AI-Enhanced DEMA: Transforming Implicit System Knowledge into Intelligent, Compliant, and Documented Processes

Dan O'Leary, Ph.D.

Allison B. Ledford, Ph.D.

This research is funded by the Interdisciplinary Center for Advanced Manufacturing Systems (ICAMS) through the Industrial Base Analysis and Sustainment Office of the Office of the Secretary of Defense.

Presentation Overview

- Background
- Introduction to DEMA
- Need for AI-Enhanced DEMA
- Reconciliation Approach
- Proof of Concept Results
- Benefits, Limitations, and Future Work

Introduction to Digitalization

- The terms “digitization” and “digitalization” are often confused with one another.
  - Digitization is the computerization of manual activities. [1]
  - Digitalization is the fundamental restructuring of an existing process to improve connectivity and information flows while taking advantage of digital capabilities. [1]

“IF YOU ADD DIGITAL,
ON TOP OF A THING THAT IS BROKEN,
YOU WILL HAVE A BROKEN DIGITAL THING.”
ORG OPEN RIGHTS GROUP

Figure 1: Graphic from Open Rights Group [2]

Defining the Data

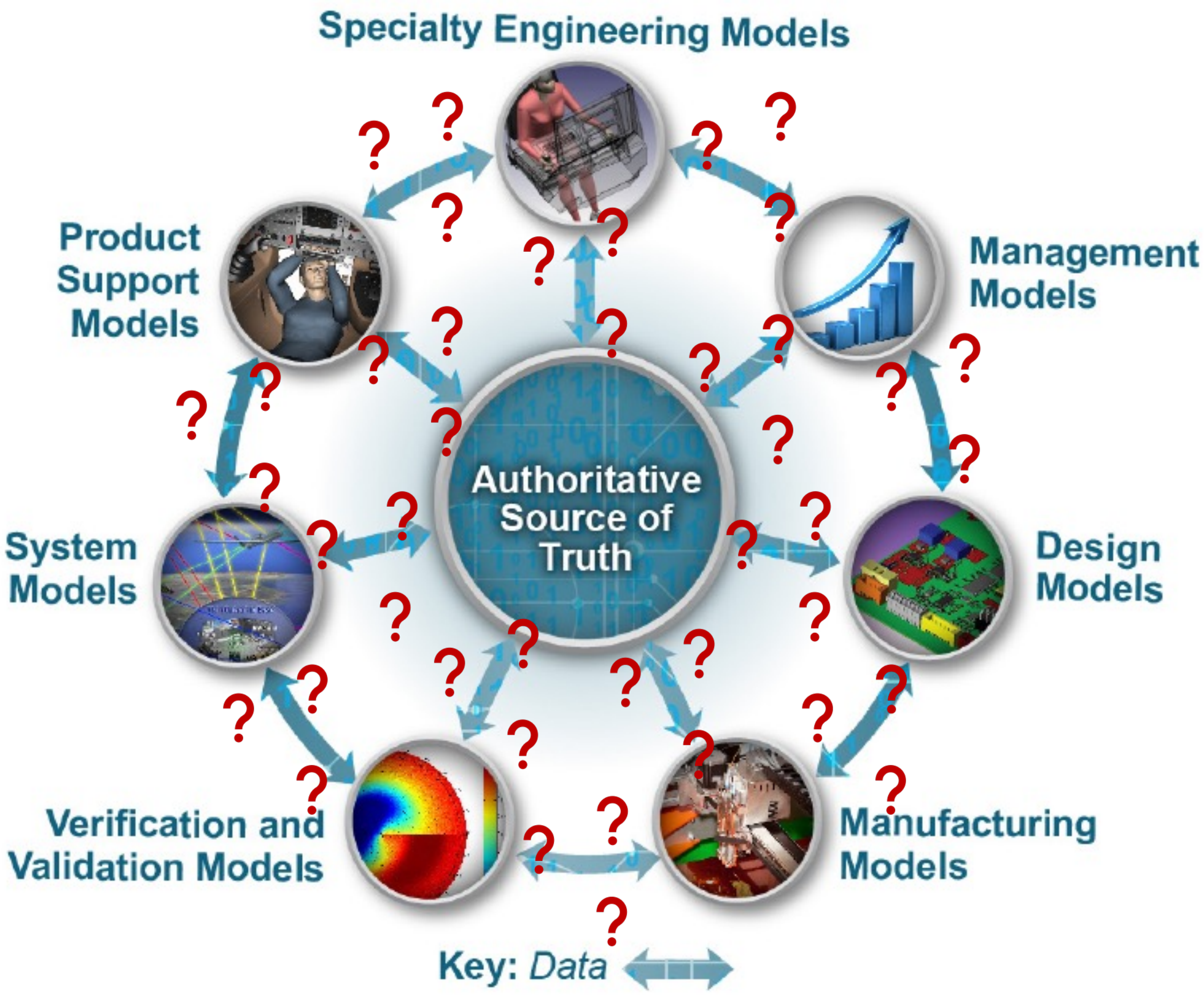


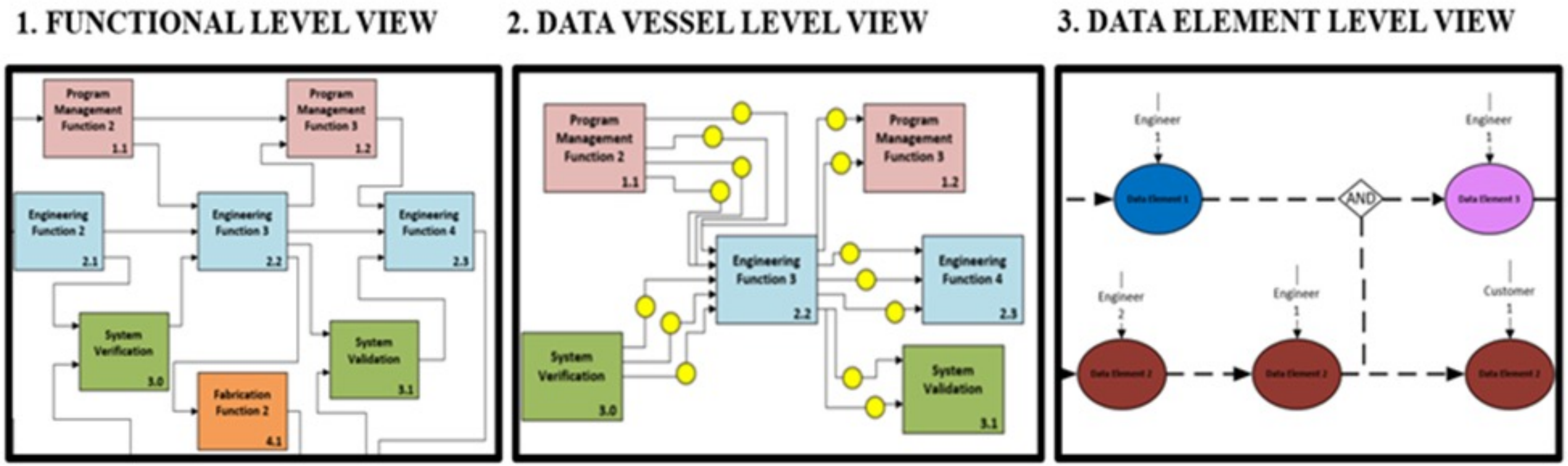
Figure 2: Adapted from Authoritative Source of Truth Figure from 2018 DoD Digital Engineering Strategy [6]

From our research [3, 4, 5]:

- Over 90% of data handling is unknown and nonstandard (hidden to the organization).
- Greater than 50% of data vessel inputs and outputs are unstructured.
- Greater than 90% of data element exchanges are manual.

DEMA Methodology Overview

- Data Element Mapping and Analysis (DEMA):
  - Combines traditional functional analysis, systems engineering elicitation, and novel data mapping techniques to provide a wholistic view of a system’s data and information flows down to the data element level.
  - DEMA itself is not software, it is a 3-step approach to be used with process mapping software and tables.



1. FUNCTIONAL LEVEL VIEW

2. DATA VESSEL LEVEL VIEW

3. DATA ELEMENT LEVEL VIEW

Program Management Function 2 1.1
Program Management Function 3 1.2
Engineering Function 2 2.1
Engineering Function 3 2.2
Engineering Function 4 2.3
System Verification 3.0
System Validation 3.1
Fabrication Function 2 4.1
Program Management Function 2 1.1
Program Management Function 3 1.2
Engineering Function 3 2.2
Engineering Function 4 2.3
System Verification 3.0
System Validation 3.1
Engineer 1
Engineer 1
Data Element 1
AND
Data Element 3
Engineer 2
Engineer 1
Customer 1
Data Element 2
Data Element 2
Data Element 2

DEMA Terminology

Term	Definition
Functional Area	The highest levels by which the functional activities in the system can be grouped.
Sub-Functional Area	Sub-groupings of the functional activities within the functional areas.
Functional Activity	The activities in the system that transform data vessel inputs into outputs.
Data Vessel	The documents, emails, personal notes, drawings, CAD files, and any other possible container (i.e., vessel) of data.
Data Element	The individual pieces of data contained within data vessels such as document titles, dimensions, software file inputs, individual requirements, and due dates.

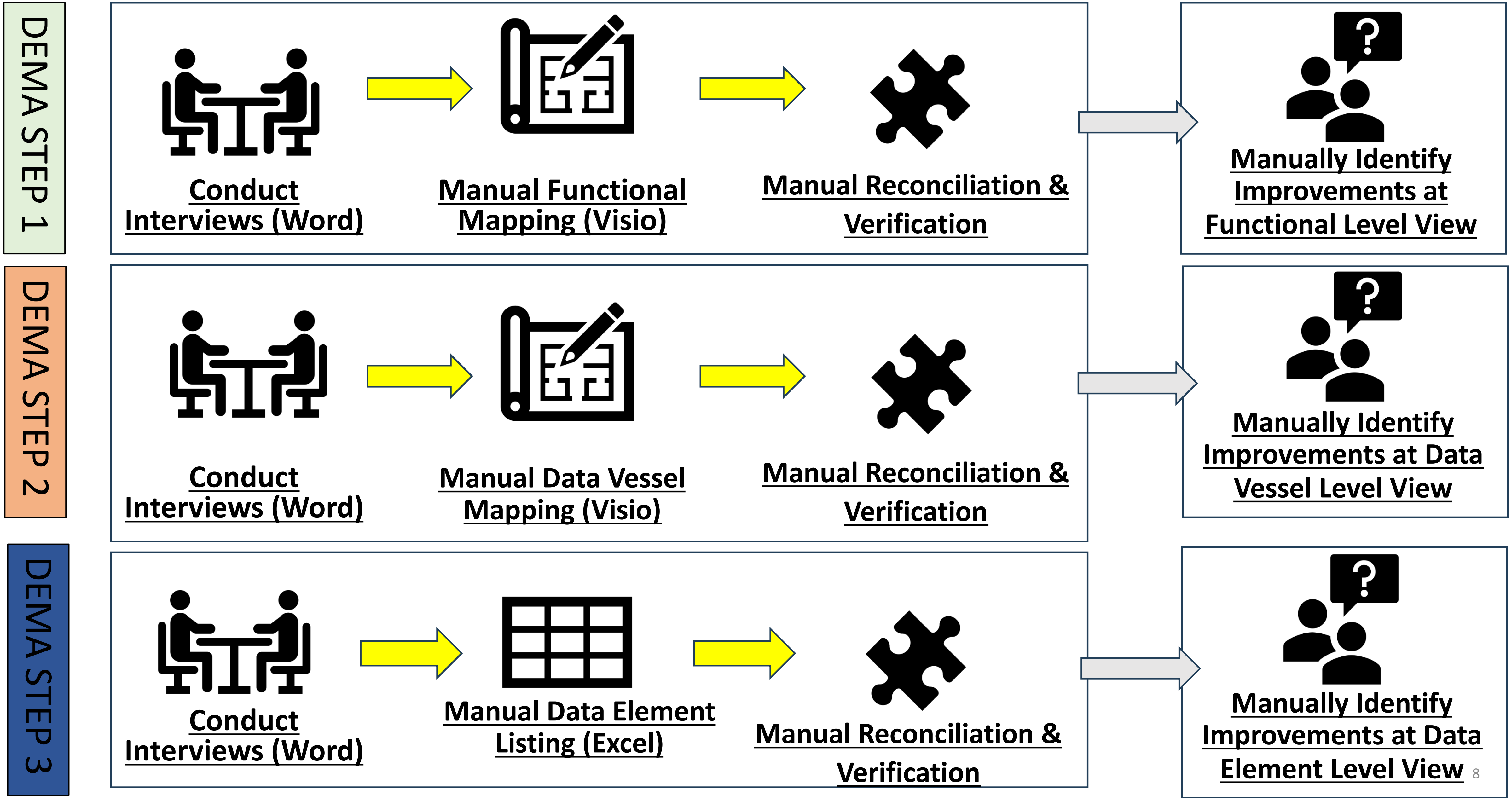
DEMA In Real Life

- DEMA Applied to Prototyping System:
  - Step 1: Six functional areas and 67 functional activities.
  - Step 2: Around 1000 data vessel inputs and outputs.
  - Step 3: Around 2,500 unique data elements and around 25,000 data element instances.
- These results were used to begin connecting the Digital Thread in one of the six functional areas of the system.
- Key data threads were identified and an improved data architecture for the engineering functional area was created.
  - 25% of the data operations were moved from manual to automated, beginning the connection of the Digital Thread.
  - Data element handling reduced by 22%, reducing workload and opportunities for quality errors.
  - A conservative estimate of the labor associated with data handling was reduced 888 hrs. to 661 hrs.
    - With a fully loaded rate of $100 an hr., this would result in $22,227 savings for the effort, with 227 manpower hrs. freed to be applied to other efforts.

***If the organization does 100 efforts a year, this will equate to more than $2 million in savings and the elimination of 11-man years of effort per year.**

AUBURN UNIVERSITY | SAMUEL GINN COLLEGE OF ENGINEERING

DEMA Without AI



DEMA STEP 1

Conduct Interviews (Word) → Manual Functional Mapping (Visio) → Manual Reconciliation & Verification → Manually Identify Improvements at Functional Level View

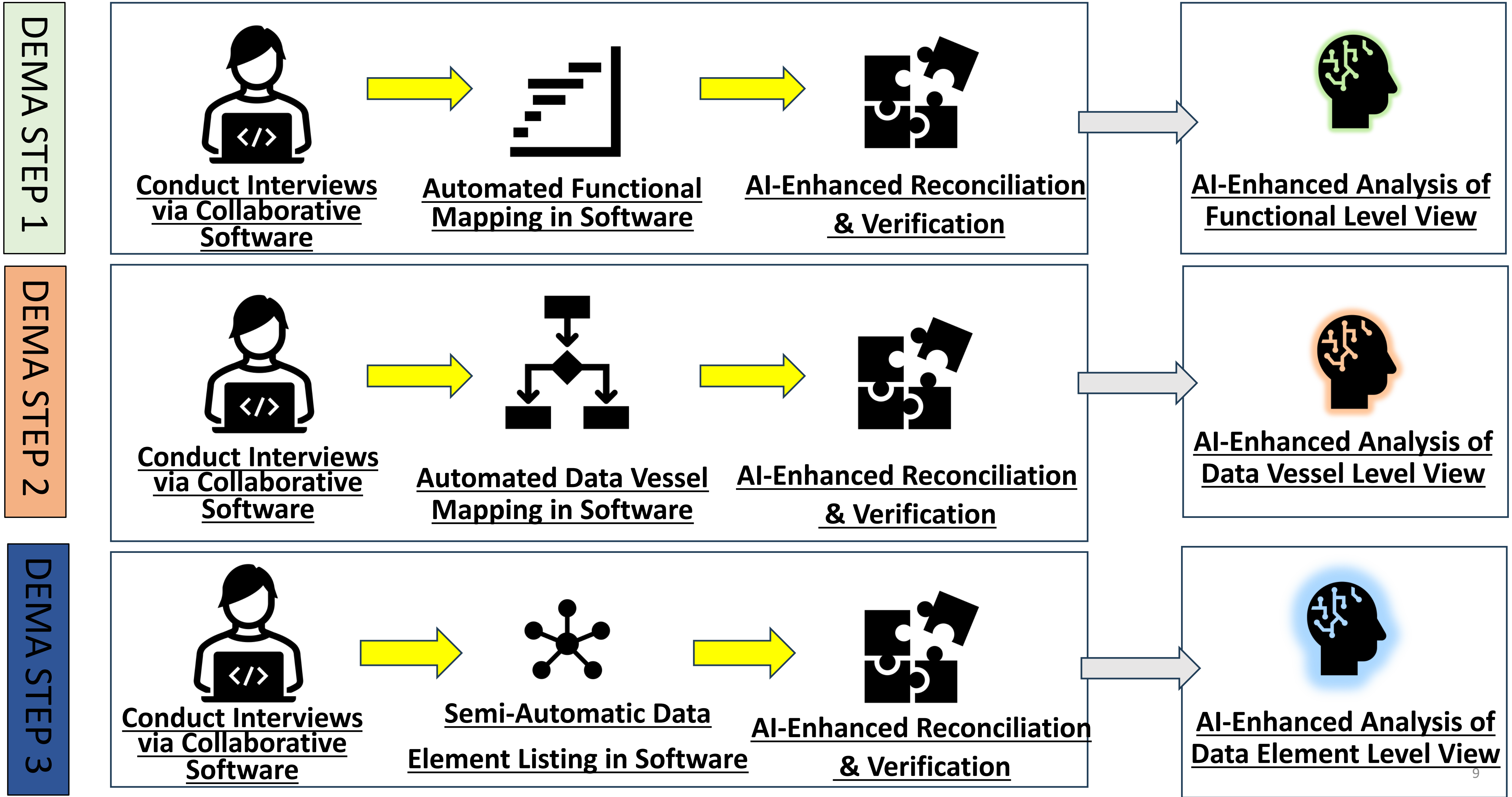
DEMA STEP 2

Conduct Interviews (Word) → Manual Data Vessel Mapping (Visio) → Manual Reconciliation & Verification → Manually Identify Improvements at Data Vessel Level View

DEMA STEP 3

Conduct Interviews (Word) → Manual Data Element Listing (Excel) → Manual Reconciliation & Verification → Manually Identify Improvements at Data Element Level View

DEMA With AI & Software



DEMA STEP 1

Conduct Interviews via Collaborative Software → Automated Functional Mapping in Software → AI-Enhanced Reconciliation & Verification → AI-Enhanced Analysis of Functional Level View

DEMA STEP 2

Conduct Interviews via Collaborative Software → Automated Data Vessel Mapping in Software → AI-Enhanced Reconciliation & Verification → AI-Enhanced Analysis of Data Vessel Level View

DEMA STEP 3

Conduct Interviews via Collaborative Software → Semi-Automatic Data Element Listing in Software → AI-Enhanced Reconciliation & Verification → AI-Enhanced Analysis of Data Element Level View

The Data Element Reconciliation Challenge

93%

Undocumented Data Handling

67%

Integration Failures from Unknown Data

6-8

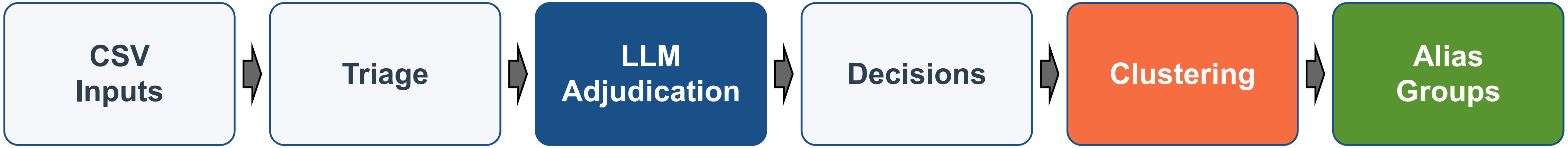
Weeks for Manual Review

Core Problem: Is "Cust_ID" (Sys A) the same as "Client Num" (Sys B)?

Traditional Approaches Fail Because:

- Rules break on semantic variations
- Little labeled training data exists
- Manual review doesn't scale

AI-Enhanced Reconciliation



CSV Inputs → Triage → LLM Adjudication → Decisions → Clustering → Alias Groups

Key Architecture Components:

- Multi-provider support (Local, Hosted)
- Two-stage processing pipeline
- Outcomes: MERGE, KEEP, ABSTAIN
- 100% citation coverage for audit trails

Technical Stack:

Python 3.12+ / Typer CLI

Pydantic schemas

LiteLLM abstraction

SQLite + JSONL storage

Provider-agnostic design

The Cross-Vessel Consideration

BEFORE

0% MERGE Recall

All cross-vessel pairs kept separate

AFTER

83% MERGE Recall

Cross-vessel equivalence recognized

KEY INSIGHT

"Vessel differences are context, not conflict"

Many true duplicates exist across different vessel types

Proof of Concept Results

Precision

80-95% ✓

Target: ≥90%

Processing

45 pairs/min ✓

Target: ≥30

Cost

$0.015/decision ✓

Target: <$0.10

Citations

100% ✓

Target: 100%

Provider	Precision	Cost/Decision	Deployment
OpenAI	95%	$0.018	Cloud only
Anthropic	90%	$0.042	Cloud only
Ollama	80%	Infrastructure	Air-gapped

Real-World Validation Example

LEFT ELEMENT

`"Proto_Name"`

From: Database System

RIGHT ELEMENT

`"Prototype Request Name"`

From: Email Template

DECISION: MERGE

Confidence: 0.91

Rationale:

"Both elements represent the prototype identifier with minor formatting differences. 'Proto_Name' in the database and 'Prototype Request Name' in the email template serve the same purpose of uniquely identifying prototype requests across systems."

Evidence Citations: `Left: Rows 3, 7 | Right: Rows 17, 22`

KEY FEATURE: Complete explainability with audit trail - not a black box!

Technical Challenges Overcome

Challenge:

Schema complexity varies by provider

→

Solution:

LiteLLM wrapper, Adaptive fallback strategies

Challenge:

Zero MERGE recall initially

→

Solution:

Re-ask mechanism for cross-vessel pairs

Challenge:

Poor confidence calibration

→

Solution:

Isotonic regression adjustment

Challenge:

$O(n^2)$ comparison space

→

Solution:

Deterministic blocking + triage

Innovation: Provider-agnostic architecture with graceful degradation

Current State & Limitations

✓ What Works

- Core reconciliation engine proven
- Multi-provider architecture validated
- Complete audit trail implemented
- Cross-vessel equivalence working
- Citation coverage at 100%
- Deterministic preprocessing effective

⚠ Known Limitations

- Local models: 80% vs 95% precision
- Scale: ~1,000 pairs/hour current limit
- Domain-specific tuning required
- Prompt optimization still needed
- More real-world data needed
- O(n^2) clustering complexity

STATUS: Ready for pilot deployments (shadow/review-assisted mode)

Implementation Roadmap

Current

Pilot Validation

- Shadow mode with 3-5 partners
- Domain-specific tuning
- Threshold optimization
- Performance metrics collection

Next

Extension & API

- Function & Vessel layers
- REST API development
- User interface creation
- Documentation & training

Future

Production Scaling

- Continuous learning pipeline
- Industry specializations
- Performance optimization
- Enterprise deployment

Value Proposition & Next Steps

Cost Reduction

100x

vs manual review

Time Savings

Weeks→Hours

reconciliation time

Auditability

100%

citation coverage

Deployment

Flexible

Cloud / on-premise / on-device

Immediate Actions Required:

1. Identify pilot partners
2. Validate with more real-world data
3. Optimize prompts with DSPy, improve observability
4. Test additional local models
5. Begin API development

Desired Enhancements:

Resources for pilot program
Production hardening
Domain specialization
User interface development

References

1. G. A. Harris, D. Abernathy, L. Lu, A. Hyre, and A. Vinel, “Bringing clarity to issues with adoption of digital manufacturing capabilities: an analysis of multiple independent studies,” Journal of the Knowledge Economy, 2021.
2. https://www.openrightsgroup.org/

3. A. B. Ledford, “A Data Element Mapping and Analysis (DEMA) Approach for Implementing a Complete Digital Thread”, Auburn University Dissertation, 2023.

4. A. B. Ledford, G. A. Harris, and G. Purdy, “Implementing a Complete Digital Thread: The Need for Data Element Mapping and Analysis,” IEEE Open Journal of Systems Engineering, 2023.

5. A. B. Ledford, G. A. Harris, S. Askew, G. Purdy, “Application of data element mapping and analysis for system definition to enable model-based systems engineering, “ INCOSE SE Journal, 2024.

6. Office of the Deputy Assistant Secretary of Defense for Systems Engineering. “Department of Defense Digital Engineering Strategy,” 2018

Appendix

A1: Actual LLM Prompt & Response Example

PROMPT TO LLM:

```
System: You are an adjudicator determining if data
elements are semantically equivalent.

Policy: Vessel differences are context, not
conflict.

Decision rubric:
• MERGE: Same field/meaning
• KEEP_SEPARATE: Different scope
• ABSTAIN: Insufficient evidence
```

LLM RESPONSE:

```
{
  "decision": "MERGE",
  "confidence": 0.91,
  "rationale": "Both elements represent...",
  "citations": {
    "left_rows": [3, 7],
    "right_rows": [17, 22]
  }
}
```

METADATA PROVIDED:

```
{
  "left": {
    "name": "Proto_Name",
    "vessel": "database",
    "function": "Engineering",
    "actors": ["Engineer", "PM"],
    "rows": [3, 7]
  },
  "right": {
    "name": "Prototype Request Name",
    "vessel": "email",
    "function": "Engineering",
    "actors": ["PM", "Customer"],
    "rows": [17, 22]
  },
  "triage": {
    "string_sim": 0.82,
    "cooccur": 3
  }
}
```

(1,247ms, 567 prompt tokens, 89 completion tokens)

A2: Evaluation Methodology

Confusion Matrix & Metrics (Merge is Positive):

		Predicted	
		MERGE	KEEP
Actual	MERGE	TP=83	FN=17
	KEEP	FP=5	TN=95

Precision
TP/(TP+FP) = 94%

Recall
TP/(TP+FN) = 83%

F1
2PR/(P+R) = 88%

Why Recall > Precision Here:

Missing duplicates (low recall):

- Hidden redundancy continues
- Integration failures persist
- Problem remains unsolved

False merges (low precision):

- Caught in review queue
- Visible and correctable
- ABSTAIN provides safety

Confidence Calibration (Isotonic Regression):

Before Calibration:

0.6 conf → 42% accurate
0.9 conf → 96% accurate

After Calibration:

0.6 conf → 60% accurate
0.9 conf → 90% accurate

Gold Standard: 40 manually labeled pairs from domain experts

A3: Failure Modes & Mitigations

Abbreviation Confusion

"Req_ID" vs "Request_Identifier"

→ Enhanced string similarity scoring

Temporal Ambiguity

"Date" fields without context

→ ABSTAIN when context insufficient

Role Variation

Same field, different actors

→ Cross-vessel gates check actors

Vessel Type Bias

Email vs System differences

→ Re-ask mechanism for high similarity

Safety Mechanisms:

1. ABSTAIN option (7-10% of decisions)
2. Confidence thresholds (configurable)
3. Human review queue (prioritized)
4. Citation requirements (100% coverage)
5. Negative edge blocking in clustering

Review Queue Prioritization:

```
def prioritize_review(decisions):
    return sorted(decisions, key=lambda d: (
        abs(d.confidence - 0.70),  # Near threshold
        -int(d.cross_vessel),      # Cross-vessel
        -d.triage_score            # High similarity
    ))
```

Continuous improvement: Learn from review decisions

A4: Why Not Use Existing MDM Tools?

Aspect	Traditional MDM	ML/Statistical	DEMA-LLM
Input Type	Structured data	Labeled training data	Messy interview text
Matching	Rules-based	Statistical patterns	Semantic understanding
Vessel Aware	No	No	Yes (breakthrough)
Explainability	Rule trace	Black box	100% citations
Schema Reqs	Predefined	Feature engineering	Discovers unknown
Cost	$100K-1M license	Data scientist time	$0.015/decision
Accuracy	60-70%	Varies (70-85%)	80-95%

Unique DEMA-LLM Advantages:

✓ Handles unstructured interview data

✓ Cross-vessel semantic equivalence

✓ No training data required

✓ Complete audit trail

Patent-Pending Innovation:

"Vessel differences are context, not conflict"

Cross-vessel methodology solves problem that plagued data management for decades

A5: Key Implementation Code Sketches

Triage Blocking (O(n²) → O(n)):

```
def generate_blocks(elements):
    blocks = defaultdict(list)
    for elem in elements:
        clean = elem.name.strip().upper()
        if clean:
            block_key = clean[0]  # First char
            blocks[block_key].append(elem.id)
    return blocks
```

Cross-Vessel Gate:

```
def check_cv_gates(edge):
    if edge.confidence < 0.85:
        return False  # Confidence gate
    actors_overlap = jaccard(
        edge.left_actors,
        edge.right_actors
    )
    if actors_overlap < 0.5:
        return False  # Actor gate
    return True
```

Confidence Calibration:

```
from sklearn.isotonic import IsotonicRegression

iso_reg = IsotonicRegression()
iso_reg.fit(
    model_confidences,  # What model said
    actual_correct      # Ground truth
)
calibrated = iso_reg.predict(new_conf)
```

Deterministic Pair ID:

```
def compute_pair_id(left_id, right_id):
    # Ensure deterministic ordering
    min_id, max_id = sorted([left_id, right_id])
    content = f"{min_id}|{max_id}|42"
    hash_hex = sha256(content.encode()).hexdigest()
    return f"P-{hash_hex[:8]}"
```

Not black magic - just thoughtful engineering with LLMs as a tool