

Research on AI-Collaborative Empowerment Methods for Liquid Rocket Engines

Ning Li¹, Chenxi Zhang¹, Tianrui Wang^{1*}

¹ Xi'an Aerospace Propulsion Institute, Xi'an, Shanxi, China

*Correspondence: Tianrui Wang

Received: 25 May 2026; Accepted: 20 June 2026; Published: 1 July 2026

Abstract:

The development of liquid rocket engines is characterized by multidisciplinary coupling, high complexity, and strict reliability requirements. Its design, simulation, testing, and fault analysis processes are accompanied by massive amounts of heterogeneous data and complex engineering knowledge. However, the current development of aerospace models generally faces practical challenges such as a weak data governance foundation, scarce fault samples, and scattered knowledge that is difficult to reuse. Traditional data-driven artificial intelligence methods that rely on large-scale, high-quality annotated data are difficult to implement directly. Focusing on the typical engineering requirements in the development process of liquid rocket engines, this paper systematically reviews three categories of applicable artificial intelligence support methods: data-driven methods, knowledge-enhanced methods, and large model agent methods, and analyzes their respective application boundaries and engineering characteristics. On this basis, aiming at typical scenarios such as model collaborative management, conceptual scheme design, simulation analysis acceleration, test data analysis, and quality and fault attribution, a scenario-method matching scheme oriented toward engineering practice is proposed. Meanwhile, addressing the realistic constraints of current suboptimal data conditions, an implementation path of “governing while using, human-machine closed-loop, and progressive evolution” is proposed to achieve data accumulation and capability iteration through expert feedback and AI collaboration. Research shows that in the field of liquid rocket engine development, artificial intelligence empowerment should not rely on a single technical route but should be combined and applied according to scenario characteristics and data conditions to achieve a progressive enhancement of intelligent capabilities. The method system and implementation path proposed in this paper can provide a reference for the intelligent development of liquid rocket engines, and also have certain reference value for AI applications in other complex equipment systems.

Keywords:

Liquid Rocket Engine, Artificial Intelligence Method System, Lightweight Multi-Agent, Human-Machine Closed-Loop, Data Governance

1. Introduction

1.1. Research Background and Dilemmas

Liquid rocket engines are the core power units of aerospace launch systems, and their development involves the deep coupling of multiple disciplines such as fluid dynamics, combustion processes, and thermal structural responses. These processes typically exhibit strong non-linearity, strong coupling, and multi-scale characteristics, posing huge challenges to traditional modeling and analysis methods. As engine performance requirements continue to increase, the design parameter space has expanded significantly, the scale of test data has continued to grow, and the data generated during the development process exhibits characteristics of high dimensionality, large scale, and multi-source heterogeneity. In recent years, the field of aerospace engines has been accelerating toward intelligence, and a large number of technical explorations have been carried out around scheme design, simulation acceleration, test monitoring, and fault attribution. The application of artificial intelligence in engine operating process simulation has become an important development direction.

The intelligent exploration of aerospace engines has covered multiple fields such as solid rocket engines, liquid rocket engines, and ramjets. Related research encompasses applications of artificial intelligence in engine operating process simulation, state recognition and parameter prediction, and design parameter optimization [1]. However, most current units face the practical dilemma of a weak business data governance foundation. Development data is scattered across isolated systems such as design, simulation, testing, and production, with problems such as inconsistent data formats, missing meta-information, misaligned timelines, scarce fault samples, and uneven annotation quality. If one waits for data governance to be perfected before starting artificial intelligence construction, the cycle will be long and the results slow. Therefore, finding a feasible intelligent empowerment path under suboptimal data conditions is a key problem that urgently needs to be solved at present.

Some scholars have pointed out that data-driven intelligent fault diagnosis faces three core challenges in the practical health monitoring applications of liquid rocket engines: the scarcity of fault samples makes supervised learning methods difficult to train, the complexity of physical models makes cross-condition migration difficult, and there is insufficient effective integration of expert knowledge[2]. This systematic analysis profoundly reveals the structural bottlenecks currently faced in the advancement of intelligence. Therefore, exploring intelligent methods that do not rely on large-scale, high-quality annotated data and can effectively integrate physical mechanisms and expert knowledge has become a key direction to break through this bottleneck.

1.2. Core Issues and Approaches

This paper focuses on two interrelated foundational issues. The first is the method issue: for the typical tasks of liquid rocket engine development, what artificial intelligence methods are available to choose from, and what are their respective applicable conditions and engineering boundaries. The second is the scenario issue: under the practical constraints of weak data governance, how to choose appropriate scenarios to cut into the development process, and organically combine multiple methods to achieve progressive empowerment.

The general idea of this paper is: first, systematically review various artificial intelligence methods applicable to engine development, and clarify their advantages, disadvantages, and application boundaries under different data conditions and physical knowledge backgrounds; then, provide targeted method matching suggestions for typical scenarios; in complex tasks involving multi-step and multi-role collaboration, introduce the lightweight agent idea as an organizational coordination mechanism.

1.3. Research Contributions

The main contributions of this paper are reflected in three aspects. First, it systematically reviews and compares three categories of artificial intelligence support methods, clarifying their respective application conditions and engineering variants. Second, it provides a targeted matching scheme between scenarios and methods oriented toward data-weak environments, serving as a direct reference for engineering practice. Third, it designs a progressive implementation path for conditions with weak data governance.

2. Materials and Methods

Based on the different characteristics of information sources and implementation paradigms, this paper categorizes the artificial intelligence methods that can be used for engine development into three types. In actual engineering, there is no absolute superiority or inferiority among these methods; instead, they are flexibly selected based on the data conditions of specific problems, the completeness of physical knowledge, and decision-making requirements. In addition, these methods are also frequently used in combination to leverage complementary advantages, but this belongs to the strategy of engineering application rather than an independent method category.

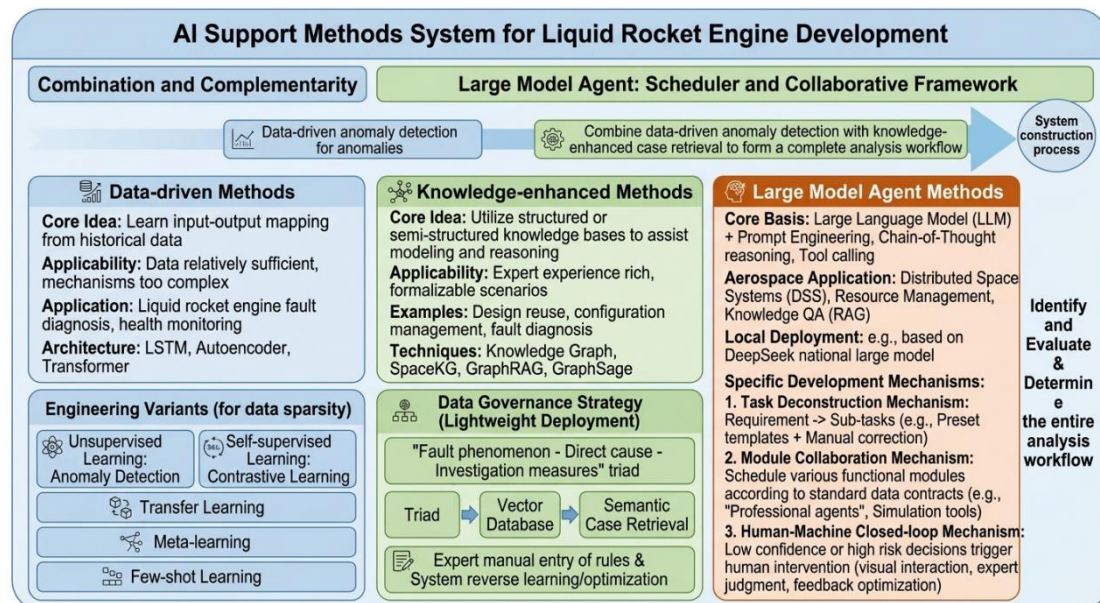


Figure 1. Diagram of the AI Support Method System.

2.1. Data-Driven Methods

The data-driven method is a representative paradigm of current artificial intelligence research, and its core idea is to learn the mapping relationship between

inputs and outputs from historical data. This type of method is most suitable for sub-problems where data is relatively sufficient but the underlying mechanisms are too complex to model accurately. In the field of fault diagnosis for liquid rocket engines, data-driven technologies such as machine learning and deep learning have been widely adopted for health monitoring. Existing studies have constructed a variety of intelligent fault diagnosis algorithms based on model architectures such as Long Short-Term Memory networks (LSTM), autoencoders, and Transformers, achieving significant results in fault recognition for rotating machinery such as gearboxes and bearings [3].

In the field of aero-engines, data-driven methods are widely used for intelligent diagnosis and pattern recognition of fault signals. However, since actual fault samples for liquid rocket engines are extremely scarce, traditional supervised learning methods that rely on large-scale annotated data face severe challenges. Therefore, aiming at the problem of sparse data annotation in engine development, engineering variants should focus on unsupervised and self-supervised learning strategies—using LSTM autoencoders to perform anomaly detection on a large amount of unannotated hot-fire test data, and extracting effective sequential features on unannotated data through contrastive learning. In addition, transfer learning and meta-learning can effectively reduce the data volume requirements for target operating conditions, utilizing knowledge from hot-fire test data of related fields such as similar models or historical records of gas turbines to make up for the lack of target data. In scenarios where fault samples are extremely rare, emerging paradigms such as few-shot learning provide a feasible path for training deep learning models [4].

2.2. Knowledge-Enhanced Methods

Knowledge-enhanced methods utilize structured or semi-structured knowledge bases to assist modeling and reasoning. In the aerospace field, knowledge-driven technological exploration has a history of many years. Knowledge-based engineering methods are widely applied in scenarios such as design reuse, configuration management, and fault diagnosis. Chen and other scholars proposed a hybrid knowledge extraction method combining human expert knowledge and interpretable machine learning to support early concurrent engineering in the aerospace industry. This method is used to formalize knowledge models and instantiate them into specific design rules, laying a methodological foundation for the intelligence of scheme design [5].

Recent SpaceKG research further proposed combining knowledge graphs with aerospace systems engineering to build real-time, data-driven cognitive digital twins, driving intelligent decision-making, saving resources, and empowering experts through a deducible reasoning engine. The framework proposed in this study aims to combine the full life-cycle activities of satellite projects with the semantic and operational dynamics of knowledge graphs to achieve digital continuity in aerospace systems—this framework has been preliminarily verified in a fault event detection case of the space shuttle main engine [6].

To advance knowledge-driven aerospace engineering management, a systematic literature review completed by Wittenborg and other scholars constructed a knowledge graph containing more than 700 aerospace engineering processes, software, and data entities. This review systematically organized the main application directions

of knowledge engineering in the aerospace field, laying a methodological foundation for knowledge-driven aerospace engineering management [7].

At the same time, oriented toward assisting the fault isolation of spacecraft, domestic scholars proposed a method based on GraphRAG. Through failure analysis (zeroing) knowledge ontology modeling, community detection, and graph multi-hop retrieval, it drives large models to accurately assist fault isolation [8]. In the field of civil aircraft fault diagnosis, a fault diagnosis method based on GraphSage and attention mechanisms has also achieved good results [9].

In engine development, knowledge-enhanced methods are mainly applicable to scenarios where expert experience is rich and can be formalized, such as historical fault mode retrieval, design specification compliance checks, and similar case recommendations. A significant advantage of this type of method is its low reliance on large-scale training data—the core information comes from the explicit knowledge of experts and the structured storage of historical documents, rather than massive annotated samples.

Under the constraint of weak data governance, knowledge-enhanced methods should also adopt a lightweight deployment strategy. In the initial stage, there is no need to pursue the construction of a comprehensive and fine-grained knowledge graph; instead, triplets of "fault phenomenon - direct cause - troubleshooting measure" can be extracted from semi-structured documents such as fault reports and design manuals, and stored in the form of a vector database to support case retrieval based on semantic similarity. Meanwhile, the system should open an interface for experts to manually enter rules—when the model encounters an unseen case, it allows experts to directly supplement new fault modes, correct retrieval results, and add judgment rules in the interface. These manual inputs and corrections are automatically recorded by the system, gradually settling into structured content in the knowledge base, and ultimately reversing to promote the formalization and governance of the knowledge system.

2.3. Large Model Agent Methods

The large model agent method is a cutting-edge paradigm that has emerged in recent years. Based on large language models (LLMs), it builds agents with autonomous planning and execution capabilities through mechanisms such as prompt engineering, chain-of-thought reasoning, and tool invocation. Internationally, the ReAct framework proposes interleaving the generation of reasoning traces and task-specific actions, enabling the model to induce, track, and update action plans, while interacting with external environments like knowledge bases to gather information. This framework outperforms chain-of-thought prompting on language understanding tasks and achieves significant absolute success rates compared to imitation learning and reinforcement learning methods in interactive decision-making tasks [10]. Barbosa and other scholars explored the use of LLMs in multi-agent systems to solve complex tasks in a collaborative and collaborative manner. Through a case study of a product manufacturing scenario, they demonstrated the effectiveness of LLM-based multi-agent systems in collaborative decision-making tasks, but simultaneously pointed out that the operating cost of large language models remains a major challenge for widespread application [11].

In the aerospace field, systematic research on multi-agent related methods has received increasing attention. Shahdadi and other scholars conducted a comprehensive

review of agent-based approaches in distributed space systems (DSS) and mission management, proposing a classification system based on agent types, architectures, communication protocols, etc., and pointing out that existing methods have made significant progress in dealing with real-time decision-making, coordination, and fault recovery in highly dynamic large-scale networks, though challenges such as interoperability and scalability under communication constraints still exist [12]. Meanwhile, Li Meng and other scholars discussed the technical paths, method systems, engineering challenges, and development opportunities of multi-agent reinforcement learning empowering space unmanned systems, particularly pointing out the potential of multi-agent reinforcement learning in key space scenarios such as dynamic spectrum allocation, onboard edge computing, and robust collaborative control [13].

Large model methods also provide powerful knowledge question-answering (Q&A) capabilities. Through RAG technology, relevant information can be retrieved from unstructured documents and knowledge bases, and then precise answers can be generated by the large language model, which is of great value for scenarios such as design knowledge queries and fault case retrieval in engine development.

In terms of distributed collaborative decision-making, the review by Shahdadi et al. emphasized that agent-based approaches provide a promising path for solving complex problems such as large-scale network resource management, coordination, and real-time decision-making in complex aerospace systems through self-governance, decentralization, and adaptive capabilities.

Engineering practice at the Xi'an Aerospace Base shows that the 11th Institute of the Sixth Academy of China Aerospace Science and Technology Corporation relied on the localized deployment of domestic large models such as DeepSeek to build an intelligent tool platform, achieving remarkable results in core engine knowledge acquisition. However, the “large model agent method” discussed in this paper is not limited to fully automated autonomous decision-making by large language models, but rather intervenes in the engine development process in a more engineering-feasible manner.

In the specific scenarios of liquid rocket engine development, the core value of large model agents is reflected in three interrelated mechanisms.

Task Deconstruction Mechanism. Facing high-level engineering requirements such as “analyze the cause of thrust drop in a certain hot-fire test,” the task planning agent deconstructs it into several relatively independent sub-tasks, such as data loading and preprocessing, anomaly detection and time window positioning, multi-measurement-point correlation feature extraction, historical similar case retrieval, causal reasoning for root cause hypotheses, simulation verification, and report generation. In the initial stage where data preparation is insufficient and workflows are not yet standardized, the deconstruction does not pursue full automation all at once, but adopts a semi-automatic mode of “preset template + manual correction”: the system recommends a default workflow based on the scenario type, and engineers can add or delete sub-tasks and adjust the execution order through the interactive interface. As more task planning instances are accumulated during use, the system can gradually learn the habits and preferences of engineers, improving the accuracy and satisfaction rate of automatic deconstruction.

Module Collaboration Mechanism. After completing the deconstruction, the collaborative scheduler invokes various functional modules sequentially or in parallel according to the data dependencies between sub-tasks. Each module is encapsulated as an independent “professional agent,” which can be the aforementioned data-driven model, a knowledge retrieval engine, or a traditional CFD simulation tool invocation interface. To achieve plug-and-play integration, each type of module must follow a standardized data contract: the output of the anomaly detection agent must include timestamps, anomaly scores, and a list of associated measurement points; the output of the causal reasoning agent should include the causal graph structure, root-cause node probabilities, etc. This contract mechanism allows agent modules developed by different teams at different times to collaborate smoothly under a unified collaborative framework without modifying their respective internal code implementations. Targeting the interoperability and resource management challenges faced by aerospace distributed systems, standardized service-oriented encapsulation and contract governance become the solid engineering foundation of this collaborative framework.

Human-Machine Closed-Loop Mechanism. This is a key design that distinguishes the agent method from fully automated systems, and it is also an effective means to combat weak data governance. In two types of situations, the collaborative framework will actively trigger manual intervention: first, when the output confidence of a certain agent is lower than the preset threshold—for example, the anomaly detection score is in the critical region between normal and abnormal, or the root cause probability given by the causal reasoning model does not meet the minimum confidence requirement; second, when the task involves high-risk decisions—for example, the attribution conclusion may lead to design changes or test termination. When manual intervention is triggered, the human-machine interaction agent displays intermediate results and multiple reasoning paths to experts in a visualized manner, and domain experts make the final judgment. The expert's confirmation, modification, or rejection results serve as feedback signals passed to the continuous learning module for subsequent agent fine-tuning or threshold optimization. This design transforms “data annotation” from an expensive front-end investment before AI work begins into a low-cost, accompanying activity that is “done along the way” within the process: when experts confirm the model output, they are actually providing high-quality annotated data for the system.

In order to cope with more complex engineering problems, these three types of methods often need to be used in combination. For example, a large model agent can serve as a coordinator to string together a data-driven anomaly detection model and a knowledge-enhanced case retrieval component into a complete analysis workflow; or in simulation acceleration, a multi-fidelity surrogate model combined with an active learning strategy can be adopted to complete parameter searches with the minimum number of high-cost simulation invocations. In the field of fault diagnosis, researchers have proposed a hybrid framework that comprehensively applies signal processing methods, model-driven methods, and artificial intelligence methods to make up for the natural limitations of a single method. The core advantage of these combination strategies lies in “drawing on each other’s strengths”: data-driven models solve pattern recognition problems but lack physical interpretability, knowledge-enhanced methods provide traceable reasoning logic but have limited coverage for unknown situations, while agent methods schedule, combine, and connect these scattered

capabilities from the perspective of tasks, ultimately forming an overall efficacy where “1+1>2.”

3. Application Scenarios and Targeted Method Matching

After systematically reviewing the three types of methods, this section provides targeted method matching suggestions for typical scenarios in the engine development process, as shown in Figure 2. The principle of matching is: select the most suitable method based on the core difficulties and data conditions of the scenario, without being rigidly bound to a certain fixed expression format.

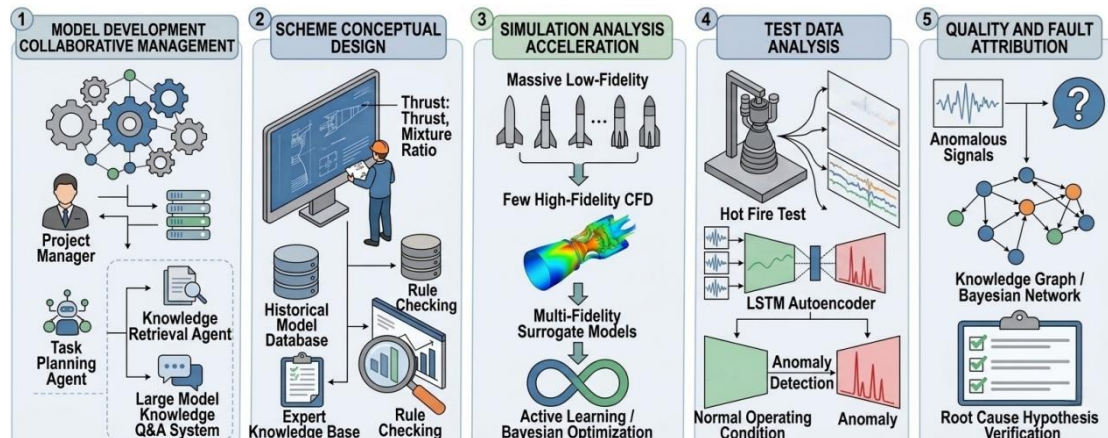


Figure 2. Diagram of Typical Scenarios and Method Matching in Engine Development.

3.1. Scenario One: Model Development Collaborative Management

Model development is a complex systems engineering endeavor covering multiple stages such as conceptual design, detailed design, manufacturing, assembly and debugging, ground testing, and flight testing. The core challenge of this scenario is the serial-parallel intersection of multiple stages, cross-disciplinary integration, and multi-role collaboration, featuring a long information flow and intensive feedback. Traditional project management methods suffer from lagging information, difficult task tracking, and untimely risk identification. This scenario has a low reliance on historical data but requires high levels of task coordination and information integration.

Aiming at this characteristic, it is suitable to adopt the agent method to construct a collaborative management framework, while introducing knowledge enhancement to assist decision-making. The task planning agent can decompose the overall development plan into sub-tasks for various specialties and stages, dynamically adjusting priorities and resource allocations; the knowledge retrieval agent pushes lessons learned and standard specifications from similar stages to engineers. In addition, a large-model-based knowledge Q&A system can be deployed, enabling engineers to query project management regulations, technical documents, and historical decision records using natural language, quickly obtaining the required information. The human-machine interaction agent summarizes progress and risks at key milestone nodes to assist project managers in decision-making. This combination can provide valuable collaborative support under data-sparse conditions.

3.2. Scenario Two: Scheme Conceptual Design

Scheme conceptual design is the starting point of engine development. The determination of overall parameters such as thrust, mixture ratio, and combustion

chamber pressure directly affects all subsequent links. The characteristic of this scenario is that historical model experience has important reference value, but structured data is relatively sparse, and it involves multi-disciplinary trade-offs. This scenario has a low demand for large-scale data-driven models and relies more on expert knowledge and historical cases.

Therefore, the knowledge-enhanced method is the preferred choice for this scenario. Case retrieval and similarity matching can be performed based on the correlation between historical model design parameters and performance indicators to recommend initial schemes; rules can be used to check design baseline compliance; and lightweight regression models can be applied to rapidly evaluate a small number of parameters. At the same time, large model methods can be utilized to build a design knowledge Q&A system, allowing designers to query design guidelines, material parameters, process constraints, and other information using natural language, with the system returning precise answers through RAG. This type of Q&A system is particularly suitable for the early stages of data governance—experts can directly supplement the Q&A library, gradually accumulating knowledge.

3.3. Scenario Three: Simulation Analysis Acceleration

High-fidelity numerical simulation is one of the most computationally expensive links in engine development, where a single CFD simulation can take several hours or even days. The core characteristic of this scenario is that the physical mechanisms are clear, but computation is expensive, and the cost of acquiring simulation data is high. This scenario naturally lends itself to introducing physical constraints to compensate for data deficiencies. However, since this paper focuses on data-driven, knowledge-enhanced, and agent methods, a combination strategy is recommended, namely fusing a small amount of high-fidelity simulation results with a large number of low-fidelity approximate models.

Specifically, multi-fidelity surrogate modeling methods can be utilized to fuse a small amount of high-fidelity CFD simulation results with a large number of low-fidelity approximate models through Gaussian processes or neural networks, training a surrogate model with relatively high precision. Meanwhile, an active learning strategy is launched, where the surrogate model intelligently suggests the next set of parameter points that need to run high-fidelity simulations based on uncertainty analysis, and the newly obtained simulation data is used to incrementally update the surrogate model. In addition, Bayesian optimization can also be used to efficiently explore the design space, approaching the optimal solution with the minimum number of objective function evaluations.

3.4. Scenario Four: Test Data Analysis

Hot-fire testing is the most critical verification link in engine development. A single test generates hundreds of channels of high-frequency sampled sequential data, and fault samples are extremely scarce. The core characteristic of this scenario is that the data volume is large and the dimensionality is high, but annotations are sparse, requiring the system to learn patterns from normal condition data and detect anomalies in real-time or post-event.

Unsupervised data-driven methods perform prominently in this scenario. LSTM autoencoders or Isolation Forests can be used to perform anomaly detection on a large amount of unannotated test data. The model is trained using only normal condition data, and an anomaly is triggered when the reconstruction error exceeds a threshold.

The task chain of this scenario is relatively fixed, and can be directly completed by a single model. During post-event analysis, it can work with a large-model-based knowledge Q&A system to query handling records of similar anomaly modes, assisting engineers in quickly locating problems.

3.5. Scenario Five: Quality and Fault Attribution

When an anomaly occurs during a test or an engine fails, it is necessary to locate the cause quickly and accurately. The core characteristics of this scenario are that fault samples are rare, but rich domain knowledge has been accumulated in historical fault reports, and the attribution process usually requires multi-step reasoning: starting from anomalous signals, retrieving similar cases, proposing root cause hypotheses, verifying hypotheses, and making comprehensive judgments. This scenario features a long task chain and involves multiple information sources.

Knowledge enhancement combined with causal reasoning is an appropriate solution. Knowledge graphs or vector databases can be utilized to retrieve historical similar fault cases; Bayesian networks or structural causal models can be adopted to infer the most likely root cause from anomalous signals; for critical hypotheses, lightweight simulation verification can be invoked. At the same time, a large-model-based fault knowledge Q&A system can be deployed, enabling engineers to ask questions using natural language. By retrieving the fault knowledge base and causal graphs, the system provides answers with confidence levels and supporting evidence chains. The entire attribution workflow can be encapsulated as a decision support tool, used interactively by engineers step-by-step. In cases where higher automated orchestration is desired, the agent method can also be introduced as an organizational framework.

The matching schemes for the above five scenarios follow a common logic: in scenarios with extremely weak data conditions but rich expert knowledge, such as scheme design and fault attribution, knowledge-enhanced methods are prioritized; in scenarios with large data volumes but sparse annotations, such as test analysis, unsupervised data-driven methods are prioritized; in scenarios with complex processes and multi-role collaboration, such as model collaborative management, agent methods are preferentially introduced; in computationally intensive scenarios like simulation, a combination strategy is adopted. In engineering practice, flexible adaptation can be carried out according to this logic based on the specific characteristics of the actual problem.

3.6. Scenario-Method Matching Matrix

Based on the above analysis, the corresponding relationships between the five core scenarios and the adapted methods are organized into Table 1 Scenario-Method Matching Matrix, facilitating quick reference in engineering practice.

Table 1. Scenario-Method Matching Matrix.

Core Scenario	Recommended Method	Auxiliary Method	Description
Model Development Collaborative Management	Agent Method	Knowledge Enhancement, Large Model Q&A	Multi-role and multi-task collaboration, supporting natural language queries
Scheme Conceptual Design	Knowledge Enhancement	Data-Driven, Large Model Q&A	Relies on historical cases and expert rules; Q&A assists in querying design knowledge

Simulation Acceleration	Combination Strategy	Data-Driven	Fuses a small amount of high-fidelity data to train surrogate models
Test Analysis	Data-Driven	Statistical Process Control, Large Model Q&A	Unannotated sequential anomaly detection; can work with Q&A post-event to query handling experience
Fault Attribution	Knowledge Enhancement + Causal Reasoning	Optional Agent, Large Model Q&A	Multi-step reasoning; Q&A system assists in fault case retrieval

4. Implementation Path Oriented Toward Data-Weak Environments

After clarifying the various methods and their applicable scenarios, a natural question arises: under the practical constraints of weak data governance and scarce annotated samples, how does one deploy these methods into actual work from scratch? Different methods have different requirements for data conditions. Knowledge-enhanced methods rely barely on historical data and can be launched first; unsupervised data-driven methods require only a small number of normal samples to start; while agent-based methods require a certain degree of process standardization and feedback accumulation to be effective. Therefore, it is necessary to design a phased, progressive path to let various methods land sequentially according to their dependency conditions, and gradually accumulate data and enhance capabilities during use. This section proposes an implementation strategy of “governing while using, human-machine collaboration,” and provides specific operations for three stages, as shown in Figure 3.

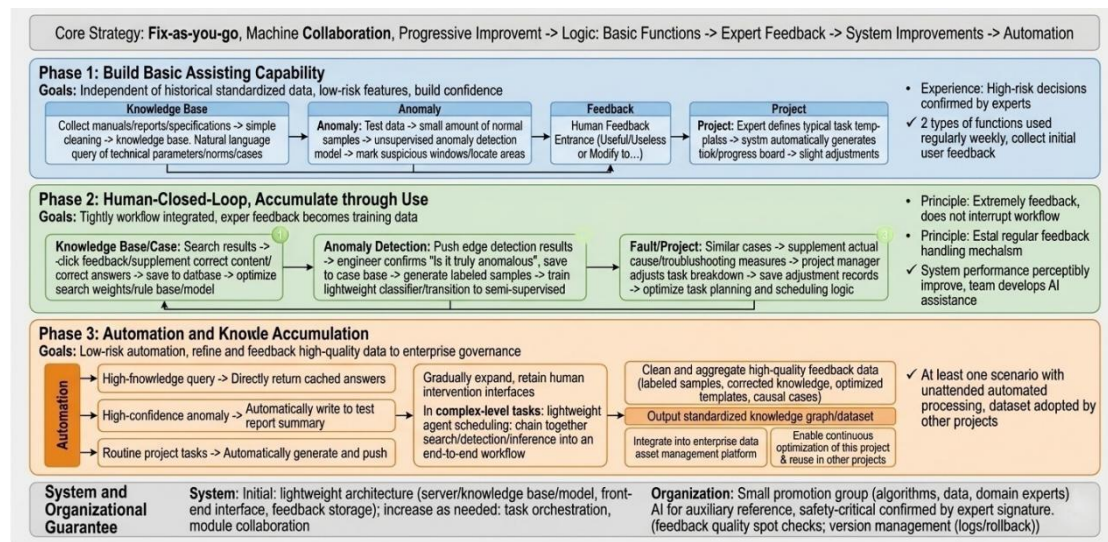


Figure 3. Diagram of the Implementation Path Oriented Toward Data-Weak Environments.

4.1. Core Strategy

Under the practical constraints of weak data governance, artificial intelligence empowerment should not wait for data to be complete before starting, but should adopt the strategy of “governing while using, human-machine collaboration, and progressive enhancement.” The core logic is: start with the most easily launched functions that have the lowest data requirements, allowing the system to generate auxiliary value in actual use; simultaneously integrate expert judgment into the

workflow, where every manual confirmation and correction becomes material for system improvement; as data accumulates, system capabilities progressively enhance, ultimately achieving the transition from “manual assistance” to “automated assistance.” The three types of methods—utilizing existing knowledge for retrieval and reasoning, learning patterns from unannotated data, and task planning and scheduling for complex processes—supplement each other during the implementation process, and do not need to be deliberately separated; instead, they should be flexibly configured according to the needs of specific problems.

4.2. Stage One: Building Basic Auxiliary Capabilities from Scratch

The goal of this stage is to rapidly launch several low-risk auxiliary functions without relying on historical annotated data, allowing the team to establish initial confidence. Specific work includes the following aspects.

First, collect existing documents such as design manuals, fault reports, and technical specifications, and build a retrievable knowledge base after simple cleaning, enabling engineers to query technical parameters, design guidelines, or historical cases using natural language. This kind of assistance based on existing knowledge does not require any training data and can be used as soon as it goes online. Second, targeting test data, train an unsupervised anomaly detection model using only a small number of samples from normal conditions to flag suspicious time windows in real-time, helping engineers quickly locate focus areas from hundreds of measurement points. This represents the basic capability to automatically discover patterns from data. Third, in model project management, experienced project managers define deconstruction templates for several typical tasks, and the system automatically generates task lists and progress dashboards according to the templates—project managers only need minor adjustments to use them, laying a foundation for subsequent, more complex task planning capabilities. Finally, equip all auxiliary functions with a simple manual feedback entry, where engineers can mark results as “useful/useless” or “modify to...”.

This stage does not pursue accuracy, and all high-risk decisions are still ultimately confirmed by experts. The sign of success for this stage is: at least two types of auxiliary functions are regularly used by the team every week, and initial user feedback has been collected.

4.3. Stage Two: Human-Machine Closed-Loop, Allowing Use Processes to Precipitate Data

Once basic functions are accepted, the system needs to be integrated more tightly with daily work, so that every expert confirmation, correction, and supplement can be automatically transformed into training data or knowledge items, which in turn improves the intelligent level of the system.

In terms of knowledge retrieval and case recommendation, the system adds a one-click feedback button when returning results, and allows experts to supplement correct content or fix wrong answers. This feedback is directly saved into the database and periodically used to optimize retrieval weights, expand the rule base, or fine-tune underlying models. In terms of anomaly detection, detection results located in boundary regions are pushed to engineers to confirm “whether it is a true anomaly,” and the engineer's click generates an annotated sample. After accumulating a certain amount, this annotated data can be used to train a lightweight classifier, progressively

reducing the false alarm rate and even transitioning from unsupervised to semi-supervised learning. In terms of fault attribution and project collaboration, when engineers retrieve similar fault cases through the system, they can supplement the actual cause and troubleshooting measures of the current fault and save them into the case library; when project managers adjust the task deconstruction scheme automatically generated by the system, the adjustment records are saved to optimize subsequent task planning and scheduling logic. These adjustments essentially provide supervisory signals for the agent's decision-making.

The key is to make feedback operations extremely simple and ensure they do not interrupt the main workflow. Meanwhile, establish a mechanism to periodically process feedback data. The sign of success for this stage is: a perceivable improvement in system performance occurs (for example, answers to the same questions become more accurate, anomaly false alarms decrease, task deconstruction aligns better with reality), and the team has become accustomed to using AI assistance as a normal part of daily work.

4.4. Stage Three: Automation and Knowledge Precipitation

When feedback data has accumulated to a sufficient scale and the stability of the system has been verified, automated decision-making can be attempted in some low-risk links, and the precipitated high-quality data can be fed back to feed the enterprise's data governance system.

In terms of automation, for frequently occurring knowledge queries, the system can directly return cached answers without real-time retrieval and generation; for high-confidence results in anomaly detection, they can be automatically written into test report summaries without manual review; for routine project tasks, the system automatically generates and pushes them, requiring confirmation only at key nodes. These automated actions can gradually expand in scope, but always retain interfaces for manual rollback and intervention.

More importantly, content that has been cleaned and aggregated from the feedback data lake—including annotated samples, corrected knowledge items, optimized task templates, and causal cases—is outputted as standardized knowledge graphs or datasets, and incorporated into the enterprise data asset management platform. These results can not only be used for continuous optimization of the current project, but can also be reused by other models or projects, truly achieving knowledge precipitation and governance enhancement.

In addition, in complex tasks that require the collaboration of multi-step and multi-information sources, lightweight agent scheduling can begin to be introduced, stringing modules like knowledge retrieval, data detection, and causal inference into an end-to-end workflow to further improve efficiency. However, this step is not mandatory and can be decided based on actual needs.

The sign of success for this stage is: at least one scenario achieves unattended automated processing while meeting engineering requirements, and the dataset delivered to the data governance platform is adopted by other projects.

4.5. System and Organizational Guarantees

At the system level, the initial stage requires only a lightweight architecture: a server side storing knowledge bases and models, a front-end interaction interface, and

a storage area for feedback data. As requirements increase, task orchestration and module collaboration capabilities can be progressively introduced. At the organizational level, it is recommended to establish a small advancement group consisting of algorithm engineers, data engineers, and domain experts, clarifying that AI outputs serve as auxiliary references and safety-critical decisions still require expert signatures. Establish a feedback data quality sampling inspection system to periodically eliminate obviously incorrect annotations. Manage system versions, ensuring changes have logs and rollback plans.

5. Conclusions

Grounded in the reality of weak data governance in liquid rocket engine development, this paper systematically reviews the application boundaries and engineering variants of three categories of artificial intelligence support methods. The three types of methods have their own strengths: data-driven methods excel at pattern recognition but are constrained by data quality; knowledge-enhanced methods do not rely on large-scale training data but are limited by the completeness of knowledge expression; large model agent methods excel at task planning and scheduling, while providing powerful knowledge Q&A capabilities. In actual deployment, one should not pin hopes on any single method, but should carry out intelligent matching and combination based on specific data conditions and engineering constraints.

Aiming at typical scenarios, this paper provides targeted method matching schemes: model development collaborative management is suitable for introducing agent methods for task orchestration, supplemented by large model knowledge Q&A systems to improve information acquisition efficiency; scheme conceptual design should be based on knowledge enhancement, combined with large model Q&A to assist in querying design knowledge; simulation acceleration can adopt a multi-fidelity combination strategy, utilizing limited high-fidelity data to train available surrogate models; test analysis should use unsupervised data-driven methods, which can work with Q&A systems post-event to query handling experience; fault attribution is suitable for a combination of knowledge enhancement and causal reasoning, where large model Q&A can further assist in fault case retrieval. This differentiated matching strategy helps achieve pragmatic, progressive empowerment under data-weak conditions.

Intelligent fault diagnosis for liquid rocket engines faces core challenges such as scarce annotated samples and changing operating conditions, and the human-machine closed-loop and data accumulation dual-loop path proposed in this paper serves exactly as a feasible solution to this structural bottleneck. This paper argues that weak data governance should not become a technical barrier to artificial intelligence empowerment—through the feedback mechanism of a human-machine closed-loop and a phased, progressive implementation path, data can be gradually accumulated and precipitated during use, achieving synchronous iteration and enhancement of data governance and AI capabilities. The method framework and implementation strategies provided in this paper also possess certain reference value for other complex engineering systems that face similar historical data dilemmas.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

- [1] Zhang, J.B.; Xu, X.; Liu, J.X, et al. Intelligent Development Trend of Aerospace Engines (in Chinese). *Journal of Rocket Propulsion*, 2025, 51(3), 115-133.
- [2] Yang, S.M.; Wu, J.J.; Xie, C.L, et al. Application issues of data-driven intelligent fault diagnosis technologies for liquid rocket engines. *Acta Aeronautica et Astronautica Sinica*, 2025, 46(15).
- [3] Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Computing Surveys*, 2009, 41(3), 1-58. DOI: 10.1145/1541880.1541882.
- [4] Hu, H.F. Research Status of Intelligent Fault Diagnosis for Liquid Rocket Engines (in Chinese). *Aerospace Control*, 2023, 41(1), 3-14.
- [5] Duverger, E.; Arista, R.; Aubry, A , et al. A Hybrid Knowledge Extraction Method to Support Early Concurrent Engineering in the Aerospace Industry. *Aerospace*, 2026, 13(4), 337-337. DOI: 10.3390/AEROSPACE13040337.
- [6] Patrignani, L.; Laurenza, E.; Sallinger, E, et al. SpaceKG: Towards exploiting Knowledge Graphs in space systems. *Acta Astronautica*, 2025, 236, 967-981.
- [7] Wittenborg, T, et al. Knowledge-Based Aerospace Engineering — A Systematic Literature Review. arXiv:2505.10142, 2025.
- [8] Ai, S.J.; He, Y.; Zhang, W, et al. A Spacecraft Fault Auxiliary Isolation Method Based on GraphRAG (in Chinese). *Spacecraft Engineering*, 2025, 34(4), 84-90.
- [9] Li, Z.; Ma, J.; Fan, R, et al. Aircraft sensor fault diagnosis based on GraphSage and attention mechanism. *Sensors*, 2025, 25(3), 809.
- [10] Yao, S.Y, et al. React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629 (2022).
- [11] Barbosa, R.; Santos R.; Novais P. Collaborative problem-solving with LLM: A multi-agent system approach to solve complex tasks using Autogen [C]//Highlights in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins. Springer, 2025.
- [12] Jabbarpour, R.M.; Vo, Q.B.; Dalahmeh, E.G , et al. Agent-based approaches for distributed space systems and mission management: Methodologies, current practices and challenges. *Acta Astronautica*, 2026, 238(PB), 237-257. DOI: 10.1016/J.ACTAASTRO.2025.10.018.
- [13] Li, M.; Feng, Z.H.; Mei, Y.P. et al. Multi-agent Reinforcement Learning Empowering Space Unmanned Systems: Methods, Challenges and Opportunities (in Chinese). *Aerospace Control and Application*, 2025, 51(4), 17-28.



© 2026 by the author(s); licensee International Technology and Science Publications (ITS), this work for open access publication is under the Creative Commons Attribution International License (CC BY 4.0). (<http://creativecommons.org/licenses/by/4.0/>)