Research Technical Report FRITS-TR-2026-05 8 July 2026 Updated 11 August 2026 SUPPORT
ARCHITECTURE AUTOMATION SELF-HEALING HUMAN-IN-THE-LOOP AGENTS

Support at the Scale of Everyone: A Self-Healing Architecture Where AI Answers In Place and One Person Approves

Frits Lyneborg
FRITS AI ApS

ABSTRACT

Customer support does not scale: double the users and you double the people answering, or the answers get slower and worse. This report describes an architecture that breaks that link, built on one structural decision — a human is spent only where a human is irreplaceable, on genuinely new problems, and everything else is absorbed by AI in layers. AI writes the help material; AI answers from it, in place, on every page; when AI cannot answer, it does not hand the user to a queue but helps them write one good public post, which becomes the unit of work. An overnight agent drafts fixes and answers for those posts and, on approval, feeds each resolved answer back into the specific assistant that failed, so the same question is never escalated twice. A separate pass clusters the questions assistants answer correctly but repeatedly, turning "asked 200 times this week" from invisible load into a ranked list of things to design away. The single human is not the bottleneck but the approver. This is an architecture report, not a benchmark, and we state plainly what is measured and what is not.

1. The problem, and the structural fix

Customer support scales linearly with users, and that is the whole problem. The number of people asking questions grows, so the number answering must grow too, or answers get slower, shallower and more grudging. Automation has been sold as the

escape for a decade, mostly dishonestly: the deflection chatbot that exists to stop the user reaching a human, that cannot answer, and that hides the “talk to a person” link two menus deep. Users have learned to distrust it. The wall is still there.

The escape is not a better deflection bot. It is a change in what the human is spent on.

In a support operation, almost every case is a variation of one already solved. Only a small residue is genuinely new. **Spend the human only on that residue — and let AI absorb, answer and remember everything already solved — and the human cost stops tracking the number of users and starts tracking the rate of genuinely new problems, which grows far more slowly.** That residue does not double when users double. That is the lever.

Three commitments follow. Answers happen **in place**: a user asks for help on the page they are on and is answered there, never redirected into a support silo. The human is **never the front line**: layers of AI absorb everything answerable, and the one person is reserved for the residue, with the best possible conditions to act on it. And the system **heals itself**: every time an assistant fails, the resolution is fed back into that specific assistant; every time an assistant succeeds too often at the same question, that success surfaces as a product problem to fix at the source.

2. The shape of the system

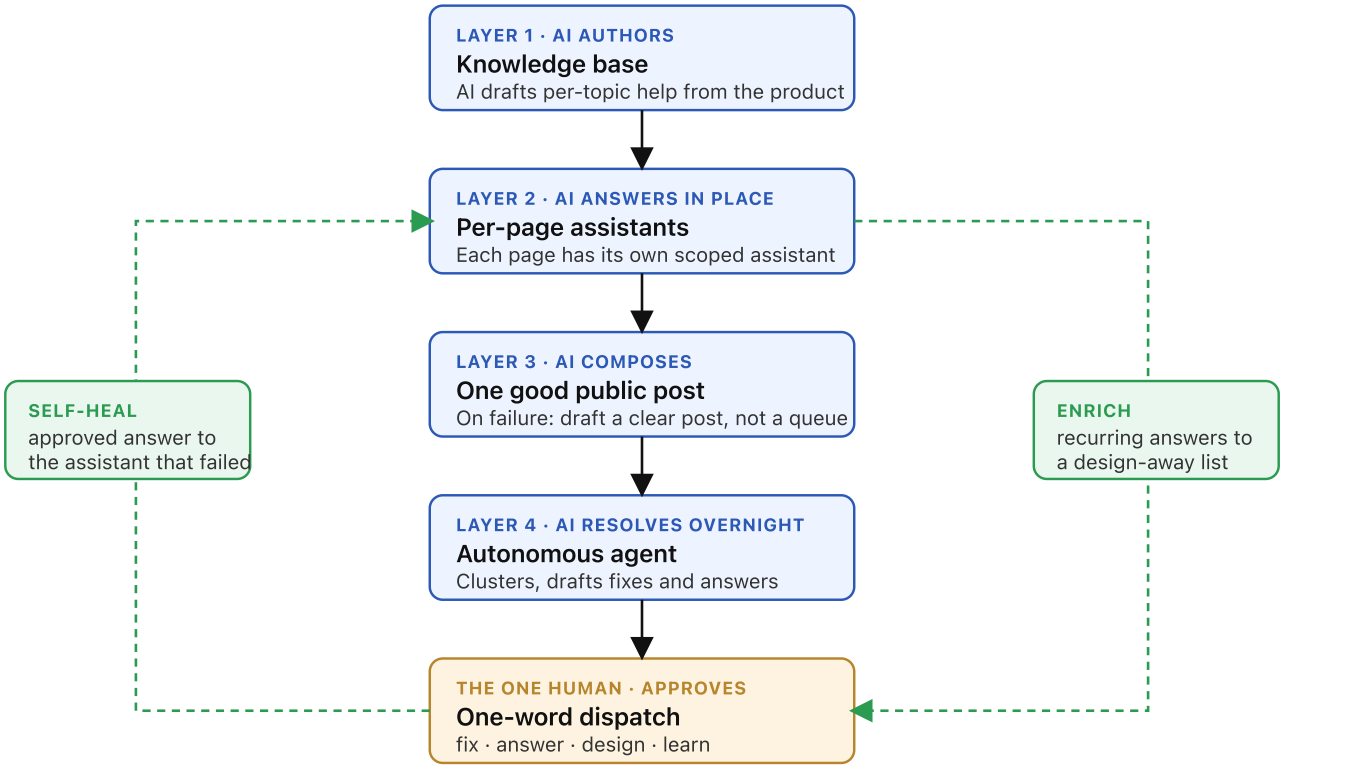

Figure 1. The pipeline and its two loops. Solid arrows: a user's question flowing down through the AI layers to the single human approver. Dashed loops: self-heal (every approved answer returns to the assistant that failed) and enrichment (every question answered too often becomes a product fix).

The rest of the report is that diagram, one box at a time.

3. Layer one: AI writes the knowledge before any user arrives

A support assistant is only as good as what it knows, and hand-writing a knowledge base for a product that changes weekly is a job nobody finishes. So the first use of AI sits upstream of any user: AI drafts the help material from the product's own description of what it does, one document per topic. Those documents are embedded into a vector index so they can be retrieved by meaning rather than keyword.

Retrieval by meaning matters more than it sounds. A support product serving a multilingual population is asked the same thing in dozens of languages and a hundred phrasings, and keyword matching fails the moment the user's words differ from the

document's [4 (#ref-4)]. Semantic retrieval is what lets one authored document answer a question nobody phrased the way it was written.

The knowledge base then keeps changing, because layer four writes back into it. It begins as an AI-authored draft and becomes, over time, a record of every question that turned out to matter.

4. Layer two: assistants that answer in place, on every page

The user never goes to "support". Support comes to the user, on the page they are already on. Every significant page carries its own assistant, scoped to that page's topic but able to draw on the whole knowledge base for anything off-topic. Ask about price on the pricing page and the pricing assistant answers; ask it about account settings and it still answers, from the shared index.

Two design choices matter more than they appear.

The assistant answers from retrieved knowledge, and is built to hand off rather than guess. The failure mode of a naive help bot is confident fabrication — an answer that sounds right and is wrong. This one is instructed to answer only from what retrieval returns and, when retrieval returns nothing matching, to say so and escalate. That is enforced structurally rather than requested [3 (#ref-3)]: the escalation signal is honoured wherever the model emits it, and the "same topic" test is explicit, so a loosely related document cannot be stretched into a wrong answer.

Right-sizing keeps this affordable. Answering hundreds of thousands of support questions with the largest available model would be wasteful and slow. The assistants run on smaller models sized to the task, escalating only when a question genuinely needs it — the same right-sizing argument, and roughly the same order-of-magnitude saving, we set out for chatbot answers generally [2 (#ref-2)]. Support questions are overwhelmingly the easy majority that small models answer indistinguishably well.

Most questions are therefore answered instantly, in place, by an assistant that knew the answer. The ones it does not know do not dead-end.

5. Layer three: when AI cannot answer, it composes one good post

This layer inverts the usual deflection model.

When the assistant genuinely cannot answer, it does not open a private ticket and it does not surface a “contact a human” link. It helps the user write one clear, public forum post, and does the writing — reading the conversation so far, drafting a title and a well-formed post with the page context baked in (“On the pricing page, I was trying to…”), showing it to the user to edit, stating plainly that it will be public, and posting under their chosen name once they confirm. What the user is spared is the labour of writing a good bug report.

Three things fall out of the inversion, and each is load-bearing.

The public post is the unit of work. Because the escalation is a well-formed post rather than a terse private ticket, it carries everything a resolver needs: the question, the page, the conversation that led there, the user’s own words. A queue of these is a queue of solvable problems rather than a queue of “it doesn’t work”.

Public means the next person may never have to ask. A private ticket helps exactly one user. A public, searchable post helps everyone who later has the same question, and becomes training material for the assistant that failed.

A private path still exists for private problems. Anything about a user’s own data escalates to a team-only report instead, and a detector forces anything containing personal or payment data down the private path regardless of which page it came from. The routing can only ever make a post *more* private, never less, so a misclassification cannot leak.

6. Layer four: the overnight agent, and the two loops

Every escalation, public posts and private reports alike, flows into a single queue. Once a day an autonomous agent works it, and this is where the system stops being a smart help desk.

The agent **clusters** same-root-cause reports across languages, so twenty reports of one bug are one item. It **drafts** the work — a code fix for a bug, a written answer for a question the knowledge already covers. It **routes** each item to the right kind of response. And for every resolved question it **proposes feeding the answer back into the specific assistant that failed to answer it.**

Loop one — self-heal. When a page's assistant could not answer a question and that question gets answered, the answer is written back into that assistant's knowledge as a new entry. The next user to ask is answered in place, instantly, by the same assistant that failed the first user. Failures are non-recurring by construction: the front line gets smarter at exactly the points where it was found wanting, every night. An escalation becomes a lesson the system keeps.

Loop two — enrichment. A separate pass looks not at what the assistants failed to answer but at what they answered *correctly and repeatedly*. It clusters successfully-answered questions over a rolling window and surfaces the ones asked over and over. This is the signal every support operation has and almost none uses. **An assistant answering the same question two hundred times a week is not a success. It is two hundred votes that the product should have made the answer obvious.** A question asked that often should stop being asked, by fixing the product rather than answering faster.

Together the loops pull in opposite directions on purpose. Loop one drives the *unanswerable* count toward zero by teaching the assistants. Loop two drives the *repetitive* count toward zero by removing the reasons to ask. What is left in the middle — genuinely new problems, asked once — is exactly the residue the human exists for.

7. The single human: an approver with optimal conditions

In a conventional operation the human is the front line, seeing raw, unclustered, unwritten complaints in every language at the rate users produce them. That is the job that does not scale.

Here the human sees none of that. By the time work reaches them it has been answered where possible, escalated only where not, clustered so duplicates are one item, drafted so each arrives with a proposed fix already written, and translated into their language. It is presented once a day as a ranked briefing and dispatched by a controlled vocabulary of one-word verbs — approve a fix, approve an answer, approve teaching an assistant, approve a product change.

The human cost scales with the rate of genuinely new, judgement-requiring problems — not with the number of users, and not with the number of questions. The person is not doing less important work than a support team. They are doing *only* the most important work a support team does, with everything else cleared away.

Nothing ships unattended. The overnight agent drafts; it does not act. Approval is required for every fix, every answer, and every write-back into an assistant's knowledge, so no automated answer reaches a user without a human having said yes. Automation does the labour; the human keeps the judgement.

8. What this is, and what it is not

This is an architecture report describing a system in production, not a benchmark study.

We report a design, not a controlled measurement of it. The claim that one person can stand behind support at very large scale is a claim about the *structure* of cost — that the human's load tracks the rate of novel problems rather than the volume of users — not a time-series of one person doing so across hundreds of thousands of cases. A longitudinal measurement of deflection rate, self-heal rate and human-minutes per thousand users is future work, and the interesting number is how fast the "genuinely new" residue actually grows with users. We do not claim a measured headcount ratio.

Self-healing is only as good as retrieval and clustering. Loop one depends on the failing assistant actually retrieving the newly written answer next time; phrased unlike the questions users ask, it will not surface. Both loops rest on meaning-based matching across languages [1 (#ref-1)], and both degrade gracefully — a missed heal is one more

escalation, not an outage. Separately, if layer one drafts a subtly wrong help document, the layers above will confidently serve and reinforce it until a human catches it. The approval gate is the safeguard, but the initial corpus warrants human review wherever it carries legal or financial weight.

The approval step is a real dependency, by design. A system requiring one human's daily approval is not autonomous and is not meant to be; if that person is unavailable, drafted work waits. That trades throughput for safety, which is right for support touching billing, personal data and trust — but it makes the single human a single point of latency, even though they are not a single point of labour.

Scope. The system serves one product's support surface. Whether the architecture transfers to domains where most cases are genuinely novel is untested, and the lever would not apply there. This report is self-published by FRITS AI ApS and is not peer-reviewed.

9. Conclusion

Support has always scaled with users because the human was the front line. Move the human to the back, to the single point where judgement is irreplaceable, and put layers of AI in front. Wrap that in two loops — one teaching the front line its own failures, one turning questions answered too often into products fixed at the source — and the queue that reaches the human is small, clustered, pre-drafted, and shrinking.

Users are answered where they stand. The human is spent only where a human must be, and the system gets quieter the more it is used.

References

1. Lyneborg, F. — *Translating by Meaning, Not by Words: A Two-Stage Method for Native-Quality Machine Translation.* FRITS AI ApS, Technical Report FRITS-TR-2026-02, 2026. https://frits.ai/research/meaning-first-translation/

   (https://frits.ai/research/meaning-first-translation/)

2. Lyneborg, F. — *Right-Sizing the Model to the Question: Cutting AI Chatbot Energy Without Losing Quality.* FRITS AI ApS, Technical Report FRITS-TR-2026-04, 2026. https://frits.ai/research/greener-ai-right-sizing/ (https://frits.ai/research/greener-ai-right-sizing/)
3. Lyneborg, F. — *Too Many Tools Break Mid-Size Models: A Two-Stage Method for Reliable Tool Use.* FRITS AI ApS, Technical Report FRITS-TR-2026-03, 2026. https://frits.ai/research/two-stage-tool-routing/ (https://frits.ai/research/two-stage-tool-routing/)
4. Lewis, P., Perez, E., Piktus, A., et al. — *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* Advances in Neural Information Processing Systems (NeurIPS), 2020. arXiv:2005.11401

How to cite

Frits Lyneborg (2026). Support at the Scale of Everyone: A Self-Healing Architecture Where AI Answers In Place and One Person Approves. FRITS AI ApS, Technical Report FRITS-TR-2026-05. https://frits.ai/research/support-at-the-scale-of-everyone-a-self-healing-architecture/ doi:10.5281/zenodo.21267980.

```
@techreport{lyneborg2026support,
  title       = {Support at the Scale of Everyone: A Self-Healing Architecture Where AI Answers
  author      = {Lyneborg, Frits},
  institution = {FRITS AI ApS},
  number      = {FRITS-TR-2026-05},
  year        = {2026},
  month       = {jul},
  doi         = {10.5281/zenodo.21267980},
  url         = {https://frits.ai/research/support-at-the-scale-of-everyone-a-self-healing-arch
}
```