controlled linkage. Clinical events use medical record numbers. Billing events use member identifiers. Research events use pseudonyms. A separate identity linking service provides controlled translation between contexts, enforcing access controls, and maintaining audit trails.

Implementation:
Each context maintains events using context-appropriate identifiers:
ClinicalEncounterEvent {eventId, timestamp, patientMRN (facility-scoped) facilityId, encounterId, encounterType}

BillingClaimEvent {eventId,timestamp, subscriberMemberId (payer-scoped) payerId, claimId, relatedEncounterId}

ResearchObservationEvent {eventId, timestamp, researchSubjectId (study-scoped pseudonym) studyId, observationType, observationValue}

Identity linking service provides controlled translation with authorization enforcement, access logging, and audit trails for all linkage operations.

Consequences: Each bounded context uses appropriate identifiers for operational needs. Cross-context linkage is explicit, controlled, and auditable. Privacy protections strengthen through security boundaries. Research de-identification maintains formal separation. Authorization and audit controls apply at linkage points.

Trade-offs include complexity of multiple identifier namespaces, linkage service latency, and dependence on accurate linkage maintenance. Identity linking service becomes a critical infrastructure requiring high availability.

Related Patterns: Represents Identity and Context Conflicts. Shares bounded context concepts with Billing versus Clinical Timing (Pattern 2). Relates to Compliance Audit versus Performance Optimization (Pattern 4) requiring explicit audit trails for identity linkage.

4.4. Pattern 4: Compliance Audit versus Performance Optimization

Context: Regulatory compliance requires comprehensive audit trails documenting all access to protected health information and changes to clinical data [20]. Operational systems require high-performance queries supporting real-time clinical workflows with sub-second response times.

Problem: HIPAA mandates tracking who accessed which patient records, when, for what purpose, and from where [20]. Clinical data integrity requires recording all changes with

timestamps and responsible parties [21]. Patient safety event analysis requires reconstructing complete activity sequences.

Audit requirements suggest append-only logs with complete event history. However, operational queries need current patient state optimized for rapid retrieval. Medication lists require current active medications, not historical administration logs. Laboratory displays need the most recent values, not complete history.

Storing data exclusively in append-only formats creates query performance problems. Retrieving current medication lists from complete administration history requires expensive filtering and aggregation [22]. Storing exclusively in current-state formats loses audit trail capabilities, creating compliance violations.

Forces: Regulatory compliance requires immutable audit trails. Query performance requires denormalized current-state structures. Storage costs increase when maintaining both logs and views. Audit queries occur infrequently and tolerate higher latency. Operational queries occur constantly and require a sub-second response. Data retention policies specify long-term audit preservation.

Solution: Implement event sourcing with audit-complete event logs and operational materialized views. All system events are recorded in immutable append-only logs serving as source of truth for compliance. Operational systems query materialized views projecting current state optimized for common access patterns. Views rebuild from event logs when needed.

Implementation:
Event store maintains complete audit trail:
PatientDataAccessEvent {eventId, timestamp, userId, userRole, patientId dataCategory, accessPurpose, sourceIPAddress, sessionId}

MedicationOrderChangedEvent {eventId, timestamp, orderId, patientId, changeType previousState, newState, changedBy, changeReason}

Events are written to append-only stores and never modify. Audit queries directly access event store.
Operational materialized view maintains current medication list:
CurrentMedicationView {patientId, orderId, medicationCode, medicationName dose, route, frequency, orderingProvider startDate, status, lastModified}

View builds by consuming Medication Order Changed Event stream. Operational queries access materialized view with high performance. Views rebuild from event log if corrupted.

Consequences: Audit requirements are satisfied through complete immutable logs. Operational queries achieve high performance through optimized views. Event log serves as source of truth enabling view reconstruction. Different retention policies apply to views versus audit logs. Multiple operational views can coexist.

Trade-offs include increased storage for both logs and views, eventual consistency creating brief lag windows, and view maintenance overhead.

Related Patterns: Represents Granularity Conflicts and Temporal Conflicts. Shares event sourcing architecture with Safety versus Workflow Granularity (Pattern 1) and Billing versus Clinical Timing (Pattern 2). Addresses compliance requirements like Identity Context Dependence (Pattern 3) requiring explicit audit trails.

5. Reference Architecture

The patterns reported in Section IV have similar architectural features regarding event sourcing, materialized views, and bounded contexts. In this section, reference architecture is provided which combines these ideas into an integrated system design for multi-perspective clinical event systems.

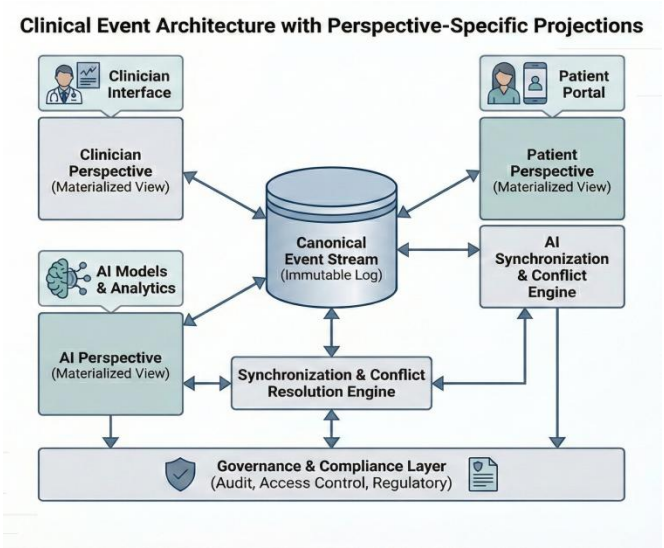

Figure 2. Clinical Event Architecture with Perspective-Specific Projections

5.1. Core Components

The architecture comprises 5 main parts: event capture, event store, projection engines, materialized views, and bounded context services. Event capture components publish clinical events to the event store as care activities occur. Clinical applications with event publishing capabilities generate events when medications are administered; orders are placed, results are documented, and other important clinical events take place. Events are immutable facts, explaining what happened, when it happened, who was involved, and relevant clinical parameters.

Event stores offer durable, append-only storage capacity for all published events. The event store serves as an extensive record to record and preserve every event to prevent future events. Events are never changed or eliminated, contributing to full audit trails which support

temporal queries. The event store also utilizes subscription functions for downstream consumers, so that they can receive events as they are published. Partitioning and indexing methods improve the append performance and the sequential reads.

Projection engines feed events from the event store and keep perspective-specific materialized views. Translating source events into correct view structures is done with the logic implemented by each projection engine. Projection engines run independently, so each perspective can keep different update frequencies and consistency models.

Materialized views manage, through views, a view-specific interpretation of event data that is optimized for specific query patterns. Views can de-normalize data, compute aggregations beforehand, or order event data into structures that correspond to stakeholder mental models. These views have their own bounded and queried contexts.

Bounded context services offer APIs for specific stakeholder groups, which query suitable materialized views and apply context-based authorization policies. Services switch, on the other hand, between internal view structures and external APIs with formats and terminology matching the stakeholder context.

5.2. Event Flow

Clinical events run through the architecture in various stages. Clinical applications publish events to the event store as care activities happen. The event store durably persists events, and they are released to subscribers. Projection engines subscribe to the correct event types and update materialized views incrementally. Bounded context services process the materialized views to act on stakeholder requests.

This flow distinguishes event capture from event interpretation. On a clinical level, applications aim to represent events accurately and are not prepared to consider what points of view will consume events. Projection engines convert events into point of view-aware representations. Bounded context services provide an interface easy enough for all interested parties to manage, but which conceals much underlying complexity.

5.3. Technology Considerations

Several choices of technology influence the road to successful implementation. Event stores need databases for append-optimized writes and sequential reads. Specialized event stores like Event Store, or PostgreSQL with append-optimized tables for event stores and other event stores. Partitioning by patient or time allows data to scale and supports data retention policies.

For high-throughput scenarios, or for relatively low-volume applications, projection engines may rely on stream processing frameworks (e.g., Apache Kafka Streams, Apache Flink) or a less complex polling mechanism. Idempotent projection logic is responsible for handling duplicate event delivery which is frequent within distributed systems.

Materialized views can rely on relational databases for structured query support, or document stores when they want to allow flexible schema evolution. Caching layers take load off view databases when data is being accessed frequently.

Event extraction from legacy databases is required for EHR system integration. Change data capture mechanisms, database triggers, or application-level event publishing input events from existing systems into an event store. Slow migration methodologies permit adoption and gradually transition without an entire system overhaul.

5.4. Operational Characteristics

The architecture finally becomes consistent from event publication to view availability. While events are durably stored immediately, projection updates are asynchronously updated. This introduces short timescales in which disparate views might display slightly different states. For the vast majority of clinical applications, this latency (usually around milliseconds to seconds) is acceptable. For example, a strong consistency application can directly query the event store but at the expense of higher latency and complexity.

Independent component scaling provides scalability. Partitioning scales an event store. Projection engines scale by adding workers consuming from partitions. Materialized views in this case scale through database replication and caching. Depending on the load characteristics of each perspective, each can scale independently.

Fault tolerance is an issue when dealing with projection failures that need to be addressed carefully. Idempotent projection logic is responsible for ensuring that replay of events creates the desired view state. Checkpoint mechanisms follow projections back to their state and enable resumption from their proper position after the projection fails. View rebuild capabilities enable recovery from corruption through replaying events from the source of truth.

6. Discussion

The proposed pattern catalogue and reference architecture in this paper respond to the gaps in literature on the design of healthcare information systems and offer methodical approaches to managing multi-perspective conflicts. However, certain factors impact it from a perspective of applicability and adoption.

6.1. Applicability Boundaries

These patterns are most apparent for the adoption of large-scale healthcare information systems that serve multiple stakeholders with heterogeneous needs. For smaller systems with homogeneous user populations, perspective conflicts might not be strong enough to justify architectural complexity. Single-department applications focused on a specific workflow could accomplish the task with simpler unified data models.

While identifying these recurring patterns, organizations should consider multiple perspectives. Multi-perspective architecture benefits systems with multiple stakeholder

groups who have documented conflicts in their existing designs. Event sourcing approaches are useful for a wide variety of applications with regulatory audit requirements. Perspective-specific projection can be useful for organizations having issues with data quality from stakeholders trying to deal with inappropriate interfaces.

On the other hand, the complexity of an application dominated by a single perspective may not make it necessary. Greenfield approaches without the constraints of legacy integration may require simpler designs. Capability gaps that companies, which do not have the technical know-how to facilitate the event-driven architecture, would have to fill before the project.

6.2. Implementation Challenges

The adoption of these patterns demands organizational and technical capabilities that other healthcare IT environments may not possess. Unlike CRUD applications, event-driven architectures develop using different practices. Event sourcing, eventual consistency, and stream processing concepts need to be understood by teams.

Challenges are in integrating with existing EHR systems. Legacy databases (not specifically developed for event extraction) need to be improved through technology to capture change data or event publication mechanisms. Incremental migration strategies enable gradual adoption but add some level of added complexity because both old and new architecture support has to be taken into account for a bit.

Organizational change management may be more difficult than technical implementation. As data-sharing databases have been adopted in other stakeholder groups, this is often resisted by bounded context separation. When it comes to clinical staff, they may require education about eventual consistency and what that means for their workflows. IT operation entails acquiring new skills to handle the management of event stores and projection engines.

6.3. Relationship to Standards

The HL7 FHIR/similar industry-relevant interoperability standards cover data exchange instead of system-level system architecture. FHIR resources can act as event payload formats, but FHIR does not prescribe event sourcing nor the architecture of multi-perspective architectures. In contrast, the patterns described here are complementary to FHIR, offering internal architecture that supports multiple perspectives. However, exposing standardized FHIR APIs externally.

By nature, clinical quality measure specifications provide calculation logic but not underlying data architecture. Such multi-perspective architecture can facilitate quality measure computation by having a dedicated quality measure projection which does not need every part of the system to understand measure logic.

6.4. Future Study Directions for Research

Appropriate implementation case studies would provide empirical validation in the form of an “experimental” demonstration of the practicality and measure performance features of the system; real-world applications should provide evidence of the practicality of these principles and performance properties. Some methods of formal procedures could confirm consistency of properties between points of view and detect situations that are not able to resolve conflicts automatically.

It is feasible that there are other probable patterns besides the four listed here. These conflicts are yet to be cataloged, and systematic analysis of EHR implementations in multiple organizations could reveal recurring conflicts. Patient-facing perspectives, such as patient portals and personal health records, bring out further stakeholder needs (not fully addressed) in this paper.

More investigation into integration patterns bridging multi-perspective clinical systems and external health information exchanges, payer systems, and public health reporting must be conducted instead. Clinical artificial intelligence and other emerging fields in machine learning create new points of view as models feed clinical information for prediction and recommendations.

7. Conclusion

Healthcare information systems must cater to diverse stakeholders with diverging needs for different information and decision requirements. The present architecture challenges the needs and wants of stakeholders by driving them to centralized, unified data models that may lead to problems of user experience, data quality, and clinical workflow that ultimately conflict with the underlying design. This document has provided a catalog of perspective conflict patterns of the four main types often found in clinical event systems, documented in clinical event-related systems, noting four basic patterns: Safety versus Workflow Granularity, Billing versus Clinical Timing, Identity Context Dependence, and Compliance Audit versus Performance Optimization.

Each pattern is designed to pinpoint one particular stakeholder conflict, articulate competing requirements, and develop architectural solutions based on event sourcing and domain-driven design principles. The reference architecture illustrates the way to apply them in the coherent designs of systems, including event stores, projection engines, and materialized views, to serve multiple perspectives without causing any transformation complexity or data duplication.

The pattern catalog offers healthcare architects and informaticists a list of reusable answers for multi-perspective issues. Instead of treating each stakeholder conflict as a unique problem, organizations can identify repeated patterns and draw documented solutions. The stepwise approach makes diagnoses based on patterns that apply to different situations and trade-offs made by different conflict resolution strategies.

Future research should aim at empirical validation based on implementation case studies, further extending the pattern catalog to include other conflict types, and tooling development for pattern-based system design. With healthcare IT being constantly shifting toward value-based care, population health management, and clinical artificial intelligence, multi-perspective architecture will be pivotal to be able to accommodate a range of stakeholder needs across unified clinical event streams.

References

[1] B. G. Arndt et al., "Tethered to the EHR: Primary Care Physician Workload Assessment Using EHR Event Log Data and Time-Motion Observations," *The Annals of Family Medicine*, vol. 15, no. 5, pp. 419–426, Sep. 2017, doi: https://doi.org/10.1370/afm.2121.

[2] A. Alanazi, W. Alalawi, B. Aldosari, "An Evaluation of Drug-Drug Interaction Alerts Produced by Clinical Decision Support Systems in a Tertiary Hospital," *Cureus*, vol. 15, no. 8, Aug. 2023, doi: https://doi.org/10.7759/cureus.43141.

[3] J. Howe et al., "Electronic health record usability issues and potential contribution to patient harm," *JAMA*, vol. 319, no. 12, pp. 1276-1278, Mar. 2018.

[4] M. Fowler, "Event Sourcing," *martinfowler.com*, Dec. 12, 2005. https://martinfowler.com/eaaDev/EventSourcing.html.

[5] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston, Mass.; Munich: Addison-Wesley, 2003.

[6] T. T. Van Vleck and Noémie Elhadad, "Corpus-Based Problem Selection for EHR Note Summarization," *AMIA Annual Symposium Proceedings*, vol. 2010, pp. 817-821, Nov. 2010, Accessed: Feb. 16, 2026. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC3041431/.

[7] American Medical Association, "CPT overview and code approval," *AMA CPT*, 2024. [Online]. Available: https://www.ama-assn.org/practice-management/cpt/cpt-overview-and-code-approval

[8] D. M. Berwick and A. D. Hackbarth, "Eliminating Waste in US Health Care," *JAMA*, vol. 307, no. 14, pp. 1513-1516, Apr. 2012, doi: https://doi.org/10.1001/jama.2012.362.

[9] CMS, "Quality Measures | CMS," *www.cms.gov*, Sep. 06, 2023. https://www.cms.gov/medicare/quality/measures.

[10] A. W. Wu, T. A. Cavanaugh, S. J. McPhee, B. Lo, and G. P. Micco, "To tell the truth," *Journal of General Internal Medicine*, vol. 12, no. 12, pp. 770–775, Dec. 1997, doi: https://doi.org/10.1046/j.1525-1497.1997.07163.x.

[11] The Joint Commission, "National patient safety goals," *The Joint Commission*, 2025. https://www.jointcommission.org/standards/national-patient-safety-goals/.

[12] "Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule," 2012. Available: https://privacysecurityacademy.com/wp-content/uploads/2021/03/HHS-OCR-Guidance-on-De-Identification-of-PHI-2012.pdf.

[13] N. Menachemi and T. Collum, "Benefits and drawbacks of electronic health record systems," *Risk Management and Healthcare Policy*, vol. 4, pp. 47–55, May 2011, Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3270933/.

[14] R. H. Dolin et al., "HL7 Clinical Document Architecture, Release 2," *Journal of the American Medical Informatics Association*, vol. 13, no. 1, pp. 30–39, Jan. 2006, doi: https://doi.org/10.1197/jamia.m1888.

[15] HL7 International, "Index - FHIR v4.0.1," *Hl7.org*, 2019. https://hl7.org/fhir/R4/index.html.

[16] J. Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals" *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 29–53, 1997, doi: https://doi.org/10.1023/a:1009726021843.

[17] M. Grissinger, "Guidelines for Standard Order Sets," *Pharmacy and Therapeutics*, vol. 39, no. 1, pp. 10-50, 2014, Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC3956384/.

[18] D. A. Kurth, B. K. Karmazyn, C. A. Waldrip, M. Chatfield, and M. E. Lockhart, "ACR Appropriateness Criteria® Methodology," J*ournal of the American College of Radiology*, vol. 18, no. 11, pp. S240–S250, Nov. 2021, doi: https://doi.org/10.1016/j.jacr.2021.03.021.

[19] J. R. Vest and L. D. Gamm, "Health information exchange: Persistent challenges and new strategies," *Journal of the American Medical Informatics Association*, vol. 17, no. 3, pp. 288–294, May 2010, doi: https://doi.org/10.1136/jamia.2010.003673.

[20] U.S. Department of Health and Human Services, "The Security Rule," *HHS.gov*, Oct. 20, 2022. https://www.hhs.gov/hipaa/for-professionals/security/index.html.

[21] Institute of Medicine, "Health IT and patient safety: building safer systems for better care." *Washington, D.C.: National Academies Press*, 2012.

[22] D. F. Sittig and H. Singh, "A new sociotechnical model for studying health information technology in complex adaptive healthcare systems," *Quality and Safety in Health Care*, vol. 19, no. Suppl 3, pp. i68–i74, Oct. 2010, doi: https://doi.org/10.1136/qshc.2010.042085.