Paper ML08

From Schedules of Activities (SoA) to USDM: Automating Protocol Extraction Using Large Language Models

Kerstin Forsberg, Data4Knowledge, Kungälv, Sweden
Johannes Ulander, Data4Knowledge, Umeå, Sweden

ABSTRACT

The SoA Grid Framework shows how Large Language Models (LLMs) can be used for clinical protocol digitalization. Schedules of Activities (SoA) become complex grids as humans attempt to represent hierarchical structures, conditions, and domain-specific relationships in tabular format. This complexity makes automation difficult. We use multimodal LLMs across the development process: refining schemas, designing transformation logic, generating code, and extracting structured data via API calls. This represents early work exploring how an LLM can work as both a development collaborator and an extraction engine. Key challenges included crafting effective prompts and system messages to handle document variations and creating evaluation methods. We adopted the 'grid' metaphor to help LLMs understand relationships beyond tabular data. As LLM capabilities rapidly evolve, we need to keep updating our methods. This framework integrates with data4knowledge's USDM technology demonstrator, creating a seamless path from protocol documents to graph-based study definitions for protocol-to-SDTM automation.

SETTING THE SCENE

PROJECT CONTEXT

At CDISC EU Interchange 2025, we demonstrated "Protocol PDF to SDTM in 15 minutes" [1], a complete protocol digitalization pipeline. Now we dive into the specifics behind the Schedule of Activities (SoA) extraction, recognizing that we have seen interesting work in this space before, including PHUSE EU Connect 2024 presentation on Development of Unified Study Definitions Model (USDM) through translation of human-readable protocols [2].

The SoA Grid Framework (now renamed to SoA2USDM) we present here builds a "live PRD" for the community. A Product Requirements Document (PRD) traditionally defines what a product should do, but in the era of AI-assisted development, the concept of a "live PRD" has emerged, particularly with the rise of "vibe coding" where developers collaborate directly with AI tools to rapidly prototype and iterate. Instead of static documents, a live PRD is a working demonstration that evolves through testing, showing both what's possible and how it works.

THE COLLABORATION VISION

Mark Kramer's foundational work identifying barriers [2,3] provides essential context. His systematic evaluation of 12 "best-in-class" PDF extraction tools with consistently poor results [4] demonstrates that incremental improvements are insufficient. Building alongside initiatives like Protocol Miner [5], this community challenge requires multiple approaches.

Our contribution: an approach where we separate structural extraction from semantic interpretation, which our experimental framework demonstrates. Rather than forcing LLMs into unnatural behaviors through complex prompts, we developed the "grid metaphor" treating SoA tables as coordinate systems where every element has a precise location, see Figure 1.

WHY THIS MATTERS NOW

USDM v4 adoption is accelerating while legacy protocols remain valuable but inaccessible. AI capabilities evolving to a point where complex document understanding becomes feasible. The industry appears ready for automation solutions that can unlock clinical trial knowledge currently locked in documents.

The integration with data4knowledge's USDM technology demonstrator creates a seamless path from protocol documents to graph-based study definitions, enabling the complete "Protocol PDF to SDTM in 15 minutes" vision. This convergence of mature standards, capable AI technology, and proven integration creates the opportunity to bridge legacy protocols with modern USDM-driven workflows.

COLLABORATIVE DEVELOPMENT NOTE

This paper and presentation exemplify the collaborative AI development we describe. As non-native English speakers, we used LLMs as writing partners to organize concepts and refine explanations. This process shouldn't be underestimated, it goes beyond translation to help shape thinking and structure complex ideas in a non-native language, demonstrating how AI can help prevent language differences from hindering international teams' ability to effectively communicate their work.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	**Study Phase**	**SCREENING**		**TREATMENT PERIOD**										**FU**
2	**Period**	**Screening**		**Period 1**				**Washout**		**Period 2**				**FU**
3	**Visit**	Pre-Screen	Screen	V1	V2	V3	V4	WO	WO+1	V5	V6	V7	V8	EOT
4	**Study Day**	Day -28	Day -7	D1	D8	D15	D22	D29	D36	D43	D50	D57	D64	D71
5	**Visit Window**[b]	±7 days	±3 days	±2	±2	±2	±2	±2	±2	±2	±2	±2	±2	±3 days
6	**In-clinic**[a]		X	X									X	X
7	**TeleVisit**[a]				X	X	X	X	X	X	X	X		
8	Informed Consent		X											
9	Medical History		X											
10	Physical Exam		X	X									X	X
11	**Vital Signs**[d]													
12	*Blood Pressure*		X	X	X	X	X	X	X	X	X	X	X	X
13	*Heart Rate*		X	X	X	X	X	X	X	X	X	X	X	X
14	**Laboratory Tests**[d]													
15	*Chemistry Panel*		X	X			X					X	X	X
16	Drug Administration[c]			X	X	X	X			X	X	X	X	
17	Adverse Events		X	X	X	X	X	X	X	X	X	X	X	X

Figure 1. The “Grid”, treating SoA tables as coordinate systems for a fictive study

INTRODUCTION: THE PROTOCOL DIGITALIZATION CHALLENGE

THE LEGACY PROTOCOL OPPORTUNITY

Clinical trial protocols contain vast amounts of structured knowledge. Huge archives across ClinicalTrials.gov, regulatory submissions, and sponsor databases contain decades of collected insights about trial design, safety learnings, and operational approaches. This knowledge captures what works, what fails, and why certain designs succeed.

However, this knowledge remains largely inaccessible to systematic usage. Richardson's examination of 40+ protocols revealed rich operational detail in formats optimized for human interpretation rather than computational processing [6]. The result: we have extensive historical data about clinical trial execution but cannot easily use it for benchmarking, meta-analysis, or design optimization.

The gap between document-based legacy protocols and emerging USDM-driven digital workflows creates both challenge and opportunity.

WHY SOA TABLES MATTERS

SoA tables serve as the primary bridge between scientific objectives and operational reality. These tables encode multiple information layers through spatial relationships, hierarchical structures, and footnote-modified semantics that determine how trials actually execute.

SoA tables provide the primary source for Electronic Data Capture (EDC) configuration. Richardson's analysis of the "meaning of X" revealed that a simple mark could represent anything from basic observation to complex workflows requiring multiple actors and sophisticated timing coordination [6].

This complexity reflects the challenge of representing multidimensional operational requirements in two-dimensional tabular formats. Protocols assume contextual knowledge about procedures, regulations, and conventions, creating information density that proven to be challenging for data extraction.

THE EXTRACTION CHALLENGE

Traditional approaches have consistently struggled with SoA complexity. Rule-based systems prove brittle against format variations. Simple Natural Language Processing (NLP) approaches miss spatial relationships conveying critical timing information. Kramer's systematic evaluation of 12 "best-in-class" PDF extraction tools produced consistently poor results [4], demonstrating that incremental improvements cannot overcome fundamental limitations.

Kramer's barrier analysis identified specific challenges: merged cells conveying temporal information, context-dependent interpretation in junction areas, semantic inconsistencies, and grid misalignments [3]. These reflect deeper issues about how operational knowledge gets encoded in human-optimized document formats.

The failure of traditional extraction approaches points toward architectural rather than technical challenges. Success requires understanding not just what appears in tables, but how spatial relationships convey meaning, how hierarchical structures represent dependencies, and how footnotes modify specifications.

LLMs with advanced document understanding create new possibilities. However, success requires approaches that work with LLM natural behaviors rather than forcing them into rule-based patterns. The key insight: treating extraction as collaboration between human domain knowledge and AI pattern recognition, rather than attempting to automate away complexity entirely.

BACKGROUND: EVOLUTION OF PROTOCOL DIGITALIZATION

STANDARDS EVOLUTION

The path toward digital protocols has been evolutionary rather than revolutionary. Early efforts focused on document standardization, moving from paper protocols to structured PDFs with consistent sections. The ICH M11 Clinical electronic Structured Harmonised Protocol represents significant progress, providing standardized organization while maintaining largely unstructured content [1].

USDM v4 changed protocols from document-based to data-based representation [3]. Developed through CDISC collaboration with TransCelerate's Digital Data Flow initiative, USDM provides standardized class diagrams with CDISC terminology for interoperability [4].

Parallel developments in healthcare interoperability have influenced clinical research. The HL7 FHIR ecosystem now includes definitional resources for research requirements, with the Vulcan Accelerator connecting CDISC, HL7, and ICH M11 initiatives [6,7]. Richardson's work demonstrates how FHIR PlanDefinition resources can represent SoA requirements [6].

RELATED WORK

Multiple approaches have emerged for transforming legacy protocols into structured formats. Last year's PHUSE presentation by Kestemont, Rogiers, and Juneja demonstrated USDM development through combined NLP and LLM approaches [1]. Their work established feasibility while highlighting SoA extraction complexity.

Protocol Miner represents another significant effort in this space, with ongoing development work over the past year was showcased at the CDISC AI Innovation Challenge at US Interchange 2025 [5]. These parallel initiatives demonstrate community recognition that automated protocol digitalization requires sophisticated approaches beyond traditional document processing.

Richardson's systematic analysis of SoA characteristic attributes provides important foundation work [6]. By examining 40+ protocols to identify minimal viable attributes for FHIR representation, his research establishes what information must be preserved during transformation.

FROM TEXT TO VISION: UNDERSTANDING EXTRACTION COMPLEXITY

Protocol digitization requires different AI approaches depending on document structure complexity. Basic protocol elements like titles, inclusion/exclusion criteria exist as straightforward text sections. These elements work well with text-based LLM extraction as the information appears in clear, sequential narrative form that matches natural language processing strengths.

However, SoA tables represent a fundamentally different challenge going beyond digitization to true digitalization. They encode multiple information layers simultaneously: temporal frameworks, procedural sequences, conditional logic, and cross-references all compressed into grid format. This spatial encoding requires multimodal LLM capabilities that can process visual document structure alongside textual content. The multimodal LLM must understand not just what text appears, but where it appears and how spatial relationships convey meaning.

Consider the difference: an inclusion criterion states "Age ≥18 years" as clear sequential text. An SoA table might encode the same visit timing through column positioning, header hierarchies, and grid indicators that require visual processing to interpret correctly. The complexity jump from sequential text to spatial grids necessitates the architectural approach we describe below.

LLM:S AS DEVELOPMENT COLLABORATORS

Recent developments in AI-assisted development introduce a new way to handle complex challenges. Rather than viewing LLMs just as extraction engines, they can be seen as collaborative partners in system design, schema evolution, and code generation.

This dual role, development collaborator AND extraction engine, enables rapid iteration on approaches while handling document variations. The key insight: successful LLM implementation requires understanding both domain complexity and AI behavioral patterns, then designing architectures that leverage both effectively.

Understanding this dual role helps explain the evolution of our approach. What follows isn't a straightforward implementation story but rather almost a year-long exploration of how to work effectively with rapidly improving LLM capabilities. Each step in our journey taught us something different about the balance between architectural vision and practical constraints, between forcing LLMs into fixed behaviors and working with their natural strengths.

OUR JOURNEY

Our work spans a year of experimentation during a period of rapid advances in LLM capabilities. What emerged wasn't a straight line from problem to solution, but cycles of learning: build something, watch it break, understand why, rebuild it differently.

STEPS 1-3: LEARNING FROM AUTOMATED APPROACHES

Late 2024, we chose Anthropic's Claude based on Johan Sanneblad's analysis identifying Claude 3.5 Sonnet as "amazingly good" for complex reasoning [7]. More importantly, we stuck with this choice. In periods of rapid AI advancement, constantly switching models creates instability. We maintained focus, accumulating domain knowledge in our prompts rather than constantly adapting to different model behaviors.

Initial experiments showed promise: LLMs could understand table structure semantically. But turning that understanding into reliable structured data took careful architectural thinking. Through spring 2025, we worked through multiple implementations. Each version fixed problems from the previous one: timing in headers, hierarchies through indentation, event-triggered visits that didn't fit fixed timepoints.

By May 2025, a three-pass structure supported our CDISC conference demonstration. The core insight had become clear: progressive refinement through sequential passes, each building context from the previous. This wasn't planned architecture, instead it emerged from observing what worked. The prompts became verbose, encoding hard-won learning. "Accept activity names with leading spaces" came from protocols using indentation for hierarchy. "Treat arrows as continuation markers" emerged from event-driven designs.

Then we tried to clean up the design: a seven-pass vision with mechanical extraction (Passes 1-5) cleanly separated from semantic analysis (Pass 6). On paper, this looked elegant. Each component with single responsibility. Clean interfaces. Easy to test. But implementation revealed problems, mechanical passes became brittle without semantic guidance. After months working toward elegant architecture, with conference demonstration weeks away, reality prevailed: pragmatic beats elegant when elegant doesn't ship.

We returned to a working multi-pass structure, refined to five passes. Pass 1 established coordinate systems, Pass 2 identified schedule properties, Pass 3 catalogued activities, Pass 4 mapped scheduled activities, Pass 5 captured annotations. The verbose prompts stayed (they encoded hard-won knowledge). The deliberate overlaps stayed (they worked in practice).

Critical lessons learned:

- **Prompt stability matters.** Well-intentioned improvements during development conversations degraded working extraction. We learned to treat prompts like production code.
- **Semantic understanding can't be separated**. LLMs naturally interpret semantically. Fighting this creates brittleness.
- **Domain expertise encodes in prompts.** Each protocol variation taught patterns with real value.
- **Coordinate systems need consistency.** Independent passes creating separate coordinate systems require complex reconciliation.

The multi-pass pipeline worked for our three test protocols with known patterns. But brittleness remained. Protocol variations required prompt refinement. Debugging happened after extraction completed. The architecture fought against LLM natural behavior rather than working with it.

Then September 2025 changed what was possible.

STEP 4: CONVERGENCE OF VISION AND SCHEMA

The multi-pass pipeline extracted components separately: one pass for columns, another for properties, another for activities. Each pass created its own view of the table structure, requiring complex reconciliation. Then September 2025 brought together two elements that changed our approach fundamentally

Improved vision capabilities emerged from two product launches in late September 2025:

Claude Sonnet 4.5 shows improved visual table understanding. (Anthropic, Introducing Claude Sonnet 4.5, September 2025). Where earlier versions struggled with merged cells, continuation markers, and multi-level headers, Sonnet 4.5 understands spatial table structure as semantic grids. The complex example SoA with compressed multi-level headers, scattered continuation markers, and hierarchical activity groupings, produced if not perfect so at least workable Excel files, yet another new functionality from Anthropic introduced in September 2025.

LandingAI's Document Pre-trained Transformer (DPT-2) takes a different approach to PDF-to-structured-data than traditional OCR. Instead of extracting text, DPT2 understands document layout as semantic structure. Tables are hierarchical information organized spatially. (LandingAI News Release, September 30, 2025). Early testing showed DPT2 could extract complex multi-level SoA tables directly to structured format, preserving merged headers, but still struggles with continuation markers, and hierarchical relationships.

The multi-pass pipeline taught us what needed to be in the schema. The vision breakthroughs made it possible to extract complete table structure. These two elements coming together enabled a different workflow.

THE SOA2USDM SCHEMA

The multi-pass pipeline taught us what information needed preservation during extraction. The vision breakthroughs made complete structure extraction feasible. But connecting these required a schema that could bridge between spatial table layouts and USDM semantic requirements

SoA2USDM v7.3 implements an insight that became clear during our journey: headers are schedule properties at different hierarchical levels. Rather than treating headers as mere column labels, the schema recognizes them as schedule-organizing information existing at multiple nested levels:

- Level 1 (top): Study phases/periods spanning multiple visits
- Level 2 (middle): Visit labels or period groupings
- Level 3 (bottom): Specific timing details (days, weeks)
- Level null: Non-hierarchical properties (modality, requirements)

This hierarchical view lets the schema capture what traditional methods miss: the meaning encoded in how the table is laid out.

Excel Row	Property ID & Type	Hierarchical Level	Property Comment (Semantic Description)
Row 1	property-001 phase_header	Level 1	*"Top-level study phase headers: **Screening** period for eligibility assessment (columns B-C), **Treatment Period** spanning multiple visits including active treatment and washout (columns D-J), and **Follow-up** phase for safety monitoring (column K)"*
Row 2	property-002 phase_header	Level 2	*"Period groupings within treatment phase: **Screening** for enrollment, **Period 1** for initial treatment (3 visits), **Washout** for drug clearance (2 visits), **Period 2** for crossover treatment (3 visits), and **Follow-up** for long-term safety"*
Row 3	property-003 visit_label	Level 3	*"Visit identifiers using sequential codes: **Pre-Screen** for initial contact, **Screen** for eligibility confirmation, **V1-V8** for treatment visits numbered sequentially, **WO/WO+1** for washout monitoring visits, and **EOT** for end of treatment assessment"*
Row 4	property-004 timing_detail	Level 4	*"Study day numbers with **Day 1 as first treatment dose**: Negative days (Day -28, Day -7) for screening period preceding treatment, then sequential weekly intervals from **D1 through D71** covering entire treatment and follow-up duration (10 weeks total)"*
Row 5	property-005 requirement	Level 5	*"Visit window tolerances in **±N days format**: Wider windows for screening visits (±7 days, ±3 days) allowing scheduling flexibility, tighter windows for treatment visits (**±2 days**) ensuring protocol adherence, and moderate window for final visit (±3 days) balancing completion and compliance"*
Row 6	property-006 visit_modality	null	*"**X marks** indicating **in-person clinic visit modality** required for eligibility assessments (Screen, V1), endpoint measurements (V8), and safety closeout (EOT) where direct examination or procedures are necessary"*
Row 7	property-007 visit_modality	null	*"**X marks** indicating **remote telehealth visit modality** permitted for interim treatment visits (V2-V7) and washout monitoring (WO, WO+1) where patient safety monitoring and compliance assessment can be conducted remotely"*

Figure 2. SoA2USDM Schedule properties from the fictive study introduced in Figure 1.

Complete traceability through source attribution, every element in the schema traces back to exact Excel coordinates:

- schedule_properties link to property_label_source (row, column, cell_value)
- activity_rows preserve activity_name_source with indentation tracking
- scheduled_activities map to precise (row_position, column_position) pairs
- merged cells get explicit tracking (merged_cell_range, is_merged_cell)

This source attribution creates complete audit trails. When USDM conversion questions arise, we can trace back to the exact source cell that informed each decision.

The schema requires meaningful semantic descriptions, this means semantic interpretation not mechanical extraction.The property_comment field isn't optional formatting, it's mandatory semantic documentation. Rather than "Visit" (repeating the label), property_comment must describe: "Visit identifier labels: Screening, Randomization, 1st eligible HAE attack, 2nd eligible HAE attack..."

This forces explicit semantic interpretation during extraction. LLMs don't just transcribe, instead they must understand what each row represents in the study design.

THE BRIDGE TO USDM

SoA2USDM JSON isn't just structured extraction, it's a USDM-ready representation. The mapping is direct:

- schedule_properties (phase_header) → StudyEpoch
- schedule_properties (visit_label) → StudyEncounter
- activity_rows → StudyActivity
- scheduled_activities → activity-to-encounter linkages

The schema structures extraction output to align with USDM entities, eliminating mechanical translation requirements. However, semantic mapping decisions remain; particularly how schedule properties map to USDM's multiple timeline constructs (StudyEpoch sequencing, StudyEncounter timing, ScheduledActivityInstance windows). SoA2USDM preserves complete source context and hierarchical relationships, providing the semantic foundation these mapping decisions require. Integration with data4knowledge's USDM technology demonstrator progresses as we refine these timeline interpretation rules.

THE TWO-STEP CONVERSATIONAL WORKFLOW

With the schema foundation established, we can now explain how it guides the two-step extraction workflow. Instead of extracting components separately, we extract complete structure once, then interpret it semantically using the SoA2USDM schema.

Step 1: PDF/Image to Excel creates a format domain experts can verify visually. Using Claude Sonnet 4.5's vision understanding and Excel generation functionality, we generate Excel files preserving spatial layout from source. Table_XX sheets maintain grid structure. Table_XX_Annotations capture continuation markers, merged cells, hierarchical indicators. Domain experts verify extraction visually before any JSON generation.

Figure 3 shows source PDF (left) and a rendering of the extracted Excel (right) side-by-side.

The CDISC Pilot Study represents our baseline case with straightforward sequential design with 16 visit columns (Visit 1-13 at Week -2 through Week 26, plus Early Termination and Retreatment visits), clear activity rows, and minimal merged cells. Initial extraction captures the complete structure accurately. Domain experts verify by scanning the extracted table against the source. Errors are obvious. For simple tables like this, verification confirms extraction is ready for next step.

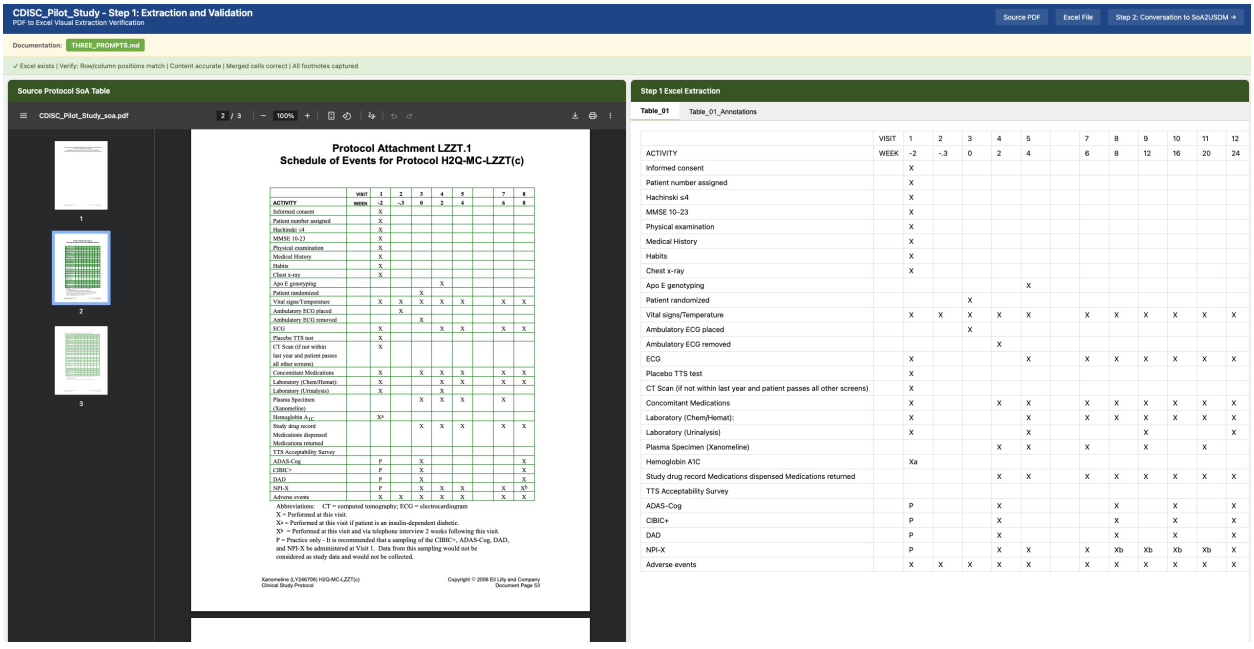

Figure 3. Step 1 Extraction, CDISC Pilot Study

Why Excel intermediate? Let us pick a more challenging SoA table from another protocol, a trial from Alexion, (NCT04573309) demonstrates this value. Initial extraction captured the complete 24-column structure with all activities and cell markers. The foundation was solid. But visual verification revealed issues that took minutes to fix in Excel.

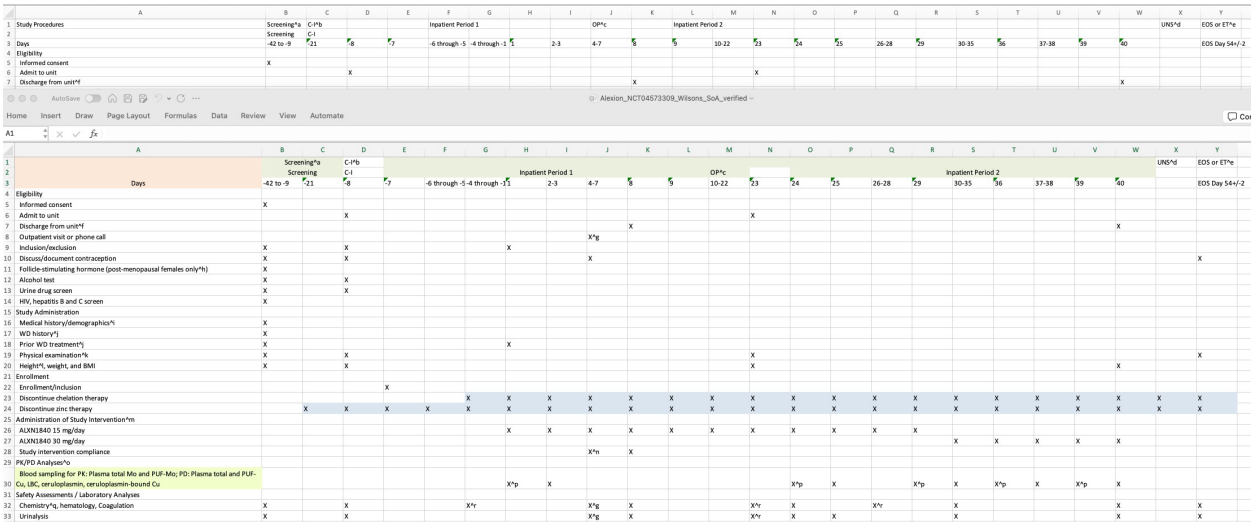

Figure 4. Excel Verification and Corrections

Figure 4 shows extracted table (top) and verified table (bottom). Color coding marks correction types: pink highlights removed structural rows like "Study Procedures" (section headers, not activities), green shows refined header merge ranges, blue indicates clarified continuation markers, beige marks page boundary handling.

The verification reality: Initial extraction gets the hard parts right. Column structure, activity hierarchy, marker placement. What needs human verification? Merge ranges in complex multi-level headers. Distinguishing section headers from actual activities. Edge cases where continuation markers could be interpreted multiple ways. The table shows the pattern: capture structure automatically, refine details visually.

Step 2: Excel to Validated JSON applies the SoA2USDM Schema v7.3 conversationally. Claude interprets validated Excel semantically, generating USDM-ready JSON with complete context preserved. This is where the schema insight matters. Claude understands that hierarchical header rows are schedule properties at different levels, not just formatting. When Claude identifies interpretation questions (visit window formats, activity groupings, temporal relationships), we address them through conversation before JSON generation.

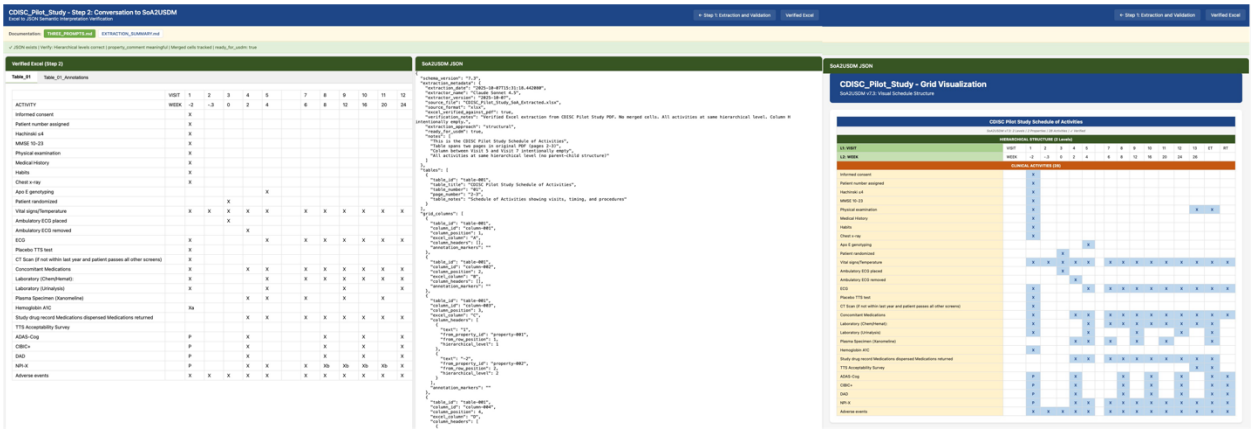

Figure 5. Step 2 Conversion to SoA2USDM, CDISC Pilot Study

Figure 5 shows Excel-to-JSON conversion with three-panel view. Verified Excel (left) provides the source structure. Generated JSON (center) shows the SoA2USDM output with schema validation. Grid visualization (right) renders the JSON back as a visual table, proving the conversion preserved complete context. The CDISC Pilot Study JSON includes tables metadata, grid columns with visit/week headers, schedule properties (Visit and Week rows at different hierarchical levels), activities with parent-child relationships, and scheduled_activities mapping each X marker to its activity and column position. The round-trip visualization confirms: what you see in Excel equals what's captured in JSON.

The conversational approach allows real-time clarification. When Claude encounters interpretation questions: visit window formats, activity grouping conventions, temporal relationshipsm we address them before JSON generation. Simple tables like CDISC Pilot need minimal conversation. Complex tables with compressed headers, ambiguous timing notation, or protocol-specific conventions require more dialogue. The schema provides the structure. Conversation provides the semantic clarity.

What does mean in practce? Let us have a lock some snippets of on the conversation to extract the Excel and to JSON for the complex example.

Figure 6 shows the actual interaction during Stage 1 extraction. The left panel displays the prompt visible to Claude with instructions for the three-stage extraction process with specific guidance on column structure verification. The right panel shows Claude's response asking "How many columns do you count in the table?" and proposing its analysis: 24 distinct timepoint columns based on day numbers and visit labels. Human verification confirms this count, allowing Stage 1 to complete.

The collaborative review continues. The summary shows successful structure capture: Table_01 created with columns A-X, annotations placed. But reviewing the extracted Excel against the source PDF reveals issues with continuation lines. Row 23 (Discontinue chelation therapy) and Row 24 (Discontinue zinc therapy) show activities spanning multiple visits. The X marks need adjustment to correctly populate all intended cells. This human-identified correction gets addressed before proceeding to JSON generation.

This is conversational extraction in practice. Claude handles structure capture. Human expert reviews results and identifies domain-specific issues like continuation marker placement. Together, they refine the extraction before moving to the next stage. For complex tables, this collaborative pattern catches issues that pure automation would miss.

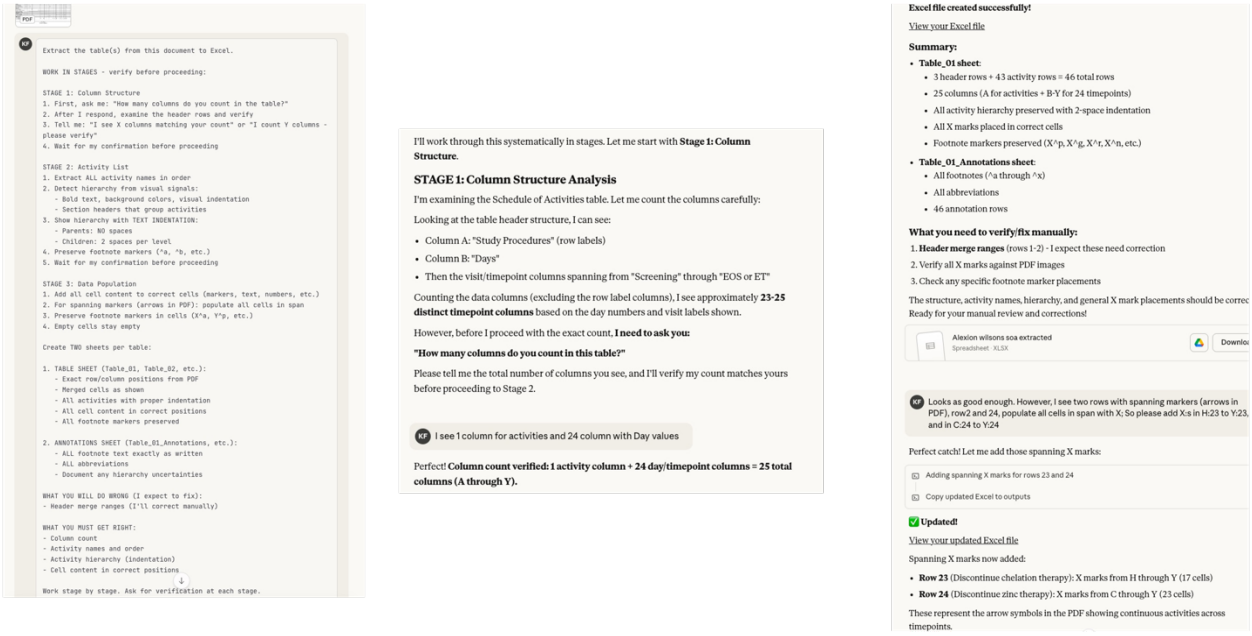

Figure 6. Conversational Extraction in Practice

REALITY CHECK: WHAT WORKS, WHAT DOESN'T

Current capabilities: Complex tables that previously failed now extract successfully. The complex SoA table; 24 columns, three header levels, interleaved properties, extracts reliably through the three-prompt workflow. Processing takes approximately one hour per protocol including conversational refinement and validation.

Current limitations: Each protocol type may require conversational adjustment. Visit window formats vary across therapeutic areas. Activity grouping conventions differ by sponsor. Temporal relationship encoding varies by protocol design. The conversational workflow handles these variations through dialogue. But extraction isn't fully automated.

Beyond conversation: what's really happening: The term "conversational" undersells the architectural reality of Step 1, where complex SoA tables become Excel files. When you describe a table and Claude delivers Excel, what actually happens is: vision analysis of PDF structure → Python code generation for extraction → execution in sandboxed container → Excel file creation with merged cells and proper positioning → delivery for verification. If issues are identified and needs to be handled, the cycle repeats: correction code → re-execution → re-delivery.

This is collaborative development through natural language. The user bring domain expertise and visual verification capability. Claude generates and executes extraction programs. The interface hides significant capability: code generation, container execution, file I/O, iterative refinement. What appears as "conversation about tables" is actually pair programming with direct code execution.

The breakthrough emerged in September 2025 when improved vision understanding (Claude Sonnet 4.5) combined with execution capability (Computer Use, October 2024). This architectural shift explains why the three-prompt workflow succeeds where our multi-pass pipeline struggled. Instead of fighting LLM natural behavior through predetermined mechanical passes, we work with it: semantic understanding guides code generation, domain expertise guides refinement, execution delivers verifiable artifacts.

Brittleness we're honest about: Protocol-specific conventions require human judgment. Footnote interpretation sometimes needs clarification. Unusual continuation marker patterns may need explanation. This isn't failure. It's acknowledgment that protocols encode decades of evolving conventions that resist complete automation.

Oncology protocols with cycle notation (C1D1, C2D15, etc.) require schema extensions beyond v7.3's visit-based model. Extended prompts provide workarounds, but systematic cycle support needs explicit cycle_number, day_within_cycle properties and repeating pattern representation.

What we achieved: The conversational workflow with SoA2USDM Schema makes complex SoA extraction feasible. Tables that resisted all previous automation attempts now extract with human-guided assistance. The two-step pattern (complete structure extraction, then semantic interpretation) works with LLM natural strengths rather than fighting them. The schema provides structure. Conversation provides clarity. Together, they bridge the gap between legacy protocol documents and modern digital workflows.

FROM LIVE PRD TO COMMUNITY RESOURCE

We introduced this work as a "live PRD", an executable demonstration rather than static specification. The SoA2USDM schema exemplifies this approach. It didn't emerge from requirements documents but from direct experimentation: attempting extractions, analyzing failures, refining architecture, testing against complex protocols.

Several figures in this paper are screenshots from a user interface mockup, itself built through collaborative AI development using Python notebooks that generate interactive validation views of the extracted JSON.

The live PRD documents not just what worked, but why certain patterns proved reliable while others needed judgment. Technologies like LandingAI DPT2 and Claude Sonnet 4.5 API can be evaluated against these specific learned patterns. Organizations building automated extraction systems now have a foundation: an architectural pattern that works, a schema that's USDM-ready, and documented understanding of where automation succeeds and where human expertise adds value.

CONCLUSION

The conversational workflow works for the complex SoA tables we tested. Processing takes about an hour per protocol with expert guidance. Our work with the USDM protocol collection and additional protocols during development taught us about different patterns.

Modern design concepts continue evolving. Adaptive trial designs, master protocols with multiple sub-studies, platform trials, basket trials, each brings scheduling patterns we haven't fully explored. Therapeutic area conventions vary. Different sponsors have different “SoA styles”. The conversational approach handles variation through dialogue, but we need broader experience to understand where automation works well and where expert conversation remains essential.

Decades of clinical trial knowledge sit in legacy protocols across ClinicalTrials.gov, regulatory submissions, and sponsor databases. USDM v4 adoption is accelerating. Organizations need systematic transformation of protocol portfolios. The technology is capable. The architecture works for the cases we've tested. The question is how to scale while learning from the variety we'll encounter.

The SoA2USDM JSON feeds directly into data4knowledge's USDM technology demonstrator. Schedule properties become USDM study design elements (epochs, visits, timing windows). Activities map to USDM activity types. Scheduling markers create relationships between activities and visits. From there, USDM drives downstream SDTM generation. This completes the "Protocol PDF to SDTM in 15 minutes" vision we demonstrated at CDISC EU Interchange 2025. SoA extraction is the critical middle layer, bridging legacy protocols with modern USDM-driven workflows.

REFERENCES

1. Kestemont J, Rogiers S, Juneja S Development of USDM through translation of human-readable protocols. PHUSE EU Connect, ML02, 2024.
2. Kramer M. The Hidden Challenge of Digitizing Clinical Trials: Why Converting Legacy Protocols Is Harder Than You Think. Medium, 2025.
3. Kramer M. Barriers to Automatic Conversion of Clinical Trial Schedules of Activities to Structured Data. [White Paper], 2025.
4. Kramer M. I Tested 12 "Best-in-Class" PDF Table Extraction Tools, and the Results Were Appalling. Medium, 2025.
5. Kramer M. Protocol Miner for the AI Innovation Challenge. CDISC, 2024. [Video presentation] [Author].
6. Richardson A. Representing Clinical Study Schedule of Activities as FHIR Resources: Required Characteristic Attributes. Journal of the Society for Clinical Data Management, 2024.
7. Sanneblad, J. (2024, June 24). The biggest news was the launch of Claude 3.5 Sonnet that performs amazingly good for coding. Tech Insights [Newsletter]

ACKNOWLEDGMENTS

The author thanks Mark Kramer for his foundational work identifying barriers to SoA extraction and his systematic evaluation of PDF extraction tools, which provided essential context for our work.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Kerstin Forsberg
Company: data4Knowledge ApS
Address: Byledet 15, 2820 Gentofte, Danmark
Work Phone: +46 703203907
Email:kf@data4knowledge.dk
Website: https://d4k.dk/