FOUNDATIONAL CONCEPTS: TECHNICAL BRIEF

Case-Based Reasoning

Case-based reasoning solves new problems from records of past problems.

The system stores each solved problem and reuses the record on the next similar problem.

A library of past cases in a loop. A new problem enters the loop at the lower left.

Start Here: The Short Version

This method helps a computer remember old problems and the fixes that worked. Think of each old problem as a case in a file. Each case holds the problem, the fix, and the result. A new problem arrives. The system looks for an old case that looks most like the new problem. The system changes the old fix so the fix fits the new problem. People test the fix and check the result. The system then saves the new case in the file. The next problem starts with one more case to use.

What it can do

- It finds the closest old case fast. The system compares the new problem with past cases and ranks the best matches first.
- It reuses a fix that worked before. Staff start from a proven step and adapt that step for the new facts.
- It learns from each new result. A good result adds a strong case, and a poor result adds a clear warning.

Where people can use it

- A help desk team sees the same questions each week. The system brings back the answer that solved the same question before.
- A server team sees the same faults and alerts again and again. The system brings back the repair steps that fixed the fault last time.
- Lawyers, doctors, and repair staff use past cases in daily work. A lawyer checks old rulings, a doctor checks past patients, and a repair worker checks past jobs.

Problems it can help solve

- Teams waste time when teams solve the same problem again and again. This method saves the fix so the next person finds the fix fast.
- Most search tools find text, but the tools do not learn from the result. This method saves the result, so search gets better with each use.
- New staff do not know the old cases that senior staff hold in memory. This method puts that memory in a shared file that all staff use.

The rest of this brief gives the history, the method, the 2026 gap, and a first project plan.

1 Identification

Roger Schank published dynamic memory theory in 1982.

Janet Kolodner built CYRUS, an early case memory system.

Agnar Aamodt and Enric Plaza defined the four-step cycle in 1994.

The four steps carry the names Retrieve, Reuse, Revise, and Retain.

Teams use the short name 4R for the cycle.

Year	Event
1982	Schank published dynamic memory theory.
Early 1980s	Kolodner built the CYRUS case memory system.
1994	Aamodt and Plaza defined the 4R cycle.
1990s	Commercial helpdesk and diagnosis products used CBR.

CBR in this brief means case-based reasoning. RAG in this brief means retrieval-augmented generation.

2 Purpose

The method treats past experience as the main knowledge source.

A team records each solved problem as a case.

The system reuses a past solution for a new problem.

The system avoids a start from blank facts on each new problem.

The system learns from each new case.

The library grows with use.

The next search starts from a richer base.

The method suits domains with repeat problems.

The method suits domains with clear results.

Support work and infrastructure work fit both tests.

> A case states a problem, a solution, and a result in one record.

3 Operation

A case has three parts.

The first part states the problem.

The second part states the solution.

The third part states the result.

The cycle runs the four steps in order on each new problem.

1. **Retrieve.** The system finds the cases with the highest similarity to the new problem.
2. **Reuse.** The system adapts the old solution to the new problem.
3. **Revise.** The team tests the adapted solution and repairs the solution after a failure.
4. **Retain.** The system stores the new case with the result in the case library.

The Retain step closes the loop.

The next problem searches a library with one more tested case.

Operation: Diagrams

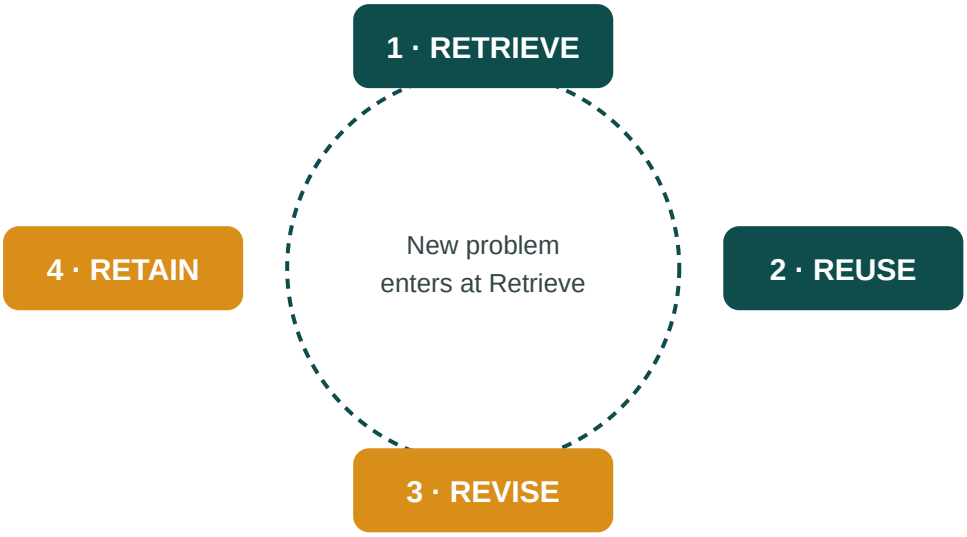

Figure 1: The 4R cycle. Amber marks Revise and Retain, the two steps with the 2026 gap.

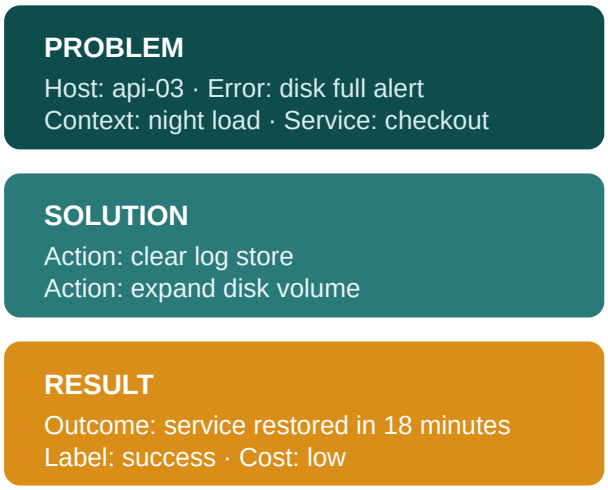

Figure 2: A case record. Three stacked blocks hold the problem, the solution, and the result.

Retrieve reads the Problem block first.

Reuse reads the Solution block next.

Revise and Retain write and use the Result block.

4 History

Commercial helpdesk products used CBR in the 1990s.

Commercial diagnosis products used CBR in the 1990s.

Teams built case libraries by hand.

Teams cleaned case libraries by hand.

Teams adjusted similarity measures by hand for each domain.

Teams authored adaptation knowledge by hand.

Hand work raised cost on each new deployment.

Hand work slowed each new deployment.

Research groups moved to large data methods in later years.

Vendors sold fewer new CBR products in later years.

> The core idea stayed sound in tests and in field use. High cost caused the decline in use.

5 The 2026 Gap

RAG retrieves stored text for a language model.

RAG covers Retrieve and a part of Reuse.

A language model improvises the adaptation in RAG.

No fixed step checks the adapted result against a past outcome in RAG.

No step stores the new case with a result label in standard RAG.

Mainstream agent products do not run Revise and Retain as a learning loop.

Research prototypes combine CBR structure with RAG pipelines in test systems.

A mainstream product with the full 4R cycle does not exist in 2026.

Figure 3: RAG and CBR across the four steps. Grey marks steps with no presence in standard RAG.

6 Project Proposal

Goal. The desired end state is one case memory service in daily use by agents in one incident domain. The service holds solved cases with results. The service completes all four steps of the cycle on every new case.

Proof target. The project proves one claim against a retrieval-only baseline. Agents with the full cycle resolve repeat incidents with fewer steps and fewer wrong fixes. Results improve on the second occurrence of a problem because the service retains the first result.

Product at the end. The product is the Case Memory Service, version 1. The service contains a structured case store, a similarity search, an adaptation step, a result check, and a retention step with success and failure labels. Agent frameworks call the service through one API.

The project builds a case memory service for agents.

The first version serves one domain only.

The team selects support incidents or infrastructure incidents for the first version.

Components

1. **Case store.** The store holds each case in a fixed structure with problem, solution, and result fields.
2. **Similarity search.** The search uses embeddings and field weights and ranks cases by similarity.
3. **Adaptation step.** The step uses a language model and adapts the top case to the new problem.
4. **Result check step.** The step tests the adapted solution and records success or failure.
5. **Retention step.** The step stores the new case with a success label or a failure label.

Output to the agent

The service gives the agent one adapted solution for the new problem.

The service gives the agent one warning from the most similar failure case.

The warning names the past failure in plain terms.

The agent starts work with a plan and a known risk in view.

First trial

1. The team loads past incidents from the target domain into the case store.
2. The team connects one agent to the service.
3. The team runs live incidents through the full 4R cycle.
4. The team measures reuse rate, repair rate, and repeat failure rate.
5. The team compares agent results with the service against agent results without the service.

7 Risks and Limits

A poor case library gives poor retrievals.

Poor retrievals give poor solutions.

A wrong similarity measure ranks the wrong case first.

A language model adapts a case in a wrong way at times.

The result check step misses an error at times.

Teams still need curation work for case quality.

Teams still need curation work for case removal.

Private cases need access controls from the start.

Private cases need clear data rules from the start.

The method serves repeat problems well.

The method serves novel problems poorly.

A new domain starts with a thin library and thin results.

Results improve after the library gains tested cases.

> A team tests value in a small domain trial first. A team builds a broad version after proof in one domain.

Sources

Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(1).

Schank, R. C. (1982). Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge University Press.

Kolodner, J. L. Work on CYRUS and case-based reasoning, Georgia Institute of Technology, 1980s and later.

Project source report: Foundational Concepts series research, section 5, Case-Based Reasoning, 2026-10-23.