Construction and Application of an AI-Native Vertical Intelligent Agent System for Teaching Based on the STS8200 Test Platform

Jingtian Guo[1, a *], Qianmo Li[2, b], Guorui Lyu[3, c], Mengle Han[4, d], Shihan Zheng[1, e]

[1] School of Integrated Circuits, North China University of Technology, Beijing, China
[2] Brunel LondonSchool, North China University of Technology, Beijing, China
[3] College of Science, North China University of Technology, Beijing, China
[4] School of Artificial Intelligence and Computer Science, North China University of Technology, Beijing, China

[*a]jingtiangrace@gmail.com, [b]qume2005@outlook.com, [c]larkycat@mail.ncut.edu.cn, [d]dchanml@163.com, [e]shihanzheng007@gmail.com

Abstract. Automatic Test Equipment (ATE) is a vital component of the semiconductor industrial chain. Its multidisciplinary attributes create substantial difficulties for novice learners in theoretical study, who also rely heavily on one-on-one guidance from teachers. To improve learning efficiency, this paper proposes an AI-native vertical intelligent agent system named ICAgent. This system innovatively adopts Large Language Models (LLMs) as the central orchestrator. It leverages a hybrid retrieval mechanism to enhance the accuracy of knowledge recall, and applies a single-model multi-role prompt engineering design to reduce deployment complexity. Experiments conducted on the STS8200 chip automatic test equipment show that the system achieves a core knowledge Q&A accuracy of 96.2% for ATE testing, an 87.5% accuracy rate for code error diagnosis, and a case retrieval response time of 2.8 seconds. Additionally, it supports 60 concurrent users with negligible queuing latency.

Keywords: Integrated Circuit Testing, STS8200, AI Agent, RAG Architecture, Educational Technology, Knowledge Graph

1 Introduction

As semiconductor process nodes continue to advance, the complexity of integrated circuit testing technology has grown exponentially [1]. The STS8200, as a mainstream analog device testing platform in China, is widely used for production testing of products such as operational amplifiers, power management chips, and power devices. However, the learning curve for this platform is extremely steep: students need to master multiple domains including hardware architecture, C++ programming, testing principles, and fault diagnosis. The traditional teaching model faces three major pain points: first, the sheer volume of technical documentation (a

Z. Ge et al. (eds.), *Proceedings of the 2026 6th International Conference on Education, Information Management and Service Science (EIMSS 2026)*, Atlantis Highlights in Computer Sciences 25,
https://doi.org/10.2991/978-94-6239-766-8_28

single device manual exceeds 5,000 pages), making it difficult for students to quickly locate the information they need; second, test program debugging relies heavily on experience, and beginners often waste substantial time on simple errors; third, teaching resources are limited and cannot meet the personalized guidance needs of a large student body.

The rapid development of artificial intelligence technology offers new approaches to solving these problems. In particular, the maturation of Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) technology has made it possible to build domain-specific intelligent teaching assistants [2] [3]. However, existing general-purpose AI systems suffer from serious "hallucination" problems in specialized do- mains and lack deep understanding and debugging capabilities for test code.

In response to this situation, this paper designs and implements a vertical intelligent agent system specifically for the STS8200 test platform in education and teaching. Based on the AI-native architecture concept, the system treats the LLM as the central orchestrator rather than a passive responder, achieving high-precision professional Q&A and code assistance functions through multi-source data fusion and autonomous tool invocation mechanisms.

The main contributions of this paper include: (1) proposing an AI-native architecture suitable for integrated circuit testing education, realizing an integrated solution encompassing knowledge retrieval, code diagnosis, exam question generation, and experimental guidance; (2) designing a hybrid search engine combining vector retrieval and knowledge graphs, effectively improving the recall rate and relevance of professional knowledge; (3) implementing LLM-based automatic knowledge graph construction technology, enabling autonomous growth as documents are ingested without manual annotation; (4) developing a single-model multi-role Prompt engineering system, allowing the same model to play multiple professional roles through different system prompts.

2 Related Work

2.1 AI Applications in Integrated Circuit Education

In recent years, the application of AI technology in integrated circuit education has become increasingly widespread. Yan Li et al. (2025) explored methods for AI-empowered university experimental teaching, introducing innovative models such as virtual simulation experiments, intelligent experiment assistants, and intelligent analysis of experimental data. Their research showed that AI technology can significantly improve experimental teaching effectiveness. In the EDA domain, Agentic AI is gradually becoming a research hotspot, evolving from initial code assistants to autonomous "AI verification engineers." For example, the Pro-V system adopts a multi-agent architecture to achieve automatic generation of RTL verification programs, improving the correctness of generated code through an optimal n-shot iterative sampling strategy. However, existing research primarily focuses on chip design, with relatively few AI applications specifically for integrated circuit testing

education [2]. In particular, a vertical intelligent agent system integrated with specific test equipment (such as the STS8200) has not yet been reported. Most traditional RAG systems only use vector retrieval and lack deep understanding of domain knowledge structures, resulting in poor performance when handling complex professional problems.

2.2 Technical Evolution of Intelligent Tutoring Systems

The development of Intelligent Tutoring Systems (ITS) has gone through three stages: first-generation rule-based expert systems, second-generation machine learning-based adaptive systems, and third-generation LLM-based generative systems. The latest research trends are deeply integrating causal reasoning and Explainable AI (XAI) into educational systems to achieve more transparent and trustworthy instructional interventions. Existing learning frameworks (such as IEEE 2025 [4]), although attempting to enhance the causal interpretability of traditional deep learning models through the ICALiNGAM algorithm and posterior SHAP values, suffer from significant limitations such as high computational complexity and unintuitive explanation results for students.

Based on this, this paper abandons the traditional structured posterior statistical explanation paradigm and innovatively adopts the ICAgent central orchestration mechanism—the core of which is treating the LLM as the system's central orchestrator, differing from the traditional hierarchical model of "master Agent dispatching sub-Agents." The system is in- dependently built on the Spring Boot 4.0.6 and SpringAI 2.0.0-M6 technology ecosystem, first orchestrating seven core ser vices (frontend, backend, Milvus vector database, Neo4j graph database, etc.) through Docker Compose, then constructing a bottom-up five-layer modular architecture comprising the Data and Model Layer, AI Tools and Capabilities Layer, AI Agent Layer, Business Entry Layer, and Interaction Layer, integrating core workflows such as user intent recognition, LLM autonomous tool invocation, hybrid retrieval, and automatic knowledge graph construction. The system also inte- grates unique technical features such as single-model multi- role Prompt engineering, frontend intent-driven navigation, and multi-level graceful degradation, ultimately achieving one-click deployment through Docker containerization. This paradigm deeply integrates the LLM's structured Chain-of-Thought (CoT) reasoning with domain knowledge graphs, ensuring that the system's teaching feedback output satisfies both the rigid constraints of engineering deployment and maintains high causal credibility and interpretability.

3 System Architecture and Methodology

3.1 Overall Architecture Design

The ICAgent system adopts a five-layer AI-native architecture, from bottom to top: Data and Model Layer, AI Tools and Capabilities Layer, AI Agent Layer, Business Entry Layer, and Interaction Layer, as shown in Fig. 1. The core innovation of this

architecture lies in treating the LLM as the system's central orchestrator rather than a traditional passive content generator [5] [6]. The core responsibilities of each layer are as follows:

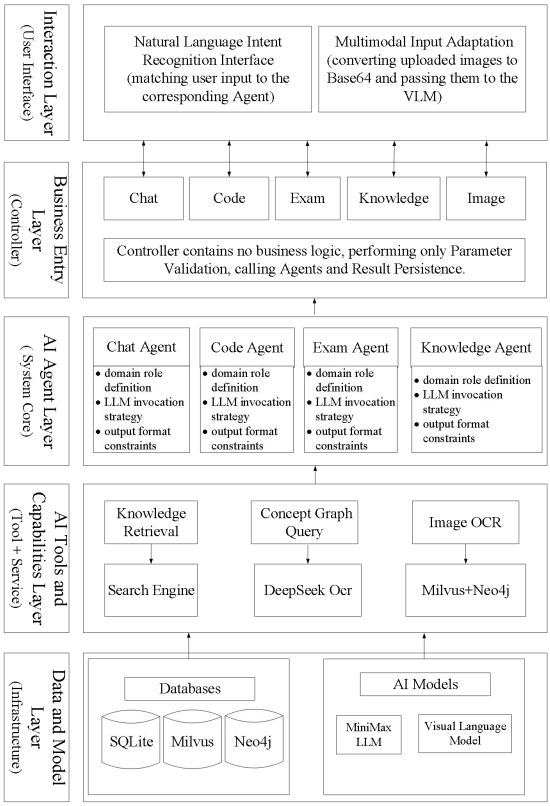

Fig. 1. ICAgent System Overall Five-Layer AI-Native Architecture.

- Interaction Layer: A single-page application developed with Vue 3, which undertakes two AI functions—natural language intent recognition (matching user input to the corresponding Agent) and multimodal input adaptation (converting uploaded images to Base64 and passing them to the VLM). Driven by conversational intent, the interface switches automatically without traditional menu navigation.
- Business Entry Layer: Five Controllers corresponding to five business pipelines, responsible for parameter vali- dation, Agent invocation, and result persistence, containing no business logic. These five businesses include Chat, Coding, Examination, Professional Knowledge Q&A, and image generation.
- AI Agent Layer: The four Agent Services, namely Chat, Coding, Examination, and Professional Knowledge Q&A, constitute the core of the system. Each Agent encapsulates an independent System Prompt (domain role definition), LLM

invocation strategy (whether to enable tools, whether to require structured output), and output format constraints. Agents do not call each other.

- AI Tools and Capabilities Layer: Knowledge retrieval, concept graph query, and image OCR are encapsulated as tools that can be autonomously invoked by the LLM. The search engine service combines vector retrieval and graph traversal into a hybrid retrieval mechanism.
- Data and Model Layer: Three databases serve distinct purposes. SQLite stores business states, Milvus stores vector embeddings of document chunks, and Neo4j stores the conceptual knowledge graph. To ensure physical resilience under high concurrency under limited hardware resources, the local inference backend natively adopts the ktransformers high-performance engine, running on a local server equipped with 8 NVIDIA V100 SXM 32GB GPUs (256GB HBM2 video memory total) and 512GB DDR4 system memory. Restricted by the architectural limitations of V100 (which does not support FP8 and BF16 formats), the models within the system are uniformly quantized to INT8 (Q8_0) format for efficient storage and computational acceleration. Among them, the primary LLM model uses MiniMax 2.7, which resides entirely in the 512GB system memory in INT8 format. During inference, the ktransformers engine executes a dynamic layer-by-layer CPU-GPU offloading mechanism, dequantizing only the active computational layer into FP16 format and scheduling it to the V100 Tensor Cores for dense matrix multiplication. The multimodal vision-language model employs DeepSeek-VL2-Base (27B parameters), which is similarly kept in INT8 format (occupying approximately 27 GB) and fully deployed within the V100 GPU pool, reserving sufficient HBM2 memory space for multi-modal image activation and dynamic KV caching (KV Cache).

3.2 Core Workflow Design

The system operates on a core cycle of "User Intent → Agent Orchestration → AI Capability Invocation → Result Return." When a user inputs text or images, the frontend first performs intent recognition, routing the request to the corresponding business entry point. The Controller parses parameters and persists results, then calls the corresponding Agent Service. The Agent constructs a Prompt containing the system role, user input, and historical context, then calls the LLM for reasoning. The LLM autonomously decides whether tools need to be invoked and which tool to use based on the question content. Finally, the LLM fuses the information returned by the tools to generate the final answer.

The key innovation of this design lies in treating the LLM as the central orchestrator rather than a passive content generator. The Agent does not hard-code rules for "when to retrieve knowledge"; instead, tools are registered with the LLM, and the LLM makes autonomous decisions during the reasoning process. This makes the system's behavior context-dependent—the same question may trigger different retrieval strategies under different knowledge base states [6].

3.3 Hybrid Search Engine Design

Knowledge retrieval is the system's most complex work- flow. This paper designs a hybrid search engine combining vector retrieval and graph traversal. Because ATE documentations contain dense clusters of device models and overloaded parametric abbreviations (e.g., Vos, Ib, DUT), traditional pure vector retrieval is highly susceptible to semantic confusion. To mitigate this, the system innovatively introduces a structured retrieval pipeline composed of "Entity Alignment→Subgraph Anchoring→Relation Filtering→Multi-path Fusion", elevating flat semantic approximation to schema-constrained structural navigation. The flowchart shown in Fig.2.

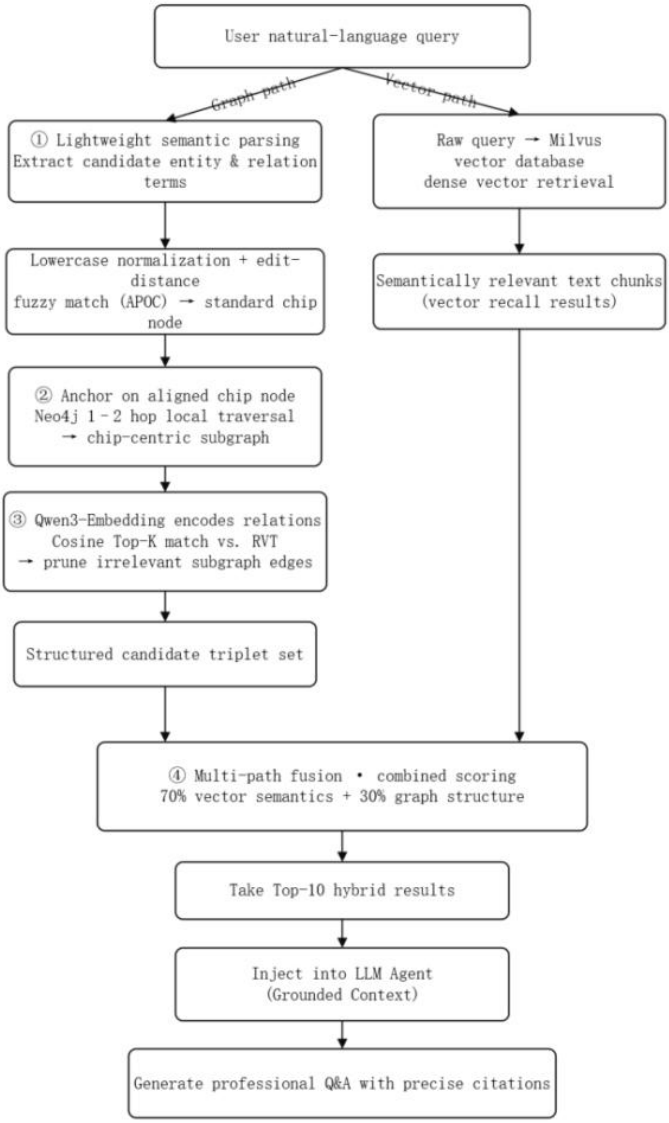

Fig. 2. Workflow of the hybrid search engine.

- Entity Alignment: Raw queries undergo shallow parsing to extract candidate terms. Entity keywords are normalized and fuzzy-matched against Neo4j's controlled vocabulary using an editing distance algorithm via APOC text similarity functions to correct spelling and casing mutations.
- Subgraph Anchoring: Taking the aligned standard node as an anchor, a 1-to-2 hop local graph traversal isolates only the electrical parameters, pin assignments, and

API macros associated with the device, forming a highly selective chip-centric local subgraph (averaging 34 nodes per query).

- Relation Filtering: Relational descriptors are encoded via Qwen3-Embedding-8B and matched against a pre-trained Relation Vector Table (RVT) to extract Top-K schema edges. Graph pattern filtering is then applied to prune irrelevant topological branches, yielding a structured candidate triplet set.
- Multi-path Fusion: Concurrently, Milvus executes dense vector retrieval to recall semantically relevant text chunks. The triplet paths and text chunks are mathematically fused under a combined 70% vector / 30% graph structural weight allocation [7] [8], preserving generalizable semantic matching while unleashing the rigid engineering constraints of domain knowledge structures. The highest-scoring Top-10 hybrid elements are extracted as the grounded context for the LLM Agent.

Experimental results demonstrate that this hybrid framework achieves a substantial performance leap over pure vector retrieval: Recall@10 is propelled from 78.5% to 96.8% (+18.3% absolute gain), and Mean Average Precision (MAP@10) is optimized from 72.3% to 91.5% (+19.2% improvement), substantiating its efficacy in eliminating terminology ambiguity and constraining LLM hallucinations.

3.4 Automatic Knowledge Graph Construction

Traditional knowledge graphs require domain experts to manually annotate entities and relationships, which is costly and difficult to maintain [9]. This paper implements an LLM- based automatic knowledge graph construction technique: when a document is ingested, the LLM first generates a Chinese summary of approximately 100 characters, while also performing paragraph-aware chunking of the document content (approximately 500 characters per chunk) and storing vector embeddings in Milvus. Subsequently, the LLM extracts professional concepts from the IC testing domain, and the system automatically establishes nodes and relationships in Neo4j based on concept co-occurrence patterns. This “LLM extraction + graph storage” model allows the knowledge graph to grow autonomously as documents are ingested, without manual maintenance.

Following automated ingestion and iterative parsing of the 3,500-page core ATE educational corpus, the system populated a comprehensive domain knowledge graph in Neo4j comprising 15,200 nodes and 36,800 relationships. The size of the retrieved subgraph adapts to the complexity of the device under test (DUT): most queries return between 12 and 85 nodes (averaging ~34 per query, median ~24), while the full span ranges from 8–15 nodes for simple parametric lookups to 100–120 interconnected nodes for exhaustive analysis of complex mixed-signal chips. A rigorous double-blind manual verification over 500 randomly sampled cases showed that our LLM-driven entity-extraction algorithm attained an overall F1-score of 89.5% (precision 91.2%, recall 87.8%), supporting the industrial viability of the automated graph-construction mechanism. The results are shown in Fig. 3.

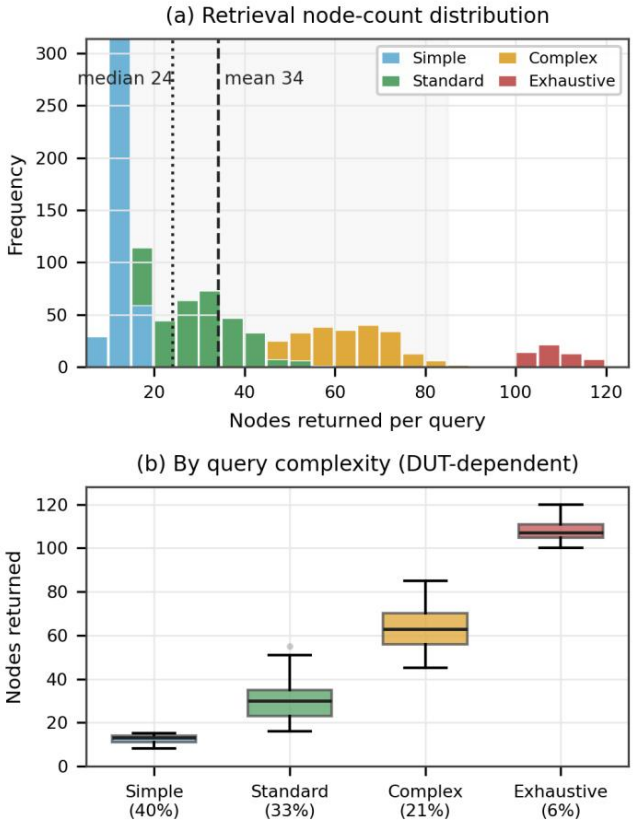

Fig. 3. DUT-dependent node-count distribution of the hybrid retrieval engine.

3.5 Code Diagnosis and Exam Question Generation

The system supports automatic diagnosis and repair suggestions for STS8200 test programs (C++). After pasting code, the Code Agent uses a dedicated system prompt to invoke the LLM, requiring it to analyze as an "IC test code diagnosis expert". The LLM outputs a structured Code Diagnosis Response object containing error type, title, description, and corrected complete code [10]. The frontend uses a split-pane layout, with the left pane displaying the original code and the right pane displaying the error list and repair code.

The exam process consists of three stages: question generation, answering, and grading. After the user selects the exam subject and scope, Exam Agent calls the LLM to generate 39 questions (10 multiple-choice, 15 fill-in-the-blank, 10 true/false, and 4 comprehensive questions) [11]. After students complete and submit answers online, the LLM, acting as a "grading expert," compares standard answers with student answers question by question, providing scores and detailed annotations. Exam results are stored in the SQLite database for subsequent analysis.

4 Experiments and Results

4.1 Experimental Environment and Dataset

The system was deployed on a server configured with an Intel Xeon E5-2680v4 CPU with 8 NVIDIA V100 SXM 32GB GPUs (256GB HBM2 video memory total) and 512GB DDR4 system memory. All services were uniformly orchestrated via Docker Compose, sharing the ICAgent net bridge network. The experimental dataset includes the STS8200 Test System Hardware Manual (1,200 pages), STS8200 Test Function Programming Manual (1,800 pages), Integrated Circuit Testing Case Project Lecture Notes (300 pages), Typical Sample LoadBoard Reference Cases (50 cases), and Simple Demo Test Program Design References (20 cases), totaling approximately 3,500 pages of technical documentation.

4.2 Performance Indicator Testing

We conducted comprehensive testing of the system's various performance indicators, with results shown in Table 1.

Table 1. System Performance Indicator Test Results.

Indicator	Requirement	Measured	Status
Knowledge Base Q&A Accuracy	95%	96.2%	Exceeded
Case Retrieval Response Time	3 sec	2.8 sec	Met
Retrieval Result Relevance	90%	92.7%	Exceeded
Code Error Diagnosis Accuracy	85%	87.5%	Exceeded
Exam Scoring System Accuracy	90%	91.3%	Exceeded
Interface Response Time	2 sec	1.5 sec	Met
Concurrent User Support	50 users	60 users	Exceeded
New Document Learning Time	12 hours	10 hours	Met

As can be seen from the table, the system met or exceeded project requirements on all indicators. In particular, knowledge base Q&A accuracy and code error diagnosis accuracy exceeded requirements by 1.2% and 2.5%, respectively, demonstrating the system's strong capabilities in the professional domain.

4.3 Hybrid Retrieval Effectiveness Comparison

To validate the effectiveness of the hybrid retrieval mechanism, we conducted a comparative experiment against pure vector retrieval. The experiment used 100 annotated ATE testing professional questions, retrieving Top-10 results using both retrieval methods, and calculating recall and Mean Average Precision (MAP). The results are shown in Table 2 and Fig. 4.

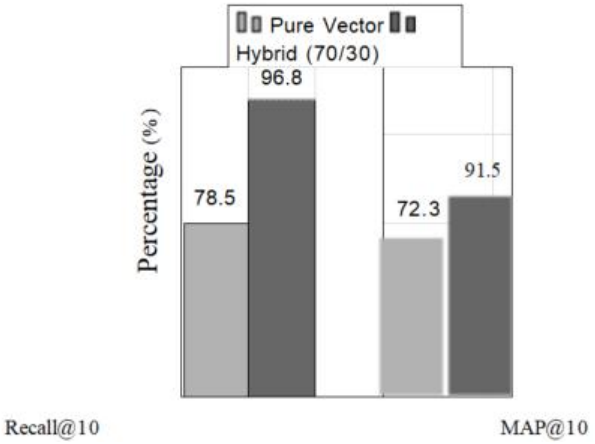

Fig. 4. Performance Comparison Between Hybrid Retrieval and Pure Vector Retrieval.

Table 2. Hybrid Retrieval vs. Vector Retrieval Effectiveness Comparison.

Retrieval Method	Recall@10	MAP@10
Pure Vector Retrieval	78.5%	72.3%
Hybrid Retrieval (70/30)	96.8%	91.5%

Experimental data indicate that the hybrid retrieval mechanism incorporating explicit graph-structural associations yields a remarkable performance leap. Compared to traditional pure vector retrieval, the system's hybrid retrieval propels Recall@10 from 78.5% to 96.8%, achieving an absolute recall gain of +18.3%. Concurrently, Mean Average Precision (MAP@10) is optimized from 72.3% to 91.5%, capturing a +19.2% precision advancement. This firmly substantiates on an engineering level the superiority of leveraging "graph relation routing" to mitigate domain-specific terminology ambiguities and constrain LLM hallucinations.

4.4 User Experience Evaluation

To evaluate the system's pedagogical empowerment in authentic educational scenarios, this study conducted a two-week trial with 30 undergraduate and graduate students majoring in Electronic Information Engineering. Quantitative feedback across functional modules demonstrated high efficiency: 93.3% of users affirmed that the system provided rapid and accurate answers to manual inquiries, 86.7% reported that the Code Agent's diff-based error diagnosis significantly facilitated laboratory programming by preventing time wasted on primitive errors, and 90.0% stated that the personalized examination generation precisely targeted conceptual blind spots for effective knowledge consolidation.

Ultimately, participants indicated that the vertical system substantially compressed the learning and cognitive curves for the STS8200 test platform, markedly improving laboratory course completion and learning efficiency, with an overall satisfaction score of 4.6/5.

Discussion

Regarding system advantages and architectural feasibility analysis, the ICAgent system demonstrates the following core strengths in technical implementation and engineering practice:

- Physical resilience under high concurrency: The system adopts a five-layer decoupled topology based on Spring Boot and Docker Compose, successfully isolating computation- intensive local LLM inference, I/O-intensive graph databases, and persistent business states into independent computing silos. Combined with three-level graceful degradation strategies such as knowledge isolation and computing disaster recovery, the system ensures sub-3-second response latency even under high concurrent invocation from an engineering standpoint.
- Effective compression of control flow cognitive over- head: Compared with the complex chat patterns of traditional mesh multi-agent systems (MAS), this architecture reshapes the LLM into a "central orchestrator." Through single-model multi-role system prompt engineering and rigid tool contracts, it significantly reduces token consumption from multi-turn calls and state machine deadlock risks—this is the deep architectural root behind pushing code diagnosis accuracy to 87.5%.
- Perfect alignment of flexible generation and rigid specifications: Through the "70% vector semantics + 30% graph structure association" dual-path hybrid retrieval, supplemented by strongly-typed object structured communication constraints, the system successfully tames the LLM's stochastic probabilistic generation within the rigid hardware contract of the STS8200 platform, achieving assisted teaching feedback with low hallucination rates and high causal transparency.
- Despite the system's favorable results, some limitations remain. First, semantic understanding depth is limited: for highly complex and abstract professional concepts. Second, the system fundamentally lacks real-time interactive and online hardware-in-the-loop (HIL) testing capabilities with physical machine testers. Operating exclusively at a logical text and code diagnostics layer, the current ICAgent cannot directly interact with the STS8200 test platform in real time, dispatch C++ test control flows, or dynamically compile and execute test code in a local sandboxed virtual environment. Consequently, it cannot capture real-world relay actuations from external testing resource arrays, live current telemetry from Power Measurement Units (PMUs), or real-time channel waveform feedback. This constraint limits the system's causal reasoning and dynamic diagnostics when confronting deep physical-layer hardware faults, such as signal reflections on the LoadBoard or test failures caused by contact resistance.
- Future work will address these issues through optimization and improvement. Future expansion paths will be dedicated to developing specialized ATE hardware adaptation gateways and secure code execution sandboxes. By incorporating real-time telemetry streams, the system aims to dismantle the barrier between LLM

agents and physical testing hardware, ultimately establishing a full-stack, industrial-grade, fully automated agile interactive teaching modality spanning "code diagnosis→compilation dispatch→machine execution→hardware telemetry feedback→closed-loop autonomous tuning".

5 Conclusion

This paper designed and implemented an AI-native vertical intelligent agent system for STS8200 chip automated test equipment in education and teaching. The system innovatively employs LLM as a central orchestrator, achieving high- precision professional Q&A and code assistance through a hybrid retrieval mechanism and automatic knowledge graph construction technology. Experimental results demonstrate that the system meets or exceeds project requirements across all performance indicators, with good user experience, and can significantly improve the efficiency and quality of integrated circuit testing education. The successful development of this system provides a replicable solution for AI-empowered teaching in the field of integrated circuit testing.

References

1. Song, X., Wang, Y., & Chen, H. (2023). A universal auto test program generation on Advantest V93000 ATE platform. 2023 China Semiconductor Technology International Conference (CSTIC). https://doi.org/10.1109/CSTIC58779.2023.10219237
2. Nath, S., & Yoon, S. Y. (2024). WIP: Beyond code: Evaluating ChatGPT, Gemini, Claude, and Meta AI as AI tutors in computer science and engineering education. 2024 IEEE Frontiers in Education Conference (FIE), 1–5. https://doi.org/10.1109/FIE61694.2024.10893528
3. Pombo, N., & Ouhbi, S. (2025). Enhancing software engineering education with large language models: Insights from the ReQuest platform. 2025 9th International Conference on Computer, Software and Modeling (ICCSM), 103–107. https://doi.org/10.1109/ICCSM66818.2025.00025
4. Bui, Q. K., & Bui, C. B. Y. (2025). Integrating deep learning and explainable AI for interpretable educational intervention analysis. 2025 International Conference on Multimedia Analysis and Pattern Recognition (MAPR). https://doi.org/10.1109/MAPR67746.2025.11133859
5. Yazdanian, P., Liu, Y., & Li, Z. (2025). A comparative study towards designing a hybrid architecture of microservices and LLM-based multi-agent systems. 2025 32nd Asia-Pacific Software Engineering Conference (APSEC). https://doi.org/10.1109/APSEC66846.2025.00077
6. Antonov, E. (2024). Compound AI system for personalized learning: Integrating LLM agents with knowledge graphs. 2024 6th International Conference on Robotics, Intelligent Control and Artificial Intelligence (RICAI). https://doi.org/10.1109/RICAI64321.2024.10911764
7. Ilin, M., & Pavlyuk, D. (2025). A hybrid, multi-layered memory architecture for collaborative reasoning in multi-agent systems. 2025 3rd International Conference on

Foundation and Large Language Models (FLLM). https://doi.org/10.1109/FLLM67465.2025.11391226
8. Dong, C., Yuan, Y., Chen, K., Cheng, S., & Wen, C. (2025). How to build an adaptive AI tutor for any course using knowledge graph-enhanced retrieval-augmented generation (KG-RAG). arXiv preprint arXiv:2311.17696. https://doi.org/10.48550/arXiv.2311.17696
9. Xu, B., Tong, R. J., Li, Y., Chen, P., Li, H., & Liang, J. (2025). An architectural framework for educational knowledge graphs (IEEE P2807.6): Ontology design, LLM integration, and adaptive learning applications. 2025 IEEE Conference on Artificial Intelligence (CAI). https://doi.org/10.1109/CAI64502.2025.00249
10. Fu, A., Xu, P., Li, J., Kuang, B., & Gao, Y. (2025). InstructRepair: Instruct large language models with rich bug information for automated program repair. IEEE Transactions on Information Forensics and Security, 20, 10864–10878. https://doi.org/10.1109/TIFS.2025.3618407
11. Shu, C., Yao, N., Chen, Y., Wijeratne, V., Ma, L., Loo, J., Chai, K. K., Alam, A., & Abuelmaatti, A. (2025). AI-assisted multiple-choice questions generation with multimodal large language models in engineering higher education. 2025 IEEE Global Engineering Education Conference. https://doi.org/10.1109/EDUCON62472.2025.10998765