RAID

REBTECH AI DATA ENGINEERING

EXECUTIVE SUMMARY

Developing modern data platforms faces a fundamental challenge: the complexity of building robust, scalable solutions requires both broad technical expertise and a deep understanding of best practices. Traditionally, this has meant long development times, high costs, and the risk of inconsistent quality.

AI Data Engineering is a methodology that combines generative AI with structured systems development to address this challenge. By documenting decades of experience in a framework of specialized AI Skills, data platforms can be developed with consistently high quality, dramatically shorter lead times, and lower risk of technical debt.

THE CHALLENGE: COMPLEXITY IN MODERN DATA PLATFORM DEVELOPMENT

A production-ready data platform is a complex system that must be scalable for growing data volumes, secure with proper access control, and robust enough to handle data quality issues. At the same time, it must be flexible enough to support different types of analysis and well-structured enough for long-term maintainability.

When complexity is high, recurring problems arise. Development time becomes long, quality uneven, and technical debt accumulates as time pressure leads to compromises. Particularly problematic is that data quality assurance, access control, and testing are often deprioritized in early phases, creating security risks and quality issues further down the line.

Traditional development methods rely on human developers writing all code manually. Writing high-quality code requires a fundamentally different level of competence and discipline than writing low-quality code, and it takes considerably longer. In practice, this leads to quality aspects being sacrificed in order to meet schedules and budgets. Another problem is that issues are often discovered late, when large parts of the code have already been written, resulting in technical debt from the very beginning.

GENERATIVE AI AS AN ENABLER

Generative AI has a unique property that fundamentally changes the economics of systems development: it takes just as long to generate complex, high-quality code as it does to generate simple code. When AI is instructed to build a data pipeline, it makes little difference whether the instruction also includes “also add data quality checks, unit tests, and documentation” or not. This breaks the traditional trade-off between quality and time.

Generative AI is, however, not a universal solution. AI performs best when problems are well bounded and clearly formulated. Instructing AI to “build a data platform” produces poor results because the task is too broad. For AI to consistently deliver high quality, the development work must be broken down into appropriately sized, well-defined tasks with clear instructions and verifiable results.

RAID: METHODOLOGY AND FRAMEWORK

Rebtech AI Data Engineering, RAID, combines a spec-driven development methodology with a framework of specialized AI Skills. Projects begin by articulating what the platform must do — its technical, functional, and non-functional requirements — before any implementation begins. The specification is documented in a structured form that both humans and AI can act on, and remains the authoritative reference throughout development.

Once the specification is in place, the RAID framework takes over. A set of specialized AI Skills, each representing a specific competence and instructed with best practices from production systems, collaborates to translate the specification into a working data platform with consistent quality.

The approach rests on three core principles: spec-driven development, where requirements govern implementation; modular decomposition, where the system is divided into manageable parts handled by specialized Skills; and continuous validation, where each component is tested before the next step begins.

FIVE DIFFERENT SKILLS

The **Project Manager Skill** coordinates the work within the AI development effort, defines goals, and breaks the task down into appropriate sub-tasks. The **Architect Skill** is responsible for the structure and design of the solution, implements architectural patterns such as the medallion structure, and designs access control solutions. The **Requirements Skill** ingests and analyses business requirements, data sources, information models, and mappings. The **Technology Skill** contains deep platform-specific knowledge and translates architectural decisions into concrete implementation. The **Test Skill** ensures quality through unit tests and validation of data quality rules.

For a typical data pipeline, the Requirements Skill is used to analyse source data and mapping rules. With the help of the Architect Skill, the structure of the pipelines is designed. The Technology Skill is then used to implement the design and generate the code. Finally, the Test Skill is used to create unit tests that verify functionality. The Project Manager Skill coordinates the entire process and ensures that the output of each Skill becomes correct input to the next.

The modular structure resolves AI's limitation in handling broad complexity. Through specialized areas of responsibility, each Skill's task becomes sufficiently well bounded for predictable results. The structure mirrors how experienced teams naturally organize themselves and makes it possible to codify and scale expertise.

PRACTICAL APPLICATION

Successful implementation requires clearly documented requirements, access to relevant metadata about source systems, and a technical environment in which generated code can be executed and tested.

A project starts with a requirements phase in which the domain is analyzed and documented by human experts, often in workshop format. Once requirements have been documented, the AI framework is configured for the specific project by selecting the relevant Skills and providing them with project-specific context. The development phase is carried out iteratively, with the work broken down into manageable parts. Once all components have been generated, a final end-to-end validation is performed.

A critical success factor is continuous quality assurance. Each component is validated through automated unit tests and through experienced systems developers reviewing the generated code to verify that architectural decisions are followed correctly and that security aspects are handled properly.

FROM REQUIREMENTS TO A RUNNING PLATFORM

RAID turns a specification into a working platform through a deliberate sequence of phases, each of which produces a concrete result that is reviewed and approved before the next phase begins. This keeps direction firmly in human hands while the framework carries the detailed work.

The sequence begins with requirements analysis. Functional and non-functional requirements are captured and analysed, and the analysis is examined and refined until it accurately reflects the intended solution. From the approved requirements, together with the technical and architectural constraints, an architecture is derived; it too is reviewed and iterated until it is sound. A detailed design is then generated from everything known about the solution, including the architecture, and is reviewed in the same way.

With the technical solution settled, the work is planned. A project plan describes how the solution will be built and a test plan describes how it will be verified. The plan is then broken down into small, well-defined activities grouped into increments, each with its own tests. The framework develops one activity at a time and validates the result before moving on — confirming both that the code is well formed and that the intended functionality is met — with each outcome recorded in a test report.

Because every phase produces a reviewable result, problems surface early, while they are still inexpensive to correct, rather than after large amounts of code have already been written.

TWO STARTING POINTS: NEW PLATFORMS AND EXISTING ONES

RAID applies whether a platform is being built from the ground up or an existing one is being extended.

When building something new, the priority is to capture as many requirements as possible before implementation begins, since it is difficult to anticipate how requirements will be realised without a concrete functional starting point. A useful practice is to include, in the very first increment, a few functional requirements that are significant to the overall structure of the solution.

When building on an existing platform, the framework can first study the current system and derive its technical, architectural and non-functional characteristics, along with the standards and conventions already in place. These become the guardrails for any new or changed functionality, so that new work extends the existing structure and respects its conventions rather than working against them. From that point on, only the functional requirements for each new increment need to be supplied.

In both modes, the cross-cutting requirements — non-functional, technical and architectural, together with standards and conventions — are established once and then reused. For larger solutions, where functional requirements arrive over time, the architecture is updated incrementally as scope grows rather than being redesigned from scratch each time.

BENEFITS AND LIMITATIONS

Organizations that apply AI Data Engineering report several benefits. Development time is dramatically shorter – weeks or months become days. Quality is consistently high because the same best practices are always implemented, which reduces technical debt. Flexibility increases, since changes can be made quickly by adding new instructions to the relevant Skill. The ability to evaluate different technical platforms cost-effectively, simply by swapping out the Technology Skill, makes objective comparisons possible.

The method is, however, not suitable for every situation. It presupposes clearly articulated requirements and a well-defined domain. In exploratory phases, traditional methods may be more appropriate. The method also requires access to experienced systems architects who can break down complex systems and review the generated code. AI does not replace human expertise but rather changes how it is applied.

Security and data protection require careful analysis. The methodology operates with a separation between structure and data, in which AI uses metadata and schemas but not actual data values. For extremely sensitive domains, the method may need to be adapted.

SECURITY AND FUTURE DEVELOPMENT

The RAID methodology minimizes security risks through the separation between metadata and actual data. AI works with schemas and business rules but not with actual data values from production systems. Access control is implemented from the start, with row-level security and column masking. For organizations with particularly high security requirements, there is the option of running the AI services in their own cloud environment or using on-premises AI models.

RAID is applicable across the entire lifecycle of a data platform. Once a platform is in production, maintenance work can be substantially accelerated. Where traditional maintenance takes weeks, AI Data Engineering can reduce this to days. Quality remains consistent, since the same framework is used in maintenance as in initial development.

A further benefit over time is the ability to build organization-specific frameworks. Once an organization has used the methodology for several projects, it can describe its own standards in the framework, creating an accumulated knowledge base for future projects.

EXAMPLE: MIGRATING FROM A LEGACY PLATFORM

Migrating from an aging data platform to a modern environment is one of the clearest examples of what RAID makes possible. Such projects are often postponed, and for good reason: a migration rarely delivers any direct business value — in practice it is simply a cost. Mapping years of accumulated business logic, reviewing all existing code, performing impact analyses, and possibly switching the entire technology stack represents extensive manual work with high cost and considerable risk.

RAID changes that calculation. The framework examines the existing system and documents its structure, its technical and architectural conditions, and the standards it relies on. Once the new platform and architecture have been defined, the same functionality can be recreated in the new environment. Because RAID separates what the solution must do from how it is implemented, this is not a matter of translating old code line by line, but of generating new code from the documented functionality — in a fraction of the time.

The real gain lies further ahead. After the migration, RAID already knows the entire structure of the new solution, leaving the organization ready for continued development: new requirements need only be specified in order to be generated and tested. The migration thus becomes not merely a cost, but an investment that unlocks future development.

CONCLUSIONS AND RECOMMENDATIONS

RAID represents a fundamental change in how data platforms can be developed. By combining the speed of generative AI with structured systems development and documented expertise, it becomes possible to deliver production-ready solutions with consistently high quality in a fraction of the time required by traditional development.

The methodology is most valuable where requirements are well-defined, time pressure is high, and quality and scalability are critical. To succeed, clear requirements, access to experienced systems architects, a technical environment for testing, and a thoughtful strategy for security and data protection are required.

Organizations should start with a limited scope in order to build an understanding of the methodology and establish ways of working. Once the team is comfortable, the scope can gradually be expanded to larger systems.

AI technology continues to develop rapidly and will further improve AI Data Engineering. Coming generations of AI models will have a deeper understanding of complex systems and more robust code generation. The combination of better AI models and higher organizational maturity points toward a future in which data platforms can be developed and maintained with a fraction of today's resources and time.

ABOUT REBTECH

- Rebtech has more than **20 years of experience** developing data platforms and data analytics solutions. Our expertise in AI Data Engineering is built on practical lessons learned from hundreds of implemented data platforms.
- Our **RAID methodology** has been developed through experimentation and practical application. The expertise documented in our framework represents the combined experience of thousands of working hours in data platform development.
- **We support organizations throughout the entire journey,** from initial security analysis and requirements definition, through development and testing, to deployment and long-term maintenance.

RAID – A methodology for the efficient development of modern data platforms.
This white paper describes Rebtech's AI Data Engineering methodology as of May 2026.

REBTECH

www.rebtech.se