Paper SA02

The Art of Timely Deliverables:
How We Learned To Stop Worrying and Love the Deadline

Søren Fjelstrup, Novo Nordisk A/S, Søborg, Denmark

ABSTRACT

In the dynamic environment of Health Technology Assessment (HTA), timely data delivery is critical to meeting the demands of governmental bodies. This paper outlines our approach to managing HTA data requests efficiently. We will explore our centralized document system that tracks incoming requests, enabling effective prioritization, and by leveraging a sprint board for task management, we assign tasks rapidly. To further streamline the process, we use an LLM agent to translate incoming requests into a CDISC friendly format. This ensures that standard requests can be processed through a series of independent, modular programs, while custom programs are developed only for unique requirements.

This bifurcated task management method reduces turnaround times and enhances productivity and allows for parallel workflows. Discover how these simple optimizations of workflow processes and pipeline structures can transform your operations, enabling you to work more efficiently and with greater reliability and stop dreading short deadlines.

INTRODUCTION

The Health Technology Assessment (HTA) environment presents unique challenges for pharmaceutical companies. Governmental agencies frequently request new analyses with little warning and no room for negotiation, creating what we internally refer to as “bombs from a blue sky” that have major impact if organizations are unprepared. Since HTA works within a framework of individual countries simultaneously submitting, this is exacerbated even more than in a regulatory setting.

Traditional approaches to handling these requests suffer from numerous problems: endless email chains, unclear request formats, manual translation of business needs into technical specifications, and scattered, unfindable documentation. These issues are compounded by the cross-departmental nature of HTA work, where maintaining data flow across multiple departments becomes critical for success.

Without proper systems in place, organizations face enormous time consumption, project-derailing bottlenecks, and coordination nightmares. This paper outlines a structured, scalable method that works in practice to address these challenges and transforms reactive crisis management into proactive, predictable operations and while this approach was designed for HTA, it is useful for any project.

Our solution consists of five interconnected components that work together to create a seamless delivery pipeline: a single Excel request form, AI-powered specification translation, Kanban board management, modular standard programs, and an automated documentation system.

SINGLE EXCEL REQUEST FORM

The entry point of our system is a standardized form. Rather than allowing requests to come through multiple channels, we implemented a single Excel form as submission point. The fields of which are shown in Table 1.

Excel was chosen deliberately for several practical reasons. First, it is universally familiar, everyone knows how to use it, eliminating the need for specialized training. Second, access control is straightforward through SharePoint, allowing us to manage permissions efficiently. Third, the format naturally constrains verbosity because it is difficult to write extensively in Excel cells, requesters must be concise with their descriptions, creating a cleaner overview. The detailed specifications are then provided in a follow-up email to the HTA contact.

This approach provides a clear process: requesters complete the form and send a follow-up email to a single contact for elaboration. This standardization is especially important given our diverse stakeholder base, many of whom only submit requests once or twice a year and cannot be expected to remember complex procedures. The standardized form enables rapid response to regulatory requests, shows requesters what other projects are ongoing, and gives a convenient overview for expected deliveries and deadlines. The compromise here is to give enough flexibility to allow bespoke analysis, but enough friction to encourage standard implementations. Importantly, the Excel document is publicly accessible to all stakeholders, which further nudges standardisation, allows requesters to see the scale of current demands, and enables them to learn from previous requests and potentially discovering that similar work has already been completed or is underway.

	Requester
	Initials of contact
	Study the Analysis is requested for
	Priority of the request (high/medium/low
	Analysis requested if known
	Analysis requested by this timeline
	Is there flexibility around timelines (Yes/No)
	Comments (additional context)

Table 1: Data collection fields in the standardized Excel request form. The form captures essential information about each analysis request including requester details, study information, priority level, analysis specifications, timeline requirements, and flexibility constraints. This structured approach ensures consistent information gathering and facilitates efficient request tracking and prioritization

AI-POWERED SPECIFICATION TRANSLATION

The translation of clinical data analysis requests from natural language to structured technical specifications represents a critical bottleneck in the pharmaceutical development process. Our AI-powered specification translation system addresses this challenge by leveraging Large Language Models (LLMs) to automate the interpretation and structuring of ad hoc analysis requests, particularly those arising from regulatory submissions and agency interactions where rapid turnaround is essential.

SYSTEM ARCHITECTURE

The translation pipeline, illustrated in Figure 1, operates through three distinct phases: Preparation & Parsing, Analysis & Specification, and Output Generation. This modular architecture ensures both flexibility and reliability while maintaining human oversight at critical decision points.

This division of labour mirrors how human experts approach the problem—first understanding the request, then drafting specifications, then reviewing for quality, then refining based on feedback. Each agent is optimized for its specific task rather than trying to do everything at once. This design philosophy recognizes that while LLMs excel at pattern recognition and semantic matching, domain expertise remains essential for validating clinical and statistical appropriateness. The system thus functions as an intelligent assistant rather than a fully autonomous agent, amplifying human capability while maintaining human authority over critical decisions.

Figure 1 illustrates the complete pipeline architecture. The system begins by parsing natural language requests using Claude (Anthropic, 2025) and loading relevant metadata from our clinical database (NNaccess DB). AI agents then determine appropriate analysis types and combine specifications into a final validated output. If parameter mapping is incomplete, the system triggers manual review; otherwise, it automatically generates specification files and Azure DevOps work items.

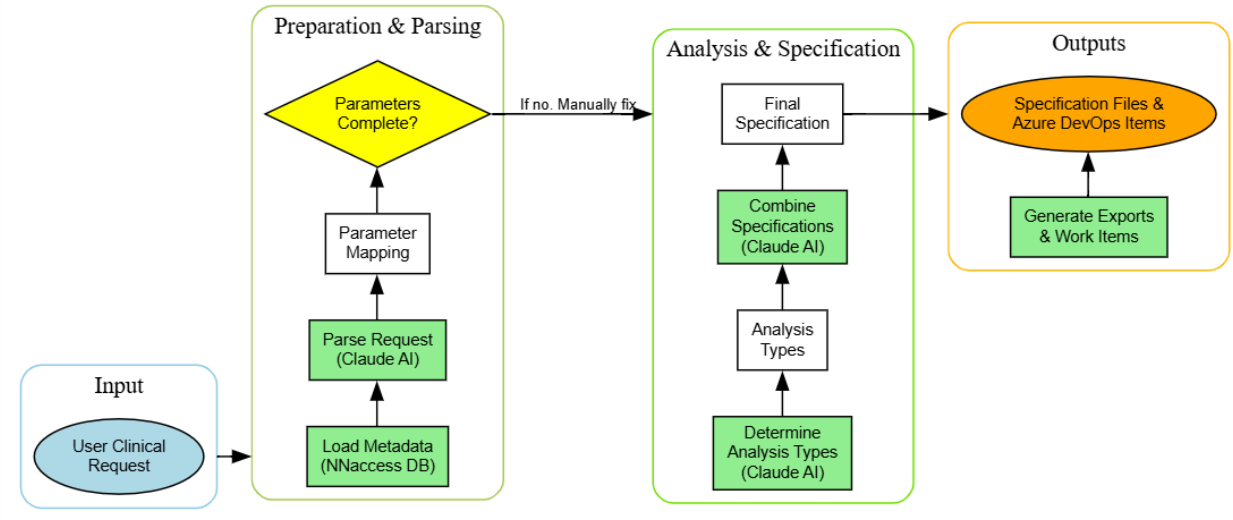

Figure 1 The AI-system operates through three sequential phases. In the Preparation & Parsing phase, user clinical requests are processed alongside metadata. Claude (Anthropic, 2025) performs parameter mapping, with a decision point for manual intervention when parameters cannot be automatically resolved. The Analysis & Specification phase determines appropriate analysis types, which are then combined with the parameter mappings to create a unified specification. The final Outputs phase generates structured specification files and automatically creates corresponding Azure DevOps work items. Green boxes indicate coding operations, while the yellow diamond represents a human decision point, illustrating the system's human-in-the-loop design philosophy.

PHASE 1: PREPARATION & PARSING
The initial phase begins with metadata ingestion from the NNaccess database system, loading CDISC-compliant parameter dictionaries and study-specific metadata. These inputs serve as the semantic foundation for subsequent translation steps, providing the controlled vocabulary necessary for accurate parameter mapping.

Upon receiving a user's clinical request in natural language, the system employs Claude Sonnet (Anthropic, 2025) to perform intelligent parsing. The parsing phase implements several heuristics: automatic removal of common prefixes like "History of" or "Proportion of" to improve matching accuracy, disambiguation using factor levels when multiple parameters could satisfy a request and separation of compound requests that contain both analysis specifications and subgroup definitions. Critically, the system identifies parameters that cannot be confidently mapped, flagging them as "MISSING" for manual review, a design choice that prioritizes accuracy over automation.

The output of this phase is a structured table containing four key columns: PARAMCD (the standardized parameter code), PARAM (the full parameter description), Initial request (the original text), and request type (classified as “Subgroup”, “Subpopulation”, “Analysis variable”, or “Not covered”). This structured representation transforms ambiguous natural language into a format amenable to downstream processing while preserving traceability to the original request.

PHASE 2: ANALYSIS & SPECIFICATION
The second phase focuses on determining appropriate statistical methodologies for each identified analysis variable. The system again leverages Claude Sonnet (Anthropic, 2025), but with a distinct prompt engineering strategy focused on statistical analysis classification.

The classification logic incorporates domain-specific rules encoded in the prompt. For instance, requests mentioning "time from randomization to event" trigger Cox regression classification, while "change from baseline" in continuous measures suggests ANCOVA. The system also extracts temporal information, identifying specific visit windows when analyses should be performed. This temporal awareness is crucial for longitudinal clinical trial analyses where the same endpoint may be evaluated at multiple timepoints.

Following individual classification, the system performs table combination operation using fuzzy string matching on the "Initial request" field, it merges the parameter mapping table from Phase 1 with the analysis type table from Phase 2. This fusion creates a specification that links each analysis variable to its standardized parameter code, appropriate statistical method, and temporal scope. The fuzzy matching approach accommodates minor textual variations between the two parsing steps, ensuring robust linkage even when the LLM produces slightly different phrasings.

PHASE 3: OUTPUT GENERATION
The final phase transforms the combined specification into actionable work items. The system generates four distinct output categories, each tailored to specific downstream processes:

Subpopulation and Subgroup Definitions: These tables enumerate the patient populations and stratification variables required for the analysis, providing clear specifications for data subsetting operations.
Cox Regression Specifications: For time-to-event analyses, the system generates a cross-product of subpopulations, subgroups, and endpoints, creating individual specifications for each required analysis. Each specification includes the landmark time (visit, baseline unless otherwise stated), endpoint parameter code, and population definitions—all information necessary for automated analysis execution.
Descriptive Table Specifications: Similarly, descriptive analyses are expanded into individual specifications linking variables, populations, and time periods. This granular specification enables automated generation of baseline characteristics tables and other descriptive summaries.

These outputs can be used in standardized programs, critically, the system integrates with Azure DevOps, automatically creating user stories for each analysis type and organizing them under a parent feature, along with the requested subgroups. This integration bridges the gap between specification and execution, ensuring that translated requirements immediately enter the project management workflow with appropriate tracking and accountability.

Input: Natural Language Request

> *"Analyze obese patients with HFpEF. Compare smokers vs non-smokers for time to CV death and heart failure hospitalization at week 104. Include baseline characteristics: age groups, BMI categories."*

Output: Structured Technical Specification

PARAMCD	PARAM	Request Type	Analysis Type
HFpEFbmi30	HFpEF subpopulation with bmi >= 30	Subpopulation	-
SMOKER	Current Smoker	Subgroup	Cox-regression
TMTCVDTH	Time to CV Death	Analysis Variable	Cox-regression
TMTHSPHF	Time to HF Hospitalization	Analysis Variable	Cox-regression
AGEGRP2	Age Group 2	Analysis Variable	Descriptive tables
BMIGRP30	BMI 30 group	Subgroup	Descriptive tables

Key Benefits:

- **Time Reduction:** Manual process (2-4 hours) → AI translation (2-5 minutes)
- **Standardization:** Automatic CDISC parameter code generation
- **Accuracy:** 95%+ parameter matching with metadata validation
- **Urgency Support:** Immediate turnaround for critical requests

Figure 2 Demonstration of AI-powered translation from natural language business requirements to technical CDISC specifications, showing automatic parameter mapping, analysis type classification, and structured output generation. The example used is reduced due to space concerns, the benefits section is based on a typical request. Input and output are actual examples, reformatted to fit into the paper.

SHORT TERM PLANNING: AZURE DEVOPS IMPLEMENTATION

Our short-term planning utilizes Azure DevOps boards. Each team has its own dedicated board (e.g., Obesity or Diabetes), providing focused project management for specific studies or initiatives. An example of which can be seen in figure 3. Swimlanes are added to make it easier to separate tasks on different projects, and aid prioritisation and programmer assignments.

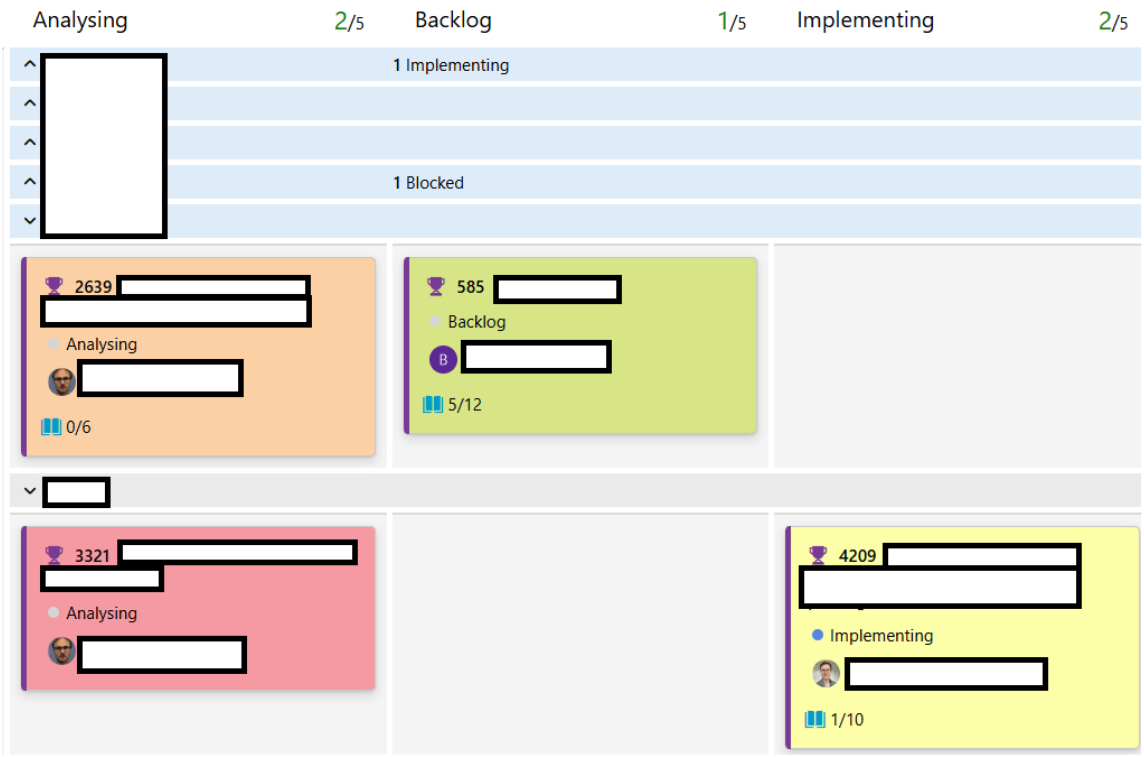

Figure 3: Azure DevOps board example showing a study workflow with features organized across analysis stages. Features are displayed in swimlanes with color-coded priority indicators (red for highest priority, yellow for lower priority, green for lowest priority) and include progress tracking for subtasks (e.g., 0/9, 0/4, 0/11). Each feature card shows the feature number, study path, instance name, assigned team member, current workflow state, and completion metrics.

FEATURE MANAGEMENT

For each request, we create a feature following a standardized naming convention:
Study path (Study name): Instance name. This allows anyone to identify where analysis is performed using which data as well as the requester at a glance.
Each feature includes:

- Estimated work hours (“effort”) for task completion
- Start and end dates for the activity
- Detailed description of the request
- Priority coding using color system (Red, Orange, Yellow, Green in decreasing priority)

A visual priority system combined with real-time status updates means that urgent regulatory deadlines are immediately visible to all team members. So when e.g. an affiliate submits an urgent survival analysis request, it appears with red priority coding, and any team member can see both its importance and current status without requiring manager intervention. To aid this overview, Features are organized in swimlanes corresponding to their study or type (e.g., "Trial A" or "Administrative"), providing visual organization and easy filtering.

FEATURE WORKFLOW STATES

Features progress through clearly defined states that provide transparency and accountability:

- **Funnel**: Initial entry point where statistics teams complete task details
- **Analyzing**: Active scoping phase for requirement clarification by Clinical Data Scientists
- **Backlog**: Ready to start but not yet initiated
- **Implementing**: Active development phase
- **Review**: Completed work undergoing quality assurance
- **Blocked**: Tasks waiting for external dependencies (requires explanatory comments)
- **Closed**: Finished and delivered to requester

Each feature is decomposed into specific user stories that represent contained sub-deliveries. Standard user stories may represent single deliverables like ``Descriptive tables'' or individual figures, depending on delivery package complexity.

USER STORY WORKFLOW

User stories follow their own progression through defined states:

- **Scoping**: Initial state for all user stories
- **Open**: Ready for active work
- **Active**: Currently being developed
- **Ready for Review**: Completed and awaiting quality assurance
- **Blocked**: Waiting for dependencies (requires explanatory comments)
- **Reviewing**: Under active review (reviewer assigns themselves)
- **Closed**: Successfully completed review

This built-in review workflow ensures that each user story progresses through defined states with reviewer assignment and clear accountability. If review fails, user stories return to "Active'' state with the original programmer reassigned and detailed feedback provided through comments and direct communication. This eliminates the common problem where completed work sits idle because no one knows who should review it or when it was completed.

AUTONOMOUS WORK

A key advantage of our system is that programmers may self-select work based on their expertise rather than waiting for assignments. In practice this is performed by setting up a query on Azure DevOps gathered open user stories across the kanban boards of all the kanban boards.
The query integrates directly to the boards and follows the same principles. Selecting the user stories in the query allows direct editing of the user stories, making it easy to tell the content at a glance as well as assigning programmers. The parent feature and area path helps guide the programmers which features they may be suited for, and user stories can either be self-assigned, or may be assigned by a manager/ scrum master, e.g. at a sprint meeting, and then started by the programmer at their own leisure.
An example of the interface can be seen in figure 4

The combination of clear task descriptions, priority coding, and real-time visibility enables team members to make informed decisions about work selection while maintaining overall project coordination and helps eliminate the traditional bottleneck where managers become overwhelmed with task assignment decisions, particularly during high-volume periods. Day-to-day task management is complemented by medium-term resource planning.

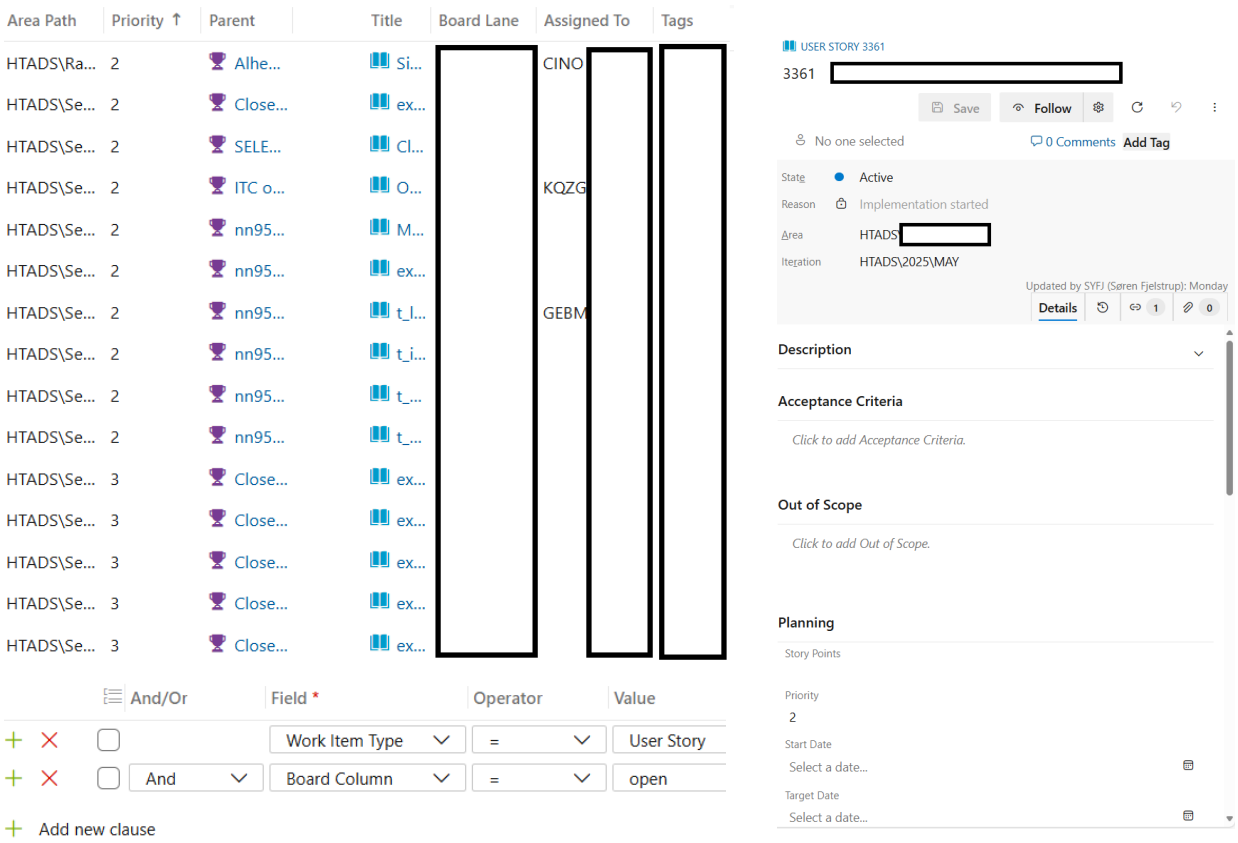

Figure 4 The query interface displays key information such as Area Path, Priority levels, Parent features, Titles, Board Lanes The right panel shows detailed work item. The filter criteria used for making the query is shown at the bottom

MEDIUM-TERM PLANNING: AUTOMATED RESOURCE MANAGEMENT

Our medium-term planning uses a specialized R-Shiny app that automatically retrieves data from Azure DevOps Kanban boards, ensuring resource planning is based on current project data rather than manual estimates. This application serves as the critical bridge between day-to-day task execution (short-term planning) and strategic organizational goals, providing real-time visibility into resource allocation and capacity constraints across multiple time horizons. A picture of the interface can be seen in figure 5.

CORE FUNCTIONALITY AND DATA INTEGRATION

The medium-term planning tool operates on a pull-based architecture that queries Azure DevOps APIs to extract work item data and associated metadata. The application provides automatic retrieval and processing of work item relationships, effort estimates, and timeline data to generate resource forecasts. Unlike traditional project management tools that require manual data entry and updates, our application maintains synchronization with the source of truth: the Azure DevOps boards that teams use daily, thereby eliminating the dual-maintenance burden that often leads to planning resources becoming outdated.

CAPACITY MANAGEMENT AND CONFLICT DETECTION

The system provides immediate capacity assessment through visual indicators giving early warnings when demand exceeds available resources. It also calculates Full-Time Equivalent (FTE) requirements by converting effort estimates into standardized weekly capacity units for easy workload comparison against available staff. The application dynamically adjusts its calculations based on the selected view, users can seamlessly switch between views to examine the same data at different resolutions, facilitating both strategic discussions and tactical problem-solving. This proactive conflict detection enables corrective action such as reprioritizing work, adjusting timelines, or requesting additional resources before stakeholder commitments are made.

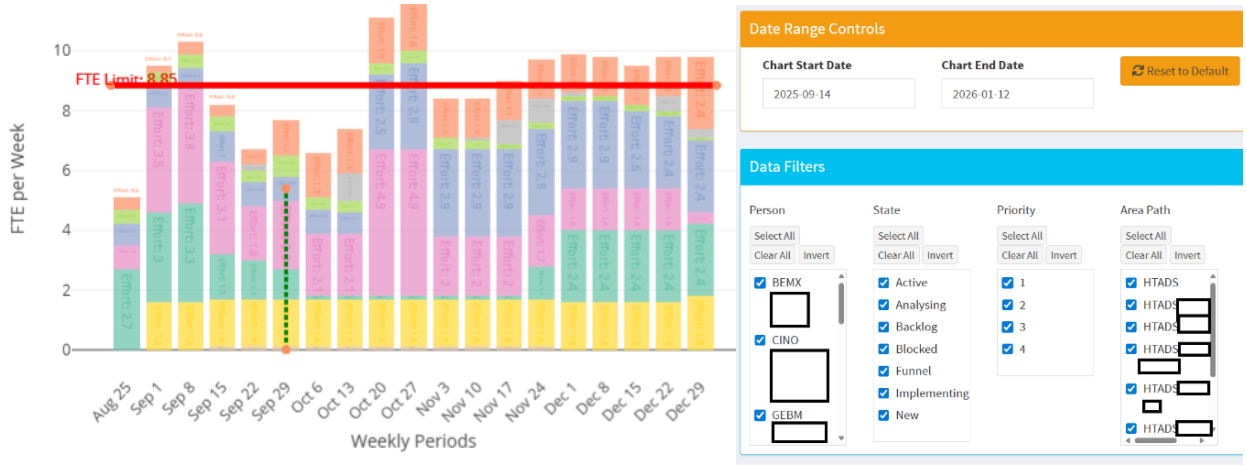

Figure 5: Left: *Weekly FTE allocation visualization showing stacked resource requirements by work item category, with capacity threshold line indicating when demand exceeds available resources (red horizontal line at ~8.85 FTE). Right: Filter controls for selecting work items by person, state, priority, and area path from Azure DevOps.*

BUCKET SYSTEM AND EFFORT ALLOCATION

A bucket system using Azure DevOps “TestedBy" relationships categorizes work items into three distinct types:

Bucket Items: Work items with other items testing them, representing placeholders features or features that encompass multiple sub-components. These items serve as containers that aggregate the effort of their constituent parts.

Constituent Items: Work items that test other items, representing the detailed implementation tasks, test cases, or sub-features that contribute to a bucket's completion. These items contain the granular effort estimates that inform realistic planning.

Standalone Items: Work items with no testing relationships, representing independent deliverables that don't participate in the bucket hierarchy. These items are planned and tracked individually without aggregation logic.

The bucket system prevents double-counting using the following calculation:

Effective effort = max(bucket_effort, ∑ (overlapping_constituents))

This formula ensures that when a bucket and its constituents overlap in time, the system counts only the larger of the two values, either the bucket's own effort estimate or the sum of its constituents' efforts during that period. This approach provides flexibility: Teams can estimate at the bucket level for early planning when details are uncertain, then refine with constituent-level estimates as implementation approaches, without inflating the total effort. This is also outlined in figure 6

DYNAMIC EFFORT PROFILLING WITH TAG NOTATION

Using tag notation like [X_X_X], teams can dynamically redistribute effort across time periods for front-loading critical tasks or scheduling reduced-intensity periods. This recognizes that work rarely proceeds at a constant pace throughout its duration. Instead, many tasks follow predictable patterns: intensive initial research or setup, sustained implementation effort, and lighter validation or documentation phases.

The effort profile syntax uses underscore-separated integers within square brackets to define relative effort weights across time segments. For example, a tag of [2_3_1] on a six-week task indicates that the first two weeks should receive 2/6 of the total effort, the middle two weeks should receive 3/6, and the final two weeks should receive 1/6. The application automatically normalizes these weights and distributes effort accordingly across the item's timeline.

This capability enables several important planning scenarios:

Front-loading: Front loaded tasks can be tagged with profiles like [3_2_1] to ensure adequate resources are allocated early, reducing schedule risk.

Managing review cycles: Deliverables requiring extensive stakeholder review can use profiles like [3_1_2] to allocate effort for initial development, a lighter review period, and then rework based on feedback.

Accommodating resource conflicts: In the case that it is discovered that resources are not available in a specific period, it is easy to move the allocation without changing the deadlines such as profiles like [1_0_1_1] to reflect that the project will have to do without resources for a period

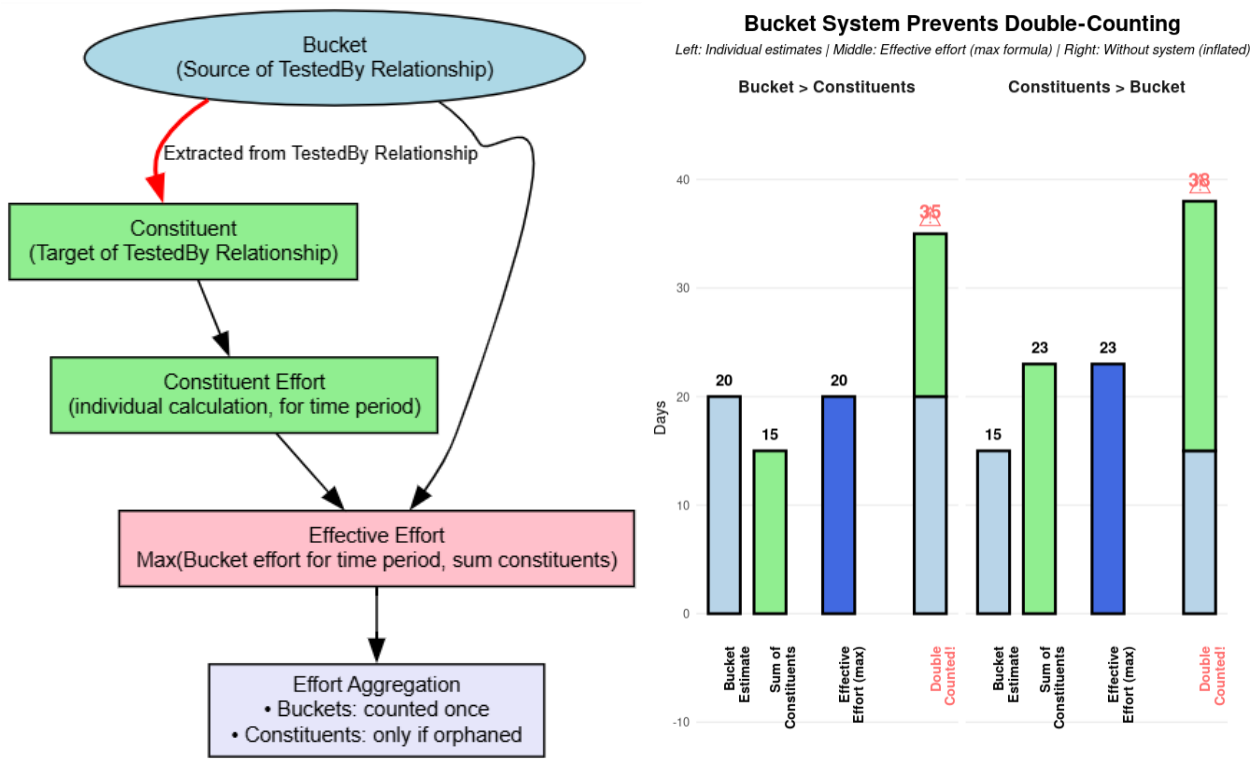

Figure 6: Demonstration of the bucket system for hierarchical effort estimation. Left: relationship between buckets (high-level items) and constituents (sub-tasks). Right: Two scenarios showing how effective effort is calculated as max(bucket estimate, sum of constituents). Without this system, naively adding both bucket and constituent estimates leads to double-counting (rightmost bars: 35 and 38 days vs. correct values of 20 and 23 days).

MULTI-DIMENSIONAL FILTERING AND ANALYSIS

The medium-term planning tool provides comprehensive filtering capabilities that enable managers to analyse resource allocation from multiple perspectives. Users can filter and colour by person (assignee), state (workflow status), priority level, and area path (organizational unit), examples of how this can be utilized could be the colour-by-person view revealing individual workload distribution, helping managers identify over-allocated or under-utilized team members or the colour-by-priority view exposing whether high-priority work is receiving adequate resources, while the colour-by-area-path view facilitates cross-team coordination.

The application also includes a text-based title filter that performs case-insensitive substring matching, allowing managers to quickly isolate work items related to specific features, customers, or technical components. All filters operate in combination, enabling complex queries like "show me all high-priority items assigned to a specific data science team that are currently in the implementing state."

TRANSPARENCY AND TRACEABILITY

The application includes a comprehensive classified features table that displays all work items contributing to the resource forecast, along with their bucket relationships, effort estimates, and chart contribution values enabling drill-down analysis to understand specific resource allocations and supports quality assurance by making it easy to spot data entry errors or inconsistent estimates. Clickable links connect each work item directly to its Azure DevOps page, enabling seamless navigation from planning analysis to detailed task information without context-switching overhead. The system's ability to process hundreds of work items across multiple projects in seconds makes it practical to maintain up-to-date resource forecasts even in large organizations without requiring proportional increases in planning overhead. Thus, allowing the programmers to focus on what they are good at doing, statistical analysis.

MODULAR STANDARD PROGRAMS

Our system employs reusable templates that handle common requests through configuration rather than custom coding. Teams simply update parameters in tested, reliable modules, reducing both development overhead and review requirements. Rather than having one big analysis program, we utilize multiple specialized programs. The design enables parallel execution of different components, accelerating overall processing and reducing bottlenecks.

This modular approach processes 4/5 of routine requests quickly and consistently through standardized workflows, while still allowing full customization for the remaining fifth that require specialized attention. Standard tasks utilize established code pipelines with minimal modifications, accepting standardized inputs such as specification files in csv

(in many cases produced by the AI translation tool), while custom programs are developed for unique requirements that demand specialized attention. By shifting development effort toward complex, unique analyses rather than rebuilding standard functionality, teams can focus their expertise where it matters most while maintaining efficiency and consistency for routine work. An example of the workflow in such an analysis is in figure 7.
The final hurdle after producing analysis is that regulatory compliance requires comprehensive documentation.

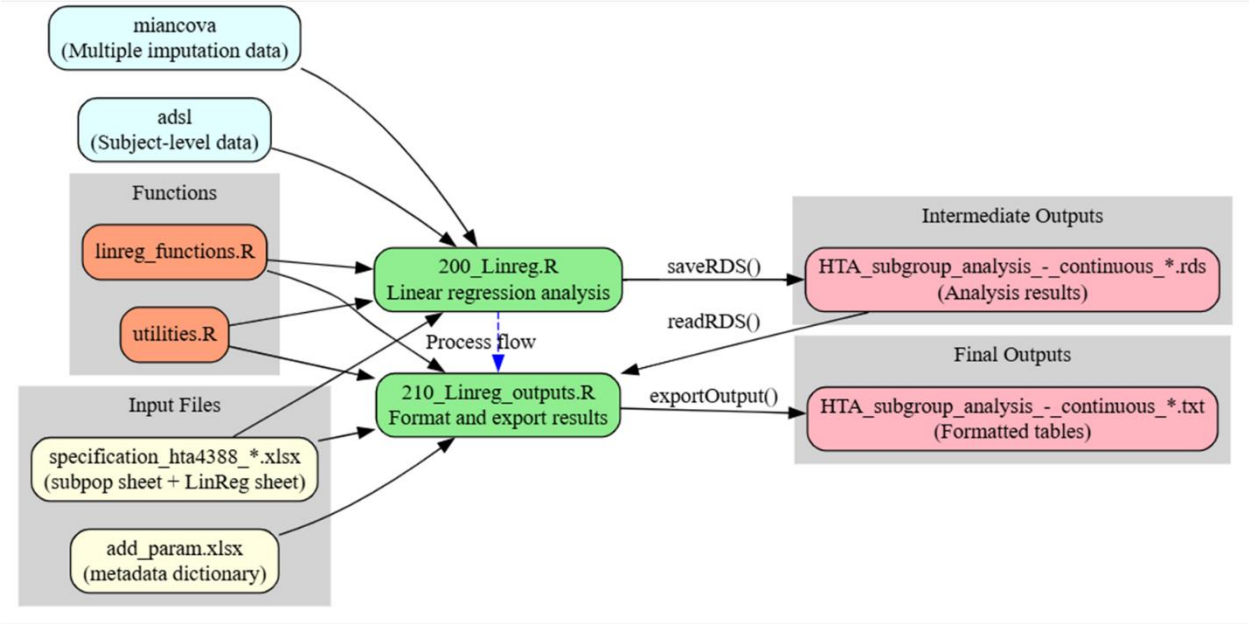

Figure 7:Workflow diagram for linear regression analysis of subgroup data for trial 4388. The analysis pipeline integrates multiple data sources with custom R functions (and configuration files. The workflow consists of two main processing steps and exports the results as formatted tables in .txt format.

AUTOMATIED DOCUMENTATION SYSTEM

In the pharma environment, documentation is not merely administrative overhead but a critical compliance requirement. Governmental bodies require complete, traceable records of analytical decisions, particularly when analyses inform pricing, reimbursement, or post-approval commitments. Our automated documentation system was designed specifically to meet these stringent regulatory requirements while eliminating the burden of duplicate manual entry.

Regulatory agencies require documentation that demonstrates:

- **Traceability**: Clear linkage from initial request through analytical decisions to final delivery
- **Completeness**: Comprehensive capture of all decisions, changes, and rationale
- **Timeliness**: Contemporary documentation created during the work, not retrospectively
- **Accessibility**: Searchable, retrievable records that can be produced during inspections
- **Integrity**: Tamper-evident records with clear audit trails of any modifications

Traditional manual documentation systems struggle to meet these requirements, particularly under the time pressure of urgent agency requests. Documentation is often incomplete, created after-the-fact, or scattered across email chains and personal notes which carries significant compliance risk.
Our system addresses these challenges through automated scraping of Azure DevOps Kanban board history, which captures every action, comment, and status change. This creates a comprehensive, contemporaneous audit trail without requiring programmers to duplicate their work in separate documentation systems.

Critically, this documentation is created automatically as work progresses rather than reconstructed afterward. When an inspector asks “Why was this analytical approach chosen?", the system can immediately produce the complete discussion thread with timestamps, participants, and final decision rationale.
An example of what this looks like in practice can be seen in figure 8

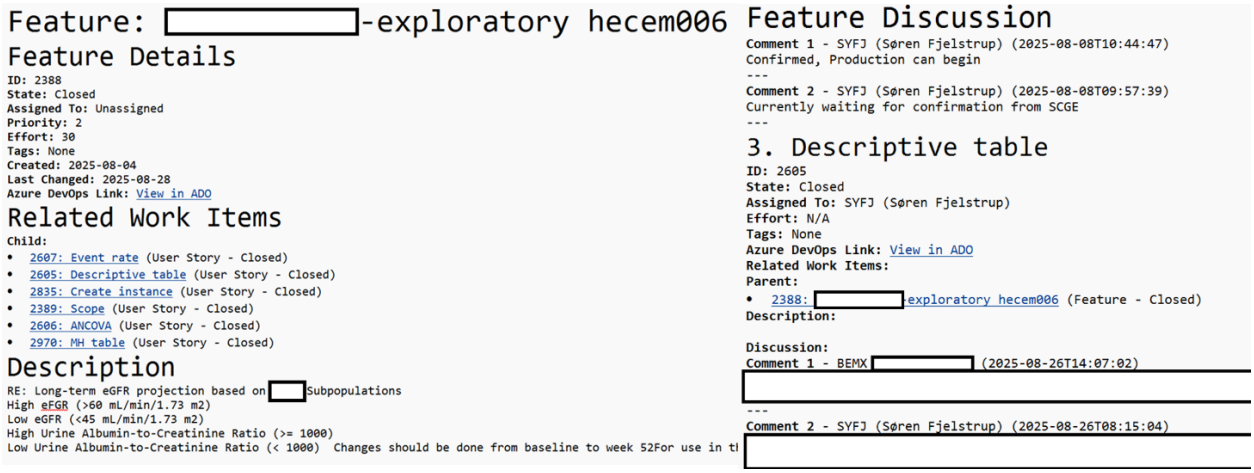

Figure 8: Feature overview showing work item metadata, related child user stories, and description of the exploratory analysis, showing timestamped comments, decision confirmation, and traceability to the parent feature through Azure DevOps linking, documentation Automatically captured from Azure DevOps Kanban board history. The system preserves the complete audit trail including feature specifications, discussion threads with timestamps and participants and hierarchical relationships between work items.

A REAL-WORLD IMPLEMENTATION EXAMPLE

To illustrate the system in action, consider this example workflow:

Step 1: Request Submission: A UK affiliate requests survival analysis for a drug using the single Excel form with a plain text description stating the clear business need in natural language.

Step 2: AI-Assisted Scoping: The LLM automatically generates dataset references, analysis parameters, comparison groups, output formats, and statistical methods. The request enters the "Analysis" state in Azure DevOps (Since it is Clinical Data Scientist initiated).

Step 3: Feature Creation: A feature is created following naming convention: "*ex9536-4388-exploratory (TRIAL A): HEGBR_001*" with appropriate effort estimates and priority coding (Red for urgent regulatory deadline).

Step 4: User Story Decomposition: The feature is broken into user stories: "Make instance", "Scope request", "Generate survival curves", "Statistical analysis", "Final delivery".

Step 5: Autonomous Assignment: Programmers self-select work from the "Backlog".' and see an urgent request that is therefore priority

Step 6: Parallel Execution: Standard modules execute with AI-generated specifications while custom survival analysis components are developed in parallel. The medium-term planning app shows resource allocation and identifies any potential conflicts.

Step 7: Quality Assurance: Completed user stories move to "Ready for Review",' then "Reviewing" with assigned reviewers, ensuring quality before delivery.

Step 8: Automated Documentation: The system captures everything: all discussions and decisions, timeline and status changes, complete history linked to the original request, and comprehensive delivery documentation.

QUANTIFYING ORGANIZATIONAL IMPACT

To demonstrate real-world benefits, we analysed email communications before and after system implementation, focusing on the lead analyst's mailbox who coordinates all client deliveries. This approach provides a consistent measurement baseline while capturing organizational coordination patterns, though it likely understates total benefits by excluding peer-to-peer coordination not involving the lead analyst.

Total coordination overhead per email thread decreased from 5.0 to 2.4 person-hours (52% reduction). The most dramatic improvements occurred in areas where the system provides explicit structure: confusion resolution (100% reduction), clarification rounds (87%), and meeting coordination (78%). Although the pre-period is much longer than the post (656 days vs 168 days), the post-implementation period includes a Q&A delivery, which typically requires more coordination than standard analyses, suggesting these improvements are robust across delivery types. A full table of the data used is available in Supplementary table 1, along with the keywords used in supplementary table 2

More importantly, the composition of coordination fundamentally changed as shown in Figure 9. In the traditional workflow, time was dominated by reactive, interrupt-driven activities, such as resolving confusion and coordinating meetings. In the new system, these categories nearly disappear, with remaining effort focused on predictable refinement work. These improvements stem from how components reinforce each other as an integrated system. The near-elimination of confusion (100% reduction) results from the combination of structured intake (capturing requirements upfront), AI specification (providing complete blueprints), and Azure DevOps visibility (making status transparent), rather than any single component in isolation. As new projects and boards are added, they

automatically flow into the same intake, planning, and documentation logic, eliminating manual synchronization across teams.

This transformation explains why we can now reliably deliver complex packages in hours rather than days while maintaining comprehensive documentation and quality standards. When urgent governmental requests arrive, the system exposes available capacity, surfaces qualified assignees, and produces reliable timelines. The measured 52% reduction represents only the visible portion of organizational transformation, prevented problems that leave no email trail (meetings that don't happen, confusion that doesn't arise, context that doesn't get lost) may even suggest total benefits may exceed quantified improvements.

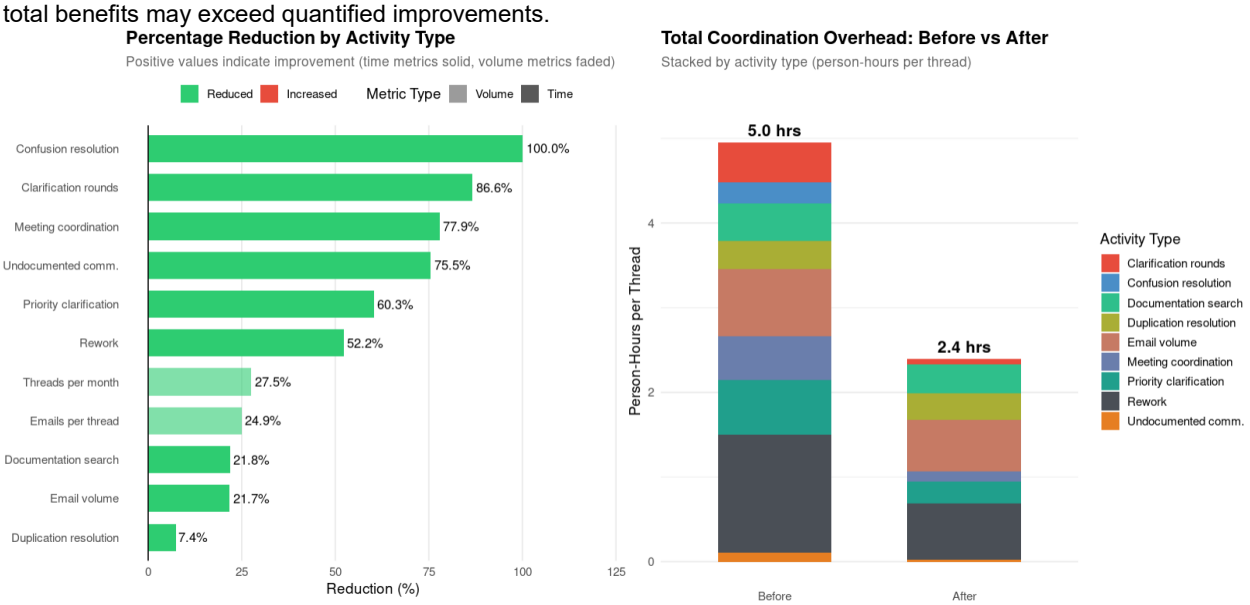

Figure 9:Percentage reduction in coordination time across activity categories. Activities ordered by improvement magnitude. Email analysis based on natural language processing of actual Outlook communications, with time estimates applied to detected patterns (5-30 min per activity type) and multiplied by number of people involved (mean 2.8-2.9 per thread).

CONCLUSION

Our integrated delivery pipeline demonstrates that thoughtful technology integration across multiple planning horizons can transform chaotic, reactive processes into smooth, predictable workflows. The success lies not in any single technology, but in how components reinforce each other: standardized intake prevents confusion, AI specification eliminates clarification cycles, and Azure DevOps visibility reduces meeting needs.

Organizations facing similar challenges can adapt these principles to their own contexts, particularly the integration of short-term task management with medium-term resource planning and the automation of data flow between planning levels. Teams can now respond to urgent regulatory requests within hours rather than days while maintaining quality and traceability, a capability that directly impacts time-to-market for critical therapies.

ACKNOWLEDGEMENTS

A thanks goes to Anders Bo Bojesen and Ulla Brasch Mogensen for collaboration on designing and making the modular programs. Pepa Polavieja for helping design the Medium term planning tool. Claude Sonnet 4.5 (Anthropic, 2025), was used for code creation and initial draft of paper from transcript of presentation. Gemini 2.5 flash image Google DeepMind, 2025) were used for figure generation in presentation.

CONTACT INFORMATION

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

Author Name: Søren Fjelstrup
Company: Novo Nordisk A/S
Email: syfj@novonordisk.com

REFERENCES

Anthropic. (2025). Claude 4.5 Sonnet (20 October version) [Large language model]. https://claude.ai
Google DeepMind. (2025). Gemini 2.5 flash (20 October version) [Large language model]. https://gemini.google.com

SUPPLEMENTARY

Supplementary Table 1: Email Volume and Coordination Overhead Changes: Note: Before = 656 days (93.7 weeks); After = 168 days (24.0 weeks). Weight = estimated hours per occurrence.

A. EMAIL VOLUME METRICS					
	Before	After	Change	% Change	
Total email threads	70	13	-	-	
Total emails	660	92	-	-	
Emails per month	30.6	16.7	-13.9	-45%	
Threads per month	3.3	2.4	-0.9	-27%	
Mean emails per thread	9.4	7.1	-2.3	-25%	
B. COORDINATION OVERHEAD (mean person-hours per thread)					
	Before	After	Δ	% Reduction	Weight (hrs)
Confusion resolution	0.25	0.00	0.25	100%	0.10
Clarification rounds	0.48	0.06	0.41	87%	0.06
Meetings	0.52	0.12	0.41	78%	0.50
Undocumented communication	0.11	0.03	0.08	76%	0.75
Priority clarification	0.65	0.26	0.39	60%	0.60
Rework	1.40	0.67	0.73	52%	1.00
Email reading/writing	0.79	0.62	0.17	22%	0.10
Documentation search	0.44	0.35	0.10	22%	0.25
Duplication resolution	0.33	0.31	0.02	7%	0.50
TOTAL OVERHEAD	4.96	2.40	2.56	52%	

Supplementary Table 2: Keyword Patterns for Coordination Issue Detection. Patterns with .* are regular expressions. Bullet points (•) separate individual keywords.

Category	Keywords
Clarification rounds (0.06 hr)	what do you mean • can you clarify • i don't understand • could you explain • not clear • unclear • confused about • what does.*mean • can you elaborate
Confusion resolution (0.10 hr)	I thought • I assumed • misunderstood • mixed up • my understanding was • wait, so • hold on
Meetings (0.50 hr)	let's meet • schedule.*meeting • can we meet • meeting to discuss • hop on a call • quick call • sync up • touch base
Duplication resolution (0.50 hr)	already.*working on • duplicate • overlap • someone else.*doing • redundant • also working on
Priority clarification (0.60 hr)	what.*priority • which.*first • urgent • asap • high priority • when.*need • deadline • how soon
Documentation search (0.25 hr)	where.*document • can't find • where is • link to • where.*stored • which folder • Sharepoint • where.*file
Rework (1.00 hr)	redo • start over • do.*again • rework • rewrite • change.*approach • go back • different direction
Undocumented communication (0.75 hr)	offline • side conversation • talked about • discussed.*separately • mentioned.*earlier • as we discussed • per our conversation