# **Architecting Intelligent Legacy Summarization: A Guide to Prompting LLMs for ADABAS Natural Programs**

## **Section 1: Introduction: Bridging Decades with AI**

### **The Enduring Legacy and Growing Challenge**

For over half a century, the ADABAS database and its companion fourth-generation language (4GL), Natural, have served as the unsung workhorses of global enterprise. These technologies power high-performance, mission-critical applications for many of the world's leading organizations, from government agencies and financial institutions to manufacturers and retailers.1 Their longevity is a testament to their robustness and efficiency in high-volume, high-transaction online processing environments.4

However, this enduring legacy now presents a formidable business challenge. Enterprises reliant on these systems face a dual pressure: the escalating costs of software licensing and, more critically, a rapidly shrinking talent pool of developers proficient in these specialized technologies.5 This scarcity of expertise creates significant operational risk, hindering the ability to maintain, enhance, and integrate these core systems with modern platforms. The business logic that confers a "distinct competitive advantage" 1 is effectively locked within a syntax that fewer and fewer developers can comprehend.

### **The Promise of AI-Driven Modernization**

Large Language Models (LLMs) offer a transformative solution to this dilemma. By leveraging their advanced code comprehension and generation capabilities, organizations can augment their diminished human expertise, enabling a new generation of developers to understand and interact with legacy ADABAS Natural code. This report presents a methodology that positions LLMs not merely as a documentation tool, but as a strategic catalyst for broader modernization initiatives, including application refactoring, migration to cloud architectures like Microsoft Azure or AWS, and the API-enablement of core business functions.3

The core of this report provides a master-level guide to this process, delivering three key artifacts:

1. A prompt designed for a generalist model like OpenAI's GPT-4.1 to produce a business-oriented summary of a Natural program.
2. A second, distinct prompt tailored specifically for a specialized reasoning model like OpenAI's o3, leveraging its unique architecture.
3. A third prompt that uses an LLM to perform quality assurance, evaluating the generated summaries against a rigorous, business-focused rubric.

The challenge of maintaining these legacy systems can be viewed not just as an exercise in managing technical debt, but as an opportunity to unlock a significant strategic asset. These applications are repositories of decades of refined, battle-tested business logic. The primary business problem is that this asset is illiquid, trapped in a format understood by a dwindling few.5 An LLM-generated summary acts as a "Rosetta Stone," translating this logic from the technical language of Natural into the universal language of business.8 Once this knowledge is made explicit and accessible, it can be analyzed by business stakeholders, used to de-risk modernization projects, and serve as a blueprint for future innovation. The return on investment for this endeavor is measured not just in reduced documentation time, but in the newfound ability to leverage an organization's most valuable intellectual property for its next generation of digital services.

## **Section 2: Deconstructing the ADABAS Natural Program for Automated Analysis**

To effectively instruct an LLM to summarize a Natural program, one must first provide the model with a semantic understanding of the language's structure and key statements. This section serves as a primer on the essential elements of a Natural program, mapping its syntactic components to their business functions to create a conceptual framework for automated analysis.

### **The Anatomy of a Natural Program: A Structured Overview**

Natural offers two distinct programming modes: Reporting Mode and Structured Mode. While Reporting Mode provides flexibility, Structured Mode is far more suitable for automated analysis because it enforces a clearer, more predictable program structure.11 This report will focus on Structured Mode programs, which are easier to read, maintain, and, consequently, parse by an LLM.

A typical Structured Mode program has a clear logical hierarchy. It begins with a DEFINE DATA block, where all data variables and database views are declared in a central location. This is followed by the main block of procedural code, which contains the program's logic. The entire program is terminated with an END statement.4 The Natural compiler translates this source code into a platform-independent byte code, which is then executed by a runtime nucleus.12 This architecture means the source code itself is the definitive representation of the program's logic, making it the ideal input for LLM analysis.

### **Interpreting Data Definitions: The Business Context**

The DEFINE DATA block is the first and most critical section for an LLM to analyze, as it defines the "nouns" of the program—the data it will manipulate.

* **Data Definition Modules (DDMs):** A crucial concept for an LLM to grasp is the DDM. DDMs are the essential bridge between a Natural program and the ADABAS database. They function as a logical "view" of a physical ADABAS file, mapping short, often cryptic, two-character field names (e.g., CU) to longer, descriptive business names (e.g., CUSTOMER).4 Instructing the LLM to prioritize DDM names is paramount for generating a business-oriented summary.
* **Data Types and Variables:** Natural is a strictly typed language, meaning every variable must be defined before use.12 An LLM must recognize common data types—such as Alphanumeric (A), Numeric (N), Packed (P), and Date (D) 4—to infer the nature of the data. For instance, a numeric field likely represents a quantity, counter, or financial value.
* **Special Data Structures:** ADABAS and Natural handle data denormalization through unique structures called "multiple-value fields" (MU) and "periodic groups" (PE).4 An MU field allows a single record to contain multiple values for that field (e.g., multiple phone numbers for one customer). A PE group allows a set of fields to repeat within a single record (e.g., multiple line items in an invoice). Recognizing these structures is key to correctly interpreting data relationships that would exist in separate tables in a standard relational model.

### **Core Logic and Control Flow: Translating Verbs into Business Rules**

The procedural section of the program contains the "verbs"—the actions and business rules.

* **Conditional Logic:** Statements like IF...THEN...ELSE and DECIDE ON...VALUE represent the core decision points and business rules within the application.12
* **Loops and Repetition:** REPEAT...END-REPEAT and FOR loops define iterative processing, which is most often used to process a set of records retrieved from the database.12
* **Program Termination and Flow Control:** The ESCAPE BOTTOM statement is a common construct used to exit a processing loop based on a specific condition, such as a user entering a special character (e.g., a period) to signal the end of their query.12 This represents a user-driven exit path or a logical stop condition within a business process.

### **Database Interaction: The Program's Heartbeat**

The heart of most Natural programs is their interaction with the ADABAS database.

* **Reading Data:** The primary statements for data retrieval are READ and FIND. READ typically processes records sequentially based on a sorted key (a "descriptor"), while FIND performs a direct lookup using an indexed descriptor for greater efficiency.4 An LLM must identify which file is being accessed (via its DDM name) and the criteria for selection (e.g.,  
  BY NAME, WITH PERSONNEL-ID = #ID) to understand the program's main data retrieval function.12
* **Modifying Data:** The statements UPDATE, DELETE, and STORE (for creating new records) represent the primary data modification actions the program can take. These are the verbs that directly change the state of business data.
* **Transaction Control:** The END TRANSACTION and BACKOUT TRANSACTION statements are critical for ensuring data integrity. END TRANSACTION commits all database modifications made during a logical unit of work, making them permanent. BACKOUT TRANSACTION reverts any changes, effectively canceling the operation.16 Recognizing these statements allows an LLM to understand the atomicity of the business operation being performed (e.g., creating a complete customer order).

The following table provides a semantic bridge, mapping key Natural statements to their likely business interpretation, which is essential for guiding an LLM toward a business-centric analysis.

Table 1: Key ADABAS Natural Statements and their Business Significance

| Natural Statement/Construct | Technical Function | Likely Business Interpretation | Example Snippet Context |

| :--- | :--- | :--- | :--- |

| DEFINE DATA LOCAL 1 #NAME (A20) | Defines a local alphanumeric variable. | Declaring a data field for user input or temporary storage, e.g., "Customer Name". | 12 |

| READ EMPLOYEES-VIEW BY NAME | Sequentially reads records from a file, ordered by a descriptor. | Retrieving a list of entities based on a specific criterion, e.g., "Fetches employee records sorted by last name". | 12 |

| FIND EMPLOYEES-VIEW WITH NAME = #NAME | Finds specific records matching a descriptor value. | Searching for a specific entity or set of entities, e.g., "Looks up an employee by their specific name". | 4 |

| IF #INPUT = '.' THEN ESCAPE BOTTOM | Checks for a specific input value to exit a loop. | Implementing a user-controlled exit from a function, e.g., "Allows the user to terminate the search by entering a period". | 12 |

| UPDATE | Modifies a record that was previously read. | Applying changes to an existing record, e.g., "Saves updates to an employee's information". | 16 |

| END TRANSACTION | Commits all database changes since the last commit. | Finalizing a business operation and making the data changes permanent, e.g., "Completes the customer order transaction". | 16 |

| BACKOUT TRANSACTION | Reverts all database changes since the last commit. | Canceling a business operation and undoing any partial data changes, e.g., "Rolls back the transaction if an error occurs". | 16 |

## **Section 3: Defining the Target Output: The Effective Business-Oriented Summary**

To guide an LLM effectively, it is crucial to first define the "gold standard" for the desired output. A high-quality, business-oriented summary is not a literal translation of code but a synthesized document that communicates function and purpose to a non-technical audience. This section establishes the characteristics of such a summary, which will serve as the target for the prompt engineering process.

### **The Four Pillars of a Business Summary**

Drawing from the principles of effective executive summaries, the target output must be structured to answer four fundamental questions for the reader 17:

1. **Purpose (The "Why"):** What is the primary business problem this program solves? What is its core function in a single, clear statement? (e.g., "This program allows an HR administrator to look up and display employee leave balances based on a name search.")
2. **Process (The "How"):** What are the high-level steps the program executes to achieve its purpose? This should be a simplified workflow, not a line-by-line algorithmic description. (e.g., "It prompts the user for an employee's last name, retrieves the matching records from the database, and displays their name, department, and available leave days.")
3. **Data (The "What"):** What are the key business entities and data points the program interacts with? (e.g., "The program accesses the Employee master file, using the employee's name, department, and leave-due fields.")
4. **Dependencies & Interactions (The "Where"):** What is the program's operational context? Does it rely on other programs, systems, or databases? (e.g., "This is an online, interactive program that directly queries the production ADABAS database. It does not call any external sub-programs.")

### **Audience-Centric Language**

The summary must be written for a business analyst, project manager, or executive, not a developer.8 This requires a deliberate translation of technical terminology into its business equivalent. For example:

* **Technical Description:** "The program executes a READ loop on the EMPLOYEES-VIEW DDM, using the NAME descriptor, and populates a screen with the DEPT and LEAVE-DUE fields."
* **Business-Oriented Summary:** "The program retrieves a list of employees, sorted by name, and displays their department and outstanding leave days."

### **Moving Beyond Line-by-Line Translation to Holistic Function**

The ultimate goal is to describe the program's emergent business function, not its specific implementation details.10 The summary should capture the program's role within a larger business process, such as finance, human resources, or inventory management.1 A successful summary is concise and provides enough information for a manager to understand the program's relevance and make a decision without ever needing to read the source code.19

This structured approach to summarization offers value far beyond simple documentation. A high-quality, LLM-generated business summary can serve as a reverse-engineered artifact for modern software development paradigms like Business-Driven Development (BDD) or Domain-Driven Design (DDD). The process of creating such a summary effectively extracts the "ubiquitous language" and identifies the "bounded context" from the legacy code. BDD prioritizes the needs of the business over technology 20, and DDD focuses on modeling the core business domain. A primary challenge in modernizing legacy systems is that this essential business and domain knowledge is often lost, existing only implicitly within the code.

The "Four Pillars" structure maps directly to these modern concepts. The "Purpose" and "Process" help define the business *capabilities* and *use cases*. The "Data" section identifies the core *domain entities* and *aggregates*. Finally, the "Dependencies" section helps to delineate the *bounded context* of the application. By structuring the LLM's output in this manner, the summarization effort generates a foundational document that a modern development team can use to accelerate a BDD/DDD-based refactoring or replacement project. This transforms the activity from technical archaeology into a strategic bridge between the old system and the new.

## **Section 4: A Comparative Analysis of LLM Architectures: GPT-4.1 vs. o3**

A one-size-fits-all approach to prompt engineering is ineffective. The optimal prompt for a given task is highly dependent on the underlying architecture of the LLM. This section provides a deep analysis of the architectural and behavioral differences between a generalist model like GPT-4.1 and a specialized reasoning model like OpenAI's o3, focusing on their implications for the task of summarizing legacy code.

### **GPT-4.1: The Power of Generalization and Explicit Instruction**

* **Model Profile:** GPT-4.1 is a highly capable, general-purpose model with an exceptionally broad knowledge base. Its strength lies in its ability to understand and follow complex, multi-step instructions that are explicitly provided within the prompt.
* **Prompting Paradigm:** The power of a model like GPT-4.1 is unlocked through detailed, scaffolded prompting techniques. This includes methods like "Chain-of-Thought" (CoT) prompting, where the user explicitly guides the model's reasoning process step-by-step. For a complex task, one might need to add supplementary instructions such as "check your answer for consistency with the facts above" to ensure a high-quality output.21
* **Application to Natural:** To summarize Natural code, GPT-4.1 will require a comprehensive prompt that effectively "teaches" it how to interpret the language. The prompt must provide context, define key terms, and lay out a clear, sequential process for analysis and summarization.

### **o3: The Paradigm of the Internal Reasoning Engine**

* **Model Profile:** The o3 model represents a different class of LLM, often referred to as a "reasoning model." It is specifically designed to devote more computational effort and time—a process of "deliberation"—to thinking through a problem before generating an answer.22 This architecture includes a "private chain of thought," an internal process of reasoning that is not exposed in the output unless requested.22
* **Prompting Paradigm Shift:** Research and official guidance on o-series models are unequivocal: for these models, *less is more*. Overly descriptive prompts, few-shot examples, and explicit "think step-by-step" instructions are generally unnecessary and can even degrade performance by interfering with the model's optimized internal reasoning pathways.21 The most effective approach is a clear, minimal, and direct prompt that states the goal and provides the necessary data.
* **Key Capabilities:** The o3 model excels at complex, multi-step logical tasks, demonstrating state-of-the-art performance in domains like coding, mathematics, and science.23 A key differentiator is its native ability to use tools, such as executing Python code or searching the web, to augment its reasoning process.23
* **Application to Natural:** The prompt for o3 should not attempt to teach it the Natural language. Instead, it should provide the source code and relevant schemas as structured data and then issue a direct command to produce a summary according to a precisely defined output format. The task relies on trusting the model's powerful internal engine to deconstruct the code's logic.

### **The "Reasoning Effort" Parameter: A Unique Tuning Knob**

A unique feature of the o-series models is the reasoningEffort parameter, which can be set to low, medium, or high.21 This provides a powerful tuning knob to balance performance, cost, and latency. For analyzing a highly complex Natural program with convoluted business logic and nested loops, setting the effort to

high would be appropriate, allocating more computational resources for the model to unravel the logic. For a simple report-generating program, medium or low would likely suffice, resulting in a faster and more cost-effective response.

The following table starkly contrasts the two prompting methodologies, providing a design specification for the prompts developed in the subsequent sections.

Table 2: Prompting Strategy Comparison: GPT-4.1 vs. o3

| Prompting Dimension | GPT-4.1 (Generalist) | o3 (Reasoning Engine) | Rationale for Difference |

| :--- | :--- | :--- | :--- |

| Core Strategy | Instructive & Guiding | Declarative & Direct | GPT-4.1 needs to be taught the reasoning path. o3 has its own internal, optimized path.21 |

| Chain-of-Thought | Explicitly prompted (e.g., "Think step-by-step"). | Avoid. The model does this internally and prompting can interfere.21 | o3's architecture is built around a "private chain of thought".22 |

| Few-Shot Examples | Often beneficial to demonstrate format and logic. | Minimal to none. Can degrade performance.21 | o3 is trained to not require examples; they can confuse its internal logic.21 |

| Prompt Length/Verbosity | Can be long and descriptive to provide full context and instruction. | Keep prompts clear, minimal, and focused on the goal.21 | o3 responds best to focused questions without extraneous text.21 |

| Context Provision | Provide context as needed. | Crucial to provide all necessary domain-specific data (e.g., DDM schemas) as its knowledge base may be narrower.21 | o3 is a reasoning specialist, not a knowledge generalist. It reasons on the data you provide. |

| Handling Complexity | Decompose complex problems into smaller steps in the prompt. | Present the complex problem as a single, well-defined task. | o3 is optimized to handle complex, multi-step problems holistically.21 |

| Unique Parameters | Standard API parameters (temperature, top\_p, etc.). | reasoningEffort parameter to tune deliberation time and cost.24 | This is a unique architectural feature of the o-series models. |

## **Section 5: Prompt Engineering for GPT-4.1: A Generalist's Approach to Natural Code**

This section presents a complete, annotated prompt designed for a generalist model like GPT-4.1. The prompt is structured as a comprehensive set of instructions that guides the model through the analysis and summarization process, compensating for its lack of specialized, built-in reasoning for this specific task. Each component is justified based on the principles established in the preceding analysis.

### **The Prompt Structure: A Multi-Part Scaffolding**

The prompt is constructed in four distinct parts, each serving a specific purpose in scaffolding the model's thought process.

Part 1: Persona and Goal Setting

This initial instruction establishes the required expertise and the overall objective, priming the model for the specific context of the task.

You are an expert legacy systems analyst specializing in ADABAS Natural. Your task is to analyze the provided Natural program and produce a concise, business-oriented summary suitable for a non-technical audience, such as a project manager or business analyst.

* **Justification:** Defining a persona helps the model adopt the appropriate tone, vocabulary, and frame of reference. Clearly stating the goal and the target audience ensures the output is aligned with business needs rather than being a purely technical code description.

Part 2: Context Provision

This section provides the raw material for the analysis, with clear placeholders for the user to insert the necessary code and data definitions.

Analyze the following ADABAS Natural program. The source code for the program is provided first, followed by the Data Definition Modules (DDMs) for any database files it accesses.  
  
### PROGRAM SOURCE CODE  
---  
  
---  
  
### DATA DEFINITION MODULES (DDMs)  
---  
  
---

* **Justification:** The model requires the complete source code to perform its analysis. Explicitly separating the program code from the DDM definitions helps the model differentiate between procedural logic and data structures, which is critical for understanding what the program *does* versus what data it *uses*.4

Part 3: Knowledge Priming & Step-by-Step Instructions (Chain-of-Thought)

This is the core of the instructive strategy for GPT-4.1. It explicitly guides the model through a logical analysis process, mimicking how a human analyst would approach the task.

To create the summary, you must follow these steps precisely:  
  
1. \*\*Identify Data Entities:\*\* First, analyze the `DEFINE DATA` section of the program and the provided DDM source code. Create a list of all key data entities (database files) and important data fields. Use the descriptive names from the DDMs for business context.  
2. \*\*Trace Program Logic:\*\* Second, trace the main execution path of the program. Pay close attention to database interaction commands (`READ`, `FIND`, `UPDATE`, `STORE`, `DELETE`), processing loops (`REPEAT`, `FOR`), and conditional logic (`IF`, `DECIDE`).  
3. \*\*Determine Primary Purpose:\*\* Third, based on the program's inputs (e.g., user prompts), its core processing logic, and its outputs (e.g., `DISPLAY`, `WRITE` statements), determine the single primary business purpose of the program.  
4. \*\*Synthesize the Summary:\*\* Finally, synthesize your findings from the previous steps into a summary using the mandatory output format specified below.

* **Justification:** This explicit chain-of-thought instruction is essential for a generalist model. It decomposes the complex task of "summarize this program" into a series of smaller, more manageable sub-tasks, ensuring a methodical and comprehensive analysis.

Part 4: Output Format Definition

This final part enforces a consistent, high-quality output structure that aligns with the definition of an effective business summary.

Present your final summary using the following four-part structure. Use clear, non-technical business language. Do not include technical jargon or code snippets in the final output.  
  
\*\*Purpose:\*\*  
[A single sentence that clearly describes the program's primary business function.]  
  
\*\*Process:\*\*  
[A bulleted list describing the high-level steps the program takes to achieve its purpose. Limit to 3-5 key steps.]  
  
\*\*Data Accessed:\*\*  
  
  
\*\*Program Type & Dependencies:\*\*

* **Justification:** Providing a strict template for the output ensures that the summary contains all the required elements (the "Four Pillars") and is presented in a format that is easy for a business stakeholder to consume and understand.17

## **Section 6: Tailoring Prompts for the o3 Reasoning Engine: A Specialist's Toolkit**

Prompting the o3 reasoning engine requires a fundamental shift in strategy from instructive guidance to declarative command. This section presents a distinct prompt tailored for o3, highlighting and justifying every deviation from the GPT-4.1 version. The design philosophy is to provide the model with perfectly structured data and a precise definition of the desired output, then trust its powerful internal reasoning capabilities to bridge the gap.

### **The Principle of Minimalist Instruction: Why Less is More for o3**

The prompt for o3 is significantly shorter and more direct than its GPT-4.1 counterpart. It deliberately omits the explicit step-by-step instructions, knowledge priming, and verbose explanations. This is because o3 possesses a powerful, built-in reasoning engine that is optimized to deconstruct complex problems holistically. Providing explicit, granular steps is not only redundant but can actively interfere with its native, more efficient processing pathways, potentially degrading the quality of the result.21 The core principle is to trust the engine.

### **The o3 Prompt: A Declarative Request**

The prompt is structured to provide data and specify a target format, leaving the analytical "how" to the model itself.

Part 1: Persona and Goal Setting (Modified)

The persona remains useful for context, but the task is framed as a direct command.

You are an expert legacy systems analyst. Your task is to analyze the provided ADABAS Natural program and its associated Data Definition Module (DDM) schemas.

* **Justification:** This is a concise and direct instruction that sets the professional context without unnecessary verbiage.

Part 2: Structured Data Input

Providing the input data in a clearly structured format is critical for a model that reasons over the provided context.

### Program Source Code

### Data Definition Modules (DDMs)

\* \*\*Justification:\*\* While the instruction is minimal, the context provided must be complete. The o3 model is a reasoning specialist, not necessarily a knowledge generalist, so it relies heavily on the specific data provided in the prompt.[21] Using clear markdown headers (`###`) helps its internal parser distinguish between the different types of input artifacts.  
  
\*\*Part 3: The Direct Command\*\*  
This is the central, declarative instruction that replaces the lengthy step-by-step guidance used for GPT-4.1.

Generate a concise, business-oriented summary of the program. The summary must adhere strictly to the following JSON schema and its descriptions.

\* \*\*Justification:\*\* This is the crucial difference. Instead of telling the model \*how to think\*, the prompt tells it \*what to produce\*. Requesting a structured format like JSON is a robust method for ensuring the output is predictable, consistent, and machine-readable. The advanced coding and reasoning capabilities of o3 make it highly proficient at generating well-formed structured data.[24]  
  
\*\*Part 4: Output Schema Definition\*\*  
Providing an explicit JSON schema is the ultimate form of direct, unambiguous instruction for a reasoning model.  
```json  
{  
 "purpose": "A one-sentence summary of the program's primary business function.",  
 "process\_flow": [  
 "A high-level description of the first main step in the program's workflow.",  
 "A high-level description of the second main step.",  
 "..."  
 ],  
 "data\_entities":,  
 "actions\_performed":  
 }  
 ],  
 "program\_context": {  
 "execution\_type": "Identify as 'Online/Interactive' or 'Batch'.",  
 "dependencies": "List any called sub-programs or other critical system interactions identified in the code. If none, state 'None'."  
 }  
}

* **Justification:** This schema leaves no ambiguity about the desired output structure. It leverages the model's strengths in code and structure generation while freeing its internal reasoning engine to perform the complex task of analyzing the Natural code and mapping its findings to the specified JSON fields. This approach is more robust, scalable, and better aligned with the architectural design of an advanced reasoning model.

## **Section 7: A Framework for Quality Assurance: The LLM Evaluation Prompt**

Generating summaries is only half the task; verifying their quality is equally critical. To build trust and enable scalable implementation, a systematic quality assurance (QA) process is required. This section provides a third prompt designed to leverage an LLM as an impartial evaluator, scoring the generated summaries against a predefined, business-centric rubric.

### **The Evaluator Persona: Establishing Impartiality and Expertise**

The prompt begins by establishing a dual-expertise persona for the evaluator LLM. This is crucial for ensuring the assessment considers both technical accuracy and business relevance.

You are a Quality Assurance analyst with deep expertise in both legacy mainframe systems (specifically ADABAS Natural) and modern business analysis principles. Your task is to perform a rigorous evaluation of an AI-generated program summary. You will compare the provided summary against the original source code and assess its quality based on the specific rubric provided.

* **Justification:** This persona primes the model to act as a discerning judge, capable of understanding the source code's technical nuances while simultaneously evaluating the summary's utility for a non-technical business audience.

### **The Evaluation Prompt Structure**

The prompt is structured to accept all necessary inputs and to guide the LLM through a methodical evaluation process.

### INSTRUCTIONS  
Based on the provided source code and the AI-generated summary, evaluate the summary using the rubric below. For each of the four criteria in the rubric, you must provide a numerical score from 1 (poor) to 5 (excellent) and a brief, clear justification for your score, citing specific examples from the code or summary where appropriate. Conclude with an overall assessment of the summary's quality and provide one or two actionable suggestions for its improvement.  
  
### INPUT 1: ORIGINAL PROGRAM SOURCE CODE & DDMs  
---  
  
---  
  
### INPUT 2: AI-GENERATED SUMMARY  
---  
[Paste the summary that needs to be evaluated here.]  
---  
  
### INPUT 3: EVALUATION RUBRIC  
  
  
### YOUR EVALUATION OUTPUT  
Provide your evaluation in the following format:  
  
\*\*Criterion 1: Factual Accuracy\*\*  
- \*\*Score:\*\* [1-5]  
- \*\*Justification:\*\* [Your reasoning for the score.]  
  
\*\*Criterion 2: Business Relevance\*\*  
- \*\*Score:\*\* [1-5]  
- \*\*Justification:\*\* [Your reasoning for the score.]  
  
\*\*Criterion 3: Completeness\*\*  
- \*\*Score:\*\* [1-5]  
- \*\*Justification:\*\* [Your reasoning for the score.]  
  
\*\*Criterion 4: Conciseness\*\*  
- \*\*Score:\*\* [1-5]  
- \*\*Justification:\*\* [Your reasoning for the score.]  
  
\*\*Overall Assessment & Recommendations:\*\*  
[Your concluding thoughts and suggestions for improvement.]

* **Justification:** This structured approach ensures the evaluation is comprehensive and consistent. By requiring a score and a justification for each criterion, the output provides both quantitative and qualitative feedback, which is invaluable for tuning the generation prompts and identifying systemic weaknesses in the summarization process.

### **Evaluation Rubric for Business-Oriented Summaries**

The following rubric defines the "gold standard" for a quality summary and serves as the core of the evaluation prompt.

Table 3: Evaluation Rubric for Business-Oriented Summaries

| Criterion | Description | Scoring (1-5) |

| :--- | :--- | :--- |

| 1. Factual Accuracy | Does the summary correctly represent the program's logic? Are the database interactions, conditions, and outcomes described accurately, without hallucination or misinterpretation of the source code? | 1: Completely inaccurate. 5: Perfectly accurate reflection of the source code. |

| 2. Business Relevance | Is the summary framed in business terms, not technical jargon? Does it focus on the "why" (business purpose) and the "what" (data entities) rather than just the "how" (implementation details)? | 1: Purely technical description. 5: Clear, concise, and entirely business-focused. |

| 3. Completeness | Does the summary capture all four pillars: Purpose, Process, Data, and Dependencies? Are any major logical branches, data interactions, or error-handling routines missed? | 1: Misses multiple key aspects. 5: Captures all essential components of the program's function. |

| 4. Conciseness | Is the summary brief and to the point? Does it avoid unnecessary detail, verbiage, and repetition while still being complete? | 1: Overly long and verbose. 5: As concise as possible without sacrificing completeness. |

## **Section 8: Strategic Recommendations and Implementation Guidance**

Transforming this prompt engineering methodology from a technical experiment into a strategic enterprise capability requires a thoughtful implementation plan. The following recommendations provide a roadmap for deploying, scaling, and communicating the value of this AI-driven approach to legacy system analysis.

### **Implementing a Human-in-the-Loop (HITL) Workflow**

It is strongly recommended not to rely on a fully automated process, especially during the initial phases of deployment. A Human-in-the-Loop (HITL) workflow is essential for building trust and ensuring accuracy. In this model, the LLM generates the first-pass summary and the initial automated evaluation. A human expert—either one of the few remaining Natural developers or a business analyst trained on the system—then reviews and validates both the summary and its evaluation. This approach leverages AI for the heavy lifting, freeing up scarce human experts to focus on the most critical tasks of validation, refinement, and handling edge cases. This feedback loop is also the primary mechanism for improving the core prompts over time.

### **Scaling the Solution for Enterprise Codebases**

To tackle an enterprise repository containing thousands of Natural programs, manual, one-by-one summarization is not feasible. Automation is key to achieving scale. This involves developing scripts to orchestrate the end-to-end process:

1. **Extraction:** Automatically extract program source code and their corresponding DDMs from legacy library management systems or modern version control repositories like Git.28
2. **API Integration:** Batch-process the extracted artifacts, feeding them to the appropriate LLM API endpoint (e.g., GPT-4.1 or o3).
3. **Storage:** Store the structured summaries (e.g., the JSON output from the o3 prompt) and the corresponding evaluation results in a centralized knowledge base, a Configuration Management Database (CMDB), or a dedicated application portfolio management tool.1 This creates a searchable, enterprise-wide repository of business logic.

### **Continuous Improvement: The Prompt Engineering Lifecycle**

The prompts provided in this report should be treated as living artifacts, not static code. The field of AI is evolving rapidly, and the models themselves are constantly being updated. An effective implementation must include a continuous improvement cycle. The quantitative and qualitative feedback generated by the evaluation prompt (Section 7) and the HITL reviews should be collected and analyzed regularly. This data will reveal patterns in summarization errors or shortcomings, providing the necessary insights to tune and refine both the generation and evaluation prompts to maintain state-of-the-art performance.

### **Communicating Value to Business Stakeholders**

Finally, to secure the necessary long-term support and investment, the project's success must be framed in terms of business outcomes, not technical metrics. Instead of reporting on the "number of programs summarized," success should be communicated through its impact on strategic enterprise goals.3 Key value propositions include:

* **Accelerated Modernization:** Demonstrating how readily available business summaries reduce the analysis phase of migration or refactoring projects.
* **Reduced Operational Risk:** Highlighting how the project mitigates the key-person dependency on a small group of aging experts.
* **Unlocked Business Logic:** Showcasing how the newly accessible business rules are being used to inform the development of new digital products, APIs, and more efficient processes.

By translating technical achievements into tangible business value, this initiative can be correctly positioned as a critical enabler of digital transformation, ensuring its role as a strategic priority for the organization.

#### Works cited

1. Account Executive (Adabas & Natural) - Experteer, accessed June 22, 2025, <https://us.experteer.com/career/view-jobs/account-executive-adabas-and-natural-reston-va-usa-52430665>
2. Account Executive (Adabas & Natural) @ Software AG - Teal, accessed June 22, 2025, <https://www.tealhq.com/job/account-executive-adabas-natural_e30404b5-4554-4eed-9184-894b93abd278?included_keywords=account+management&workplace_types=hybrid&page=57>
3. Adabas & Natural for Application Modernization - Software AG, accessed June 22, 2025, <https://www.softwareag.com/en_corporate/platform/adabas-natural.html>
4. ADABAS - Wikipedia, accessed June 22, 2025, <https://en.wikipedia.org/wiki/ADABAS>
5. Modernize Natural and Adabas Workloads on AWS with IBM ModernSystems accelerator, accessed June 22, 2025, <https://aws.amazon.com/blogs/apn/modernize-natural-and-adabas-workloads-on-aws-with-ibm-modernsystems-accelerator/>
6. Refactor mainframe computer systems that run Adabas & Natural - Azure Architecture Center | Microsoft Learn, accessed June 22, 2025, <https://learn.microsoft.com/en-us/azure/architecture/example-scenario/mainframe/refactor-adabas-aks>
7. Deploying Software AG Adabas and Natural Workloads on AWS, accessed June 22, 2025, <https://aws.amazon.com/blogs/apn/deploying-software-ag-adabas-and-natural-workloads-on-aws/>
8. What Are Business Applications? Benefits, Careers, & More - University of Cincinnati Online, accessed June 22, 2025, <https://online.uc.edu/blog/what-are-business-applications/>
9. Business software - Wikipedia, accessed June 22, 2025, <https://en.wikipedia.org/wiki/Business_software>
10. What Is Business Programming? | North Carolina Central University, accessed June 22, 2025, <https://online.nccu.edu/blog/what-is-business-programming/>
11. Natural Programming Modes - Software AG Documentation, accessed June 22, 2025, <https://documentation.softwareag.com/natural/nat841unx/pg/pg_mode.htm>
12. Getting Started with Adabas and Natural - AWS, accessed June 22, 2025, <https://cdck-file-uploads-global.s3.dualstack.us-west-2.amazonaws.com/techcommunity/original/2X/d/d0f0b927ee6024c8e6b5f5d9aa752cb651b639b5.pdf>
13. Natural System Architecture - Software AG Documentation, accessed June 22, 2025, <https://documentation.softwareag.com/natural/nat828dmf/pdf/arc.pdf>
14. ADABAS Essentials, accessed June 22, 2025, <http://www.math.wpi.edu/saspdf/adbas/c02.pdf>
15. Natural Programming Basic - Software AG Learning Portal, accessed June 22, 2025, <https://learn.softwareag.com/course/info.php?id=1467>
16. Using Natural Statements and System Variables - Software AG Documentation, accessed June 22, 2025, <https://documentation.softwareag.com/natural/nat827dmf/dbms/nsq-natsm.htm>
17. How to Write an Executive Summary, with Examples [2025] - Asana, accessed June 22, 2025, <https://asana.com/resources/executive-summary-examples>
18. Writing an Effective Executive Summary - Vista Projects, accessed June 22, 2025, <https://www.vistaprojects.com/how-to-write-an-effective-executive-summary/>
19. Writing an Executive Summary | UAGC Writing Center, accessed June 22, 2025, <https://writingcenter.uagc.edu/writing-executive-summary>
20. What is Business-Driven Development | BDD? - Functionize, accessed June 22, 2025, <https://www.functionize.com/blog/business-driven-development-what-and-how-functionize-helps>
21. Prompt Engineering for OpenAI's O1 and O3-mini Reasoning ..., accessed June 22, 2025, <https://techcommunity.microsoft.com/blog/azure-ai-services-blog/prompt-engineering-for-openai%E2%80%99s-o1-and-o3-mini-reasoning-models/4374010>
22. OpenAI o3 - Wikipedia, accessed June 22, 2025, <https://en.wikipedia.org/wiki/OpenAI_o3>
23. OpenAI's O3: Features, O1 Comparison, Benchmarks & More ..., accessed June 22, 2025, <https://www.datacamp.com/blog/o3-openai>
24. Guides: Get started with OpenAI o3-mini - AI SDK, accessed June 22, 2025, <https://sdk.vercel.ai/docs/guides/o3>
25. ChatGPT OpenAI o3 and o4‑mini models FAQ (Enterprise & Edu Version), accessed June 22, 2025, <https://help.openai.com/en/articles/9855712-chatgpt-openai-o3-and-o4-mini-models-faq-enterprise-edu-version>
26. www.techtarget.com, accessed June 22, 2025, <https://www.techtarget.com/whatis/feature/OpenAI-o3-explained-Everything-you-need-to-know#:~:text=The%20o3%2Dpro%20model%20also,require%20multiple%20capabilities%20working%20together.>
27. OpenAI o3 and o4-mini System Card, accessed June 22, 2025, <https://openai.com/index/o3-o4-mini-system-card/>
28. Natural ONE Tutorial - First Steps - YouTube, accessed June 22, 2025, <https://www.youtube.com/watch?v=-KmyxK7M1d4>