

Beyond Usagi

A Tiered Source-to-Concept Mapping Pipeline with AI-Assisted Context Resolution

Technical White Paper

September 2026

Roger Carlson

SASS Honest Broker Analytics

Corewell Health Research Institute, Grand Rapids, MI

Summary

Source-to-Concept Mapping (STCM) bridges the gap between institutional source codes and OHDSI standard concepts. The standard community tool, Usagi, performs term similarity matching between source descriptions and OMOP concept names. This works when the description is a full English phrase. It fails for abbreviated codes, institution-specific shorthand, numeric identifiers, and context-dependent codes where the same label means different things in different departments.

At Corewell Health (22 hospitals, 7.4 million patients), we replaced Usagi entirely with a four-tier pipeline running within Snowflake. The pipeline discovered 12,781 source codes across 39 vocabulary domains and resolved 100% of them, producing 8,022 production STCM rows and 6,127 custom concepts. An impact tier framework classifies every vocabulary domain by failure severity — from silent data loss (Tier 1) to documentation gaps (Tier 4) — so that reviewers and build gates prioritize accordingly. A Streamlit review interface replaced the Usagi desktop experience. A community starter seed of 1,897 pre-resolved mappings is portable to other Epic-on-Snowflake sites. Zero lines of Java were involved.

This paper describes each tier, the architecture, the impact tier framework, the results, and what generalizes.

Introduction

Usagi is a desktop Java application designed for a workflow where an analyst exports source codes, reviews them one by one in a swing-based interface, and exports the results to a CSV. At institutional scale — thousands of codes across dozens of vocabulary domains, two merged Epic instances, ongoing maintenance as source data evolves — this workflow does not hold.

The fundamental limitation is not Usagi's matching algorithm. It is the information available to the matching process. A code labeled "Rate" has no useful term similarity to any OMOP concept. But that same code, appearing in ICU encounters alongside oxygen saturation readings and ventilator settings in

the respiratory therapy department, is obviously respiratory rate. The disambiguation signal exists in the source data. Current tools do not access it.

The STCM Problem at Scale

At Corewell Health, we maintain STCM mappings across 39 vocabulary domains. These cover every CDM field that accepts local codes:

- Visit types (ambulatory, inpatient, ED, telehealth encounter classifications)
- Demographics (race, ethnicity, gender identity, sexual orientation)
- Vitals and flowsheets (blood pressure, temperature, respiratory rate, BMI)
- Clinical scoring (GCS, SOFA, MEWS, APACHE, Aldrete, PELOD, sepsis predictive models)
- Behavioral health screening (PHQ-2/4/9, GAD-7, AUDIT-C, Edinburgh, C-SSRS, PCL-5, Beck, Mini-Cog)
- Ventilator parameters (PEEP, tidal volume, FiO2, airway pressure, minute ventilation)
- Oxygenation (SvO2, ScvO2, PaO2, blood gas saturations, pulse oximetry)
- Anesthesia gas monitoring (sevoflurane, nitrous oxide)
- Fall risk assessments (predictive model scores, Morse scale)
- Intake/output (urine output, drain volumes)
- Measurement units (mg/dL, mEq/L, /HPF — hundreds of unit strings)
- Medication routes and frequencies
- Procedure names (surgical, imaging, infusion)
- Device types (lines, drains, airways, surgical instruments, implants)
- Specimen types and anatomic sites
- Social determinants of health (housing, food insecurity, tobacco, substance use)
- Note types and classifications
- Place of service codes
- Provider specialties

Prior to the category split described in this paper, seven of these domains — clinical scoring, behavioral health, ventilator, oxygenation, anesthesia gas, fall risk, and intake/output — were collapsed into a single “miscellaneous” flowsheet category containing 3.5 billion measurement rows. The catch-all label communicated nothing about clinical purpose, failure mode, or review priority.

When a source code lacks an STCM entry, the ETL assigns `concept_id = 0`. The record enters the CDM but becomes semantically invisible: no vocabulary query finds it, no hierarchical traversal includes it, no standard-concept-based cohort definition captures it. The data is present but useless.

At submission time (May 2026), approximately 4,200 codes were identified as unmapped. By September 2026, the discovery pipeline had expanded to 12,781 source codes across 39 vocabulary domains. The difference: the initial inventory was hand-curated from known vocabulary domains. The automated pipeline discovered codes in domains we did not know existed.

Methods

Tier 0: Auto-Disposition (Pattern Rules)

When the discovery pipeline runs against an EHR, it finds everything – not just clinical concepts but system artifacts, workflow flags, formatting indicators, and free-text noise. At Corewell Health, roughly a third of the 12,781 discovered codes fell into this category. Sending all of them to human review, or even to an LLM for candidate generation, would waste time and money on codes where the answer was always going to be “exclude this.”

Tier 0 runs before any matching – deterministic or AI-assisted. Its purpose is to remove codes that are definitively non-clinical, using deterministic pattern rules that cost nothing: no AI credits, no human review time.

The pattern table. A CSV seed (`stcm_not_applicable_patterns.csv`) drives the rule engine. Each row specifies four fields:

Field	Purpose
<code>source_vocabulary_id</code>	Which vocabulary domain the rule applies to
<code>pattern_type</code>	How to match: <code>equals</code> , <code>contains</code> , <code>regex</code> , or <code>all</code>
<code>pattern_value</code>	The string to match against the source code description
<code>reason</code>	Why this code is being excluded (the audit trail)

The four pattern types cover different use cases. `equals` handles exact matches – a unit string of “YYYYMMDD” is a date format indicator, not a measurement unit. `contains` handles substring patterns – any microbiology result containing “flora” or “colony” is a free-text culture narrative, not a coded value. `regex` handles more complex patterns when substring matching is too broad. And `all` handles the case where an entire vocabulary is irrelevant to the STCM pipeline.

The `all` type is particularly useful. Our flowsheet unit vocabulary contains hundreds of codes, but the ETL handles flowsheet units through a direct column mapping, not through STCM. One line in the pattern seed excludes the entire vocabulary unconditionally. Similarly, our allergy vocabulary (9,104 codes) stores allergy names as source_value text – they are never concept-mapped through STCM. One `all` rule cleared the largest single vocabulary in the pipeline.

Why patterns, not a blacklist. A blacklist enumerates specific codes. It breaks when new codes appear next quarter. Patterns generalize. “Contains ‘CDI’” catches every Clinical Documentation Improvement note class that Epic has now and every one they add in the future. “Contains ‘Retired’” catches every retired template regardless of which specific template was retired. The pattern library grows slowly while the source data grows fast.

Example rules from our production installation:

Vocabulary	Pattern	Type	Reason
OEE_UNIT	YYYYMMDD	equals	Date format indicator, not a measurement unit
OEE_NOTE_CLASS	CDI	contains	Clinical Documentation Improvement billing/coding audit tool
OEE_NOTE_CLASS	AVS Snapshot	contains	Machine-generated After Visit Summary; system artifact
OEE_VALUE_AS_CONCEPT	flora	contains	Microbiology free-text culture result
OEE_DEVICE_LDA	Wound Assessment	contains	Wound documentation template, not a physical device
OEE_ANATOMIC_SITE	Autoclave Check	equals	QC/sterilization check, not a clinical specimen
OEE_FLOWSHEET_UNIT	(all)	all	ETL handles flowsheet units via direct lookup, not STCM
OEE_ALLERGY	(all)	all	Allergy names stored in source_value, not concept-mapped

The audit trail. Every auto-excluded code receives a row in the mapping decisions table with `match_method = 'AUTO_RULE'`, `reviewed_by = 'stcm_auto_disposition'`, and the reason text from the pattern seed. A human reviewer can query all auto-excluded codes, disagree with any of them, and override by inserting a new decision. The pattern is a default, not a permanent verdict.

Interaction with impact tiers. The cost of a false exclusion depends on the vocabulary's impact tier. For Tier 1 (CRITICAL) vocabularies where the ETL uses an INNER JOIN on STCM, excluding a code that should have been mapped causes silent data loss – the ETL drops every row with that code, and no error message appears. For Tier 2 (VISIBLE) vocabularies, the row survives with `concept_id = 0`, and DQD detects the gap. The cost asymmetry is stark: a false negative (noise code reaches human review) wastes thirty seconds of a reviewer's time; a false positive in a Tier 1 vocabulary silently loses data. We wrote conservative patterns for Tier 1 vocabularies and more aggressive patterns everywhere else.

Pipeline position. Tier 0 runs as Step 2b – after source discovery (Step 2) and before AI candidate generation (Step 3). This is deliberate. Every code excluded at Tier 0 is a code the LLM never evaluates. At our scale, this saved thousands of LLM calls.

Implementation guidance. Run source discovery first. Look at the discovered codes sorted by description within each vocabulary. The noise is usually obvious: formatting strings, system artifacts, retired templates, environmental samples, billing workflow records. Write patterns that capture the category of noise, not individual codes. Start conservative – you can always add patterns later, but a false exclusion in a critical vocabulary is expensive to detect. The pattern seed is a living document. Every time a reviewer encounters a code and thinks “this is obviously noise,” a new pattern is warranted.

Result: 46 patterns across 5 vocabulary domains cleared 20,201 codes as NOT_APPLICABLE – 66% of all decisions, resolved with zero human effort. The community-facing version of the pipeline ships with 17 starter patterns across 3 domains.

Tier 1: Deterministic SQL Matching

For codes that survive auto-disposition, deterministic matching resolves what can be mapped with certainty using only the OMOP vocabulary tables – no AI, no human review, just SQL joins. Two strategies execute in sequence within a single dbt model (`stcm_ai_candidates`).

Exact vocabulary match (`EXACT_VOCAB`). The source code description is compared case-insensitively against `CONCEPT.CONCEPT_NAME` . If it matches a valid, non-deprecated standard concept within the vocabulary's allowed OMOP domain, the match is recorded at confidence 1.0. For the Unit domain, the pipeline also matches against `CONCEPT.CONCEPT_CODE` (the UCUM code), because source systems frequently store UCUM abbreviations like `MG/DL` rather than the full concept name `milligram per deciliter` .

Exact matches can be bulk-approved without individual human review. This is safe because the domain filter prevents cross-domain name collisions – a source code “Oral” in a medication route vocabulary cannot accidentally match an Observation domain concept for oral examination.

Domain filtering. A seed file (`stcm_domain_map.csv` , 41 entries) maps each vocabulary suffix to its allowed OMOP domain. UNIT maps to Unit. VISIT maps to Visit. ROUTE maps to Route. Without this constraint, the 7.6-million-concept OMOP vocabulary produces false matches from unrelated domains. If no domain constraint is defined for a vocabulary, matching is unrestricted – new vocabularies still get matched, just without guardrails.

Unit normalization (institution-specific). Measurement units are the hardest Tier 1 domain because the same unit has dozens of encoding variants. `mEq/L` , `MEQ/L` , `meq/l` , `MILLIEQUIVALENT PER LITER` – all one unit, all written differently. Our production pipeline includes a 436-line `normalize_unit()` macro that maps known variants to UCUM concept codes before matching. It recognizes 48+ variants of `mL/min/1.73m2` (GFR units), for example, and collapses them to a single UCUM code. The normalizer is idempotent: it can serve both the STCM pipeline and the CDM ETL without conflict. The community pipeline does not include this macro because encoding variants are institution-dependent, but the architecture supports plugging one in.

Synonym match (`SYNONYM`). Codes that fail exact match are compared against `CONCEPT_SYNONYM` using Snowflake's `JAROWINKLER_SIMILARITY` function with a threshold of 92 (out of 100). Jaro-Winkler weights the beginning of strings more heavily, which suits medical terminology where prefixes carry clinical meaning. The threshold of 0.92 balances coverage against precision: below 0.90, false matches increase sharply; above 0.95, legitimate word-reorderings (`GLUCOSE FASTING` vs. `FASTING GLUCOSE`) and minor variants drop below the threshold.

Synonym matches receive a confidence score of `similarity / 100` . Unlike exact matches, they are routed to human review for confirmation rather than auto-approval. The match is probably right, but “probably” is not good enough for production STCM. The best synonym match per code is kept; ties are broken deterministically.

What Tier 1 catches. Demographics (“Male”, “Female”, “White”, “Hispanic”), visit types (“Inpatient”, “Outpatient”), standard medication routes (“Intravenous”, “Oral”, “Subcutaneous”), provider specialties,

and measurement units where the source system stores UCUM codes or standard names. Synonym matching extends coverage to word-reordered variants that exact match misses.

What Tier 1 misses. Abbreviated codes where the abbreviation is not a known synonym. Institutional shorthand (“GCS Total” vs. OMOP’s “Glasgow Coma Scale total score”). Numeric IDs (a flowsheet template ID like “301070” has no textual similarity to any concept). Context-dependent codes (“Rate” could be respiratory rate, heart rate, or infusion rate). These all fall through to Tier 2.

Tier 2: LLM-Assisted Candidate Generation

For codes unresolved by Tier 1, a dbt run-operation macro (`stcm_generate_candidates`) submits each code to Snowflake Cortex COMPLETE (`llama3.1-70b`) with a vocabulary-aware prompt. The macro runs as a single SQL INSERT ... SELECT statement – Snowflake parallelizes the LLM calls across all qualifying codes in one query execution.

Vocabulary-aware prompting. The original prompt asked the LLM to “suggest the most appropriate OMOP concept.” This produced poor results for specialized domains – wrong OMOP domain, wrong concept class, wrong target vocabulary. The revised prompt includes four components:

1. **Role context.** The LLM is told it is an OMOP vocabulary mapping expert.
2. **Vocabulary context (from seed).** A `stcm_vocabulary_context.csv` seed provides, per vocabulary: which CDM table and field the vocabulary populates, the target vocabulary (UCUM, SNOMED, LOINC, etc.), and valid concept classes. This narrows the search space from 7.6 million concepts to the few thousand that are valid targets for that specific vocabulary.
3. **Few-shot examples (from approved mappings).** The macro queries up to 5 recently approved mappings from the same vocabulary and includes them in the prompt. These teach the LLM both the expected output format and the mapping conventions for the domain.
4. **The source code itself.** The vocabulary ID, source code, and description.

The difference between “find any OMOP concept for this note classification” and “find a LOINC Document Ontology concept (valid classes: Doc Kind, Doc Type of Service, Doc Subject Matter, Doc Role, Doc Setting) for this note classification” is the difference between failure and success on codes that stumped the generic prompt.

What the LLM does not see. The current implementation operates entirely on the source code description plus vocabulary metadata. It does not see co-occurring diagnosis codes, department distributions, value distributions, or encounter types. The whitepaper’s earlier versions described these as Tier 2 inputs; in the current implementation, the vocabulary-aware prompt with few-shot examples resolved the majority of codes without source data profiling. Adding encounter-level context remains a viable enhancement for the residual ambiguous cases.

Minimum frequency filter. Codes with fewer than 10 occurrences are skipped unless they were previously REJECTED and re-entered the queue. A code that appears 3 times in 7.4 million patients is unlikely to justify the LLM call.

Confidence stratification. The LLM’s self-reported confidence drives the review workflow:

- HIGH (>0.9): can be bulk-approved after spot-check
- MEDIUM (0.6-0.9): individual human review required
- LOW (<0.6): escalated for domain expert

The confidence score is directionally useful for prioritization but not calibrated – a 0.8 does not mean 80% chance of correctness. Treat it as a ranking signal, not a probability.

The macro is idempotent. Running it again processes only codes that still have no candidate. It can be run incrementally as new codes are discovered or as REJECTED codes re-enter the queue.

Tier 3: Clinical Note Context Extraction

For codes where description and vocabulary context are insufficient, the design calls for querying clinical notes from encounters where the unmapped code appears. The disambiguation signal exists in the source data: a code labeled “Rate” appearing in ICU encounters alongside oxygen saturation readings and ventilator settings is obviously respiratory rate. Current tools do not access this signal. Tier 3 would.

This tier is not implemented. The vocabulary-aware prompt with few-shot examples resolved more codes than anticipated, leaving a smaller residual set of truly ambiguous codes than expected. For that residual set, human reviewers with clinical domain knowledge resolve them in a review session. Building automated note extraction infrastructure for a few dozen codes per vocabulary is engineering effort that exceeds the manual review cost at our current scale. The pipeline architecture supports adding this tier – the review queue already flags codes with no candidate, and the vocabulary context seed stores the metadata a Tier 3 prompt would need. The implementation awaits a use case that justifies the investment.

Impact Tier Classification

Every vocabulary domain is classified by what happens to the CDM when a mapping is missing:

Tier	Name	Failure Mode	Review Gate
1	CRITICAL	Silent row loss (ETL INNER JOIN on STCM)	Must be fully mapped before build
2	VISIBLE	concept_id = 0, row survives	Can iterate after build; DQD detects
3	CORRECTNESS	Mapped to wrong concept	Requires clinical expert review
4	EXTENDED	Non-standard STCM usage	Needs documentation for community

The tier classification is stored as a CSV seed (`stcm_impact_tiers.csv` , 69 entries) and joined into the review queue model. The review app sorts by tier, so critical-tier codes surface before visible-tier codes regardless of frequency. A build gate flags any critical-tier vocabulary with zero STCM entries.

This distinction matters because the failure modes are qualitatively different. A missing visit classification STCM entry silently excludes an entire encounter from the CDM. A missing unit STCM entry produces a row with `unit_concept_id = 0` — the measurement survives and the gap is detectable. Treating both as equally urgent wastes reviewer time on visible gaps while critical gaps persist.

Human Review Interface

A Streamlit application (5 tabs) replaces the Usagi desktop experience:

Dashboard. Decision summary metrics by vocabulary. Bar chart of coverage. Impact tier coverage section showing decided/pending counts per tier, with critical-tier gaps highlighted. Progress tracking across vocabulary domains.

Review Queue. Selectable dataframe with filters (impact tier, vocabulary, match method, confidence status). Default sort: tier ascending, then frequency descending — critical-tier codes surface first. For each code: the source description, AI candidate suggestion, confidence score, and match method. The reviewer approves, rejects, marks as not-applicable, or searches for an alternative concept.

Bulk Actions. Approve all EXACT_VOCAB matches at 1.0 confidence. Mark all NO_CANDIDATE codes as APPROVED_ZERO (custom concept). Reject low-confidence SYNONYM matches below a threshold. These bulk actions clear hundreds of codes in a single click for vocabularies where the pattern is uniform.

Direct Editor. Inline editing of the mapping decisions table for corrections or manual entries.

History. Audit trail with date, vocabulary, and decision filters.

Output Formats

The pipeline produces two output types:

SOURCE_TO_CONCEPT_MAP entries. Standard OHDSI format. For codes with adequate standard concept matches. These enter the CDM's vocabulary mapping and are used by the ETL during the next build.

Custom concept entries (CONCEPT_ID >= 2,000,000,000). For codes with no standard equivalent. The pipeline allocates IDs via a Snowflake sequence (SEQ_2B), creates full CONCEPT table entries (with CONCEPT_NAME, DOMAIN_ID, VOCABULARY_ID, CONCEPT_CLASS_ID), and generates CONCEPT_RELATIONSHIP entries connecting the custom concept to its nearest standard ancestor. A stored procedure (STCM_to_CCR_macro2) handles the propagation idempotently — it inserts new concepts, updates changed ones, and removes stale ones.

Architecture

The entire pipeline runs within a single dbt project (care-chri-omop-maintenance) on Snowflake:

```
Step 1: dbt build --select vocabularies           (import Athena vocabs)
Step 2: dbt build --select stcm_pipeline         (discover + Tier 1 + Tier 2)
Step 2b: dbt run-operation stcm_auto_disposition (pattern rules)
Step 3: dbt run-operation stcm_generate_candidates (Tier 3 AI, vocabulary-aware)
Step 4: streamlit run stcm_review_app.py        (human review)
Step 5a: dbt build --select stcm_load           (approved → SOURCE_TO_CONCEPT_MAP)
Step 5b: dbt run-operation STCM_to_CCR_macro2   (approved → CONCEPT + CONCEPT_RELATIONSHIP)
```

Source Discovery Models

A set of dbt models in `models/stcm_pipeline/sources/` — one per source vocabulary — query the CDM source tables and extract unique source codes with their frequency and descriptive metadata. The models share a 5-column contract: `source_code` , `source_vocabulary_id` , `source_code_description` , `source_frequency` , `additional_info` .

A master model (`STCM_SOURCE_CODES`) unions all source models into a single discovery table. This table is the pipeline’s input.

Decision Tracking

`STCM_MAPPING_DECISIONS` stores every decision: the source code, vocabulary, the decision (APPROVED, APPROVED_ZERO, NOT_APPLICABLE, REJECTED), the target concept if approved, the match method, confidence score, and a timestamp. This table is the single source of truth. The Streamlit app reads from and writes to it. The dbt load step reads approved decisions from it.

Seed Files

Seed	Rows	Purpose
stcm_domain_map.csv	37	Vocabulary-suffix to OMOP domain routing
stcm_not_applicable_patterns.csv	46	Auto-disposition pattern rules
stcm_vocabulary_context.csv	16	Tier 2/3 prompt context (CDM field, target vocab, valid concept classes)
stcm_starter_mappings.csv	1,897	Community-portable pre-resolved mappings
stcm_procedure_name_mappings.csv	50	Procedure name to CPT4 expert mappings
stcm_impact_tiers.csv	69	Vocabulary-suffix to impact tier classification
stcm_flowsheet_categories.csv	158	Flowsheet source code to clinical category assignment

Results

Resolution Summary (September 2026)

Category	Count	% of Discovered
APPROVED (mapped to standard concept)	7,252	46.5%
APPROVED_ZERO (custom 2B+ concept)	770	4.9%
NOT_APPLICABLE (non-clinical / ETL-handled)	21,412	
EXCLUDE (non-clinical, confirmed)	7	
Total decided	29,442	
Remaining undecided	0	0%

Total decisions exceed discovered codes because some decisions were made on codes later removed from the discovery pipeline during schema consolidation.

Production Output

Metric	Value
Source codes discovered	12,781
Vocabulary domains	39
Resolution rate	100%
STCM rows in production	8,022
Custom 2B+ concepts	6,127
Starter seed mappings (portable)	1,897
Impact tier classifications	69
Flowsheet category assignments	158

The 39 vocabulary domains include seven flowsheet categories created by decomposing the original catch-all “miscellaneous” category. The pipeline also produces shared demographic and unit mappings that serve the CDM alongside the vocabulary-specific entries above.

Flowsheet Category Split

The original ETL routed all non-standard flowsheet measurements through a single MISC category. This catch-all contained 3.5 billion measurement rows spanning ventilator parameters, clinical scoring instruments, behavioral health screenings, oxygenation monitoring, fall risk assessments, anesthesia gas tracking, and intake/output measurements. The label “MISC” communicated nothing to implementers, researchers, or downstream consumers about what these measurements actually represented.

Classification of the 158 source codes in the MISC STCM produced nine clinically coherent categories:

Category	STCM Entries	Target Vocabulary	Clinical Content
OXYGENATION	42	OEE_FLWSHT_MEAS_OXYGENATION	SvO2, ScvO2, PaO2, blood gas saturations, pulse oximetry
BEHAVIORAL	34	OEE_FLWSHT_MEAS_BEHAVIORAL	PHQ-2/4/9, GAD-7, AUDIT-C, Edinburgh, C-SSRS, Beck, Mini-Cog
SCORING	28	OEE_FLWSHT_MEAS_SCORING	GCS, SOFA, MEWS, APACHE, Aldrete, PELOD, sepsis predictive
VENTILATOR	28	OEE_FLWSHT_MEAS_VENTILATOR	PEEP, tidal volume, FiO2, airway pressure, minute ventilation
VITALS (overflow)	12	OEE_FLWSHT_VITALS	Pain scores, MAP — reclassified to existing vitals vocabulary
ANES_GAS	7	OEE_FLWSHT_MEAS_ANES_GAS	Sevoflurane, nitrous oxide, inspired/expired agent concentrations
FALL_RISK	4	OEE_FLWSHT_MEAS_FALL_RISK	Fall risk predictive model scores, Morse scale
IO	2	OEE_FLWSHT_MEAS_IO	Urine output, drain volumes
TRUE MISC	1	OEE_FLWSHT_MEAS_MISC	Tube feeding documentation (genuinely uncategorizable)

Each category has its own vocabulary_map and STAGE models across three clinical settings (ambulatory, inpatient, anesthesia). The ETL_MODULE column in the CDM now reflects the clinical category: a row from STAGE_MEASUREMENT_HOSP_FLOWSHEET_SCORING is immediately identifiable as a clinical score, whereas a row from STAGE_MEASUREMENT_HOSP_FLOWSHEET_MISC previously required examining the source value to determine what it measured.

The residual MISC category contains exactly one source code. A category that started at 3.5 billion rows and ended at one code is evidence that “miscellaneous” was never a category — it was an admission that the categorization had not been done.

Decisions by Vocabulary (Top 10)

Vocabulary	Approved	Custom	N/A	Total
OEE_ALLERGY	0	0	9,104	9,104
OEE_UNIT	1,874	0	6,695	8,569
OEE_DEVICE_SURG_IMPLANT	2,983	0	0	2,983
OEE_POS	8	0	1,814	1,822
OEE_ANATOMIC_SITE	693	17	60	770
OEE_SPECIALTY	392	0	214	606
OEE_FLOWSHEET_HT_WT	0	0	579	579
OEE_VALUE_AS_CONCEPT	392	22	155	569
OEE_FLOWSHEET_UNIT	156	49	281	486

The largest single vocabulary (OEE_ALLERGY at 9,104 codes) is entirely NOT_APPLICABLE: allergy names are stored as source_value text, not concept-mapped. Discovering this early saved months of mapping effort on codes that would never need mapping.

OEE_DEVICE_SURG_IMPLANT (2,983 codes) is the largest APPROVED vocabulary — surgical implant identifiers mapped to SNOMED device concepts. This vocabulary was built as part of a separate surgical device expansion project and integrated into the STCM pipeline.

Resolution Completeness

All 12,781 discovered source codes have decisions. The pipeline reached 100% resolution through iterative application of the four tiers plus systematic domain expert review. The remaining work is maintenance: new source codes that appear as the source system evolves are detected by the discovery models and queued automatically.

Community Contribution: The Starter Seed

1,897 pre-resolved APPROVED mappings are exported as a CSV seed. These cover source codes that are universal across Epic implementations:

- **Unit strings** (MG/DL, MEQ/L, /HPF, MM HG): identical text across all Epic sites
- **UB-04 admit descriptions** (Emergency Room, Hospice, SNF): federal standard, same everywhere
- **Visit type classifications** (Office Visit, Telehealth, Home Visit): standard Epic encounter types
- **Provider specialty codes** (Family Medicine, Cardiology): standard Epic specialty list

Portability boundary: Text-based source codes (unit strings, description text) are portable. Numeric ID-based source codes (Epic internal IDs for flowsheet templates, note types, LDA rows) are not — these are

instance-specific.

A new site running this pipeline for the first time loads the starter seed via `dbt run-operation stcm_load_starter_mappings`. The operation is idempotent. Approximately 1,900 codes resolve immediately before the site discovers its first local mapping gap.

What Usagi Cannot Do

This is not a criticism of Usagi. It is a statement of architectural limitations that any term-similarity tool shares:

No source context. Usagi sees a code description. It does not see what diagnoses co-occur, what department the code appears in, what values it takes, or what the clinician documented when the code was used. Our pipeline's architecture supports querying all of this; the current implementation uses vocabulary-domain context and few-shot examples from approved mappings, with encounter-level profiling as a planned enhancement.

No demand prioritization. Usagi processes codes in whatever order they appear. It has no concept of which codes matter most for research. Our pipeline prioritizes by historical research project frequency — derived from a scan of 315 completed honest broker projects.

No vocabulary awareness. Usagi searches the entire 7.6-million-concept vocabulary for every code. Our pipeline restricts candidates to the expected OMOP domain and valid concept classes for the specific CDM field the vocabulary populates. This eliminates false matches from unrelated domains.

No automation. Usagi is a manual, code-by-code desktop workflow. Our pipeline auto-disposes 66% of codes via pattern rules, auto-approves deterministic matches, and bulk-processes high-confidence AI candidates. The human reviewer focuses on the genuinely ambiguous 2%.

No feedback loop. Usagi does not know when the CDM encounters a new unmapped code. Our pipeline's source discovery models run with every dbt build and automatically detect new codes.

Generalizable Findings

1. Most source codes do not need mapping

66% of our discovered codes were NOT_APPLICABLE — non-clinical, ETL-handled, or workflow artifacts. Discovering this early prevents months of wasted mapping effort. Auto-disposition rules should be the first step in any STCM pipeline, before matching algorithms are invoked.

2. Source data context disambiguates what term similarity cannot

A code labeled "Rate" has zero useful similarity to any OMOP concept. The same code with a respiratory department distribution, oxygen saturation co-occurrences, and ICU encounter type is unambiguously

respiratory rate. This context exists at every OMOP site. It is free to query. No one queries it during mapping.

3. Demand-driven prioritization focuses effort where it matters

Of 12,781 codes, the ~500 appearing in historical research projects cover 90%+ of research utility. A scan of completed research queries provides the prioritization signal. Sites with honest broker archives already have this data.

4. The pipeline is incrementally adoptable

Tier 0 (auto-disposition) requires only pattern rules. Tier 1 (deterministic SQL) requires only the OMOP vocabulary tables. Tier 2 (LLM) requires an LLM with access to vocabulary metadata and a few approved examples. Tier 3 (note extraction, not yet implemented) would require clinical note access. Each tier adds value independently. A site that implements only Tier 0+1 resolves 70%+ of codes without any AI involvement.

5. Catch-all categories hide clinical specificity

Our “miscellaneous” flowsheet category contained 3.5 billion rows spanning ventilator parameters, depression screenings, GCS scores, and oxygen saturation measurements. Decomposing it into clinically meaningful categories required classifying only 158 source codes — the STCM entries that route flowsheet data into the CDM. The effort was modest. The information gain was substantial: researchers can now query by clinical category, curation checks can set different thresholds per category, and the review app can prioritize by clinical impact tier rather than treating all flowsheet codes as equivalent.

Any OMOP site with a “miscellaneous” or “other” STCM vocabulary should examine what it contains. The catch-all is almost certainly hiding clinically distinct subgroups that matter to different research consumers.

6. Impact tiers focus reviewer effort on what breaks the build

Classifying vocabulary domains by failure severity (silent data loss vs. visible `concept_id = 0` vs. incorrect mapping vs. non-standard usage) changed review behavior. Before tiers, the review queue sorted by frequency — high-volume codes first. After tiers, a critical-tier code with 50 occurrences surfaces before a visible-tier code with 50,000 occurrences, because the critical-tier gap silently excludes data while the visible-tier gap produces a detectable `concept_id = 0`. The tier framework is a CSV seed (69 entries) that any OMOP site can adapt to its own vocabulary inventory.

7. Custom concepts fill a real gap

When no standard concept exists, the choice is `concept_id = 0` (data present but invisible to queries) or a custom 2B+ concept (data queryable via local vocabulary). We created 6,127 custom concepts. Every one of them represents source data that would otherwise be invisible despite being in the CDM.

Post-Pipeline Evaluation

After achieving 100% resolution, we evaluated the approved mappings using a three-layer audit.

Layer 1: Automated Structural Checks

Seven SQL-based checks ran against all 8,022 approved mappings:

Check	Failures
Deprecated target concept (INVALID_REASON set)	3
OMOP domain mismatch	0
Duplicate source codes	0 (functional)
APPROVED with concept_id = 0 (should be APPROVED_ZERO)	30
Orphaned APPROVED_ZERO (no custom concept)	0
Non-standard target concept	12
Missing target concept	0

These are binary pass/fail checks. Every failure is a definite error.

Layer 2: LLM Semantic Evaluation

4,459 mappings were evaluated using Snowflake Cortex (llama3.1-70b) as an LLM judge. The evaluation was stratified by risk:

- **All 387 critical-tier mappings** (vocabularies where missing STCM causes silent data loss)
- **All 3,701 low-trust-method mappings** (CSV_IMPORT, BULK_TRIAGE, STARTER_SEED, and records with no match method)
- **10% random sample of 3,934 high-trust-method mappings** (371 evaluated)

For each mapping, the LLM received the source code, target concept metadata, CDM field context, and valid concept classes, then rated the mapping as CORRECT, PLAUSIBLE, INCORRECT, or INSUFFICIENT_INFO.

Stratum	Evaluated	CORRECT	PLAUSIBLE	INCORRECT
Critical (Tier 1)	387	198 (51%)	149 (39%)	40 (10%)
Low-trust method	3,701	2,008 (54%)	889 (24%)	797 (22%)
High-trust sample	371	123 (33%)	202 (54%)	43 (12%)
Total	4,459	2,329	1,240	880

Layer 3: Human Triage of LLM-Flagged Mappings

The 880 INCORRECT ratings were triaged into three categories:

- **False positives (~530, 60%):** The LLM applied strict UCUM standards to legitimate OMOP unit concepts, misunderstood APPROVED_ZERO semantics, and narrowly interpreted value_as_concept scope. The mappings were correct.
- **Definite errors (~180, 20%):** Deprecated target concepts, non-standard concepts without standard equivalents, non-clinical content in clinical vocabularies, wrong concept classes, and wrong semantic mappings.
- **Debatable (~170, 19%):** Edge cases where OMOP's vocabulary lacks a precise concept for the institutional source code (e.g., UB-04 admit source codes for newborns).

121 corrections were applied: 30 decision-type corrections (APPROVED to APPROVED_ZERO), 11 concept replacements (deprecated or non-standard targets updated to current standard concepts), 14 reclassifications (non-clinical content removed from clinical vocabularies), 1 semantic fix, and 65 specialty mappings rejected for re-evaluation through the pipeline.

The evaluation confirmed that the pipeline's high-trust match methods (deterministic SQL, expert review) have low error rates. Legacy CSV imports are the primary source of mapping errors, consistent with the pipeline's design premise: inherited mappings of unknown provenance should be validated, not trusted.

Limitations

- Single institution, Epic source system. Non-Epic sites have different local code structures and vocabulary domains.
 - Tier 3 (clinical note context extraction) is not implemented. Results reported are from Tiers 0-2. The vocabulary-aware prompt resolved more codes than anticipated, reducing the need for note-level disambiguation.
 - LLM-generated candidates require human validation. The pipeline accelerates review but does not eliminate the need for domain expertise.
 - Custom 2B+ concepts are institutional extensions, not contributions to the standard OHDSI vocabulary. They enable local querying but do not improve cross-network comparability.
 - The 100% resolution rate includes NOT_APPLICABLE dispositions. If measured as "codes with a positive standard concept mapping," the rate is 62.8% (8,022 of 12,781). The remaining codes are correctly classified as non-clinical — they genuinely do not need mapping — but the headline number should be understood in this context.
 - The starter seed is validated at one Epic site. Portability to other Epic configurations is assumed but not empirically confirmed.
-

References

1. Observational Health Data Sciences and Informatics. Usagi: Large-scale Vocabulary Mapping. ohdsi.github.io/Usagi/
2. Snowflake Inc. Snowflake Cortex LLM Functions. 2024. docs.snowflake.com/en/user-guide/snowflake-cortex/llm-functions
3. Snowflake Inc. AI_EXTRACT Function Reference. 2024. docs.snowflake.com/en/sql-reference/functions/ai_extract
4. Observational Health Data Sciences and Informatics. The Book of OHDSI. Chapter 6: ETL — Source to Concept Map. 2024. ohdsi.github.io/TheBookOfOhdsi/
5. Philofsky A. Custom OMOP Concepts: STCM vs Concept/Concept_Relationship. OHDSI Forums. 2020.
6. Carlson R, Phad A. The 2-Billionaire Pattern: Custom Concept Creation for Institutional STCM. Corewell Health internal documentation. 2024.

Roger Carlson | Corewell Health | SASS Honest Broker Analytics