EPiC Series in Computing

Volume 109, 2026, Pages 41–51

Proceedings of EUNIS 2026 Annual Congress

Ritchy – Federated AI Support: Quality Assurance, Architecture, Operations

Ingo Hengstebeck[1*], Bernd Decker[1†], Sarah Grzemski[1‡] and Sara Erdem[1§]
[1] RWTH Aachen University, Germany
hengstebeck@itc.rwth-aachen.de, decker@itc.rwth-aachen.de,
grzemski@itc.rwth-aachen.de, erdem@itc.rwth-aachen.de

Abstract

We present Ritchy, an AI-powered support chatbot that has been in productive use at the IT Center of RWTH Aachen University since spring 2025. Ritchy combines a large language model (LLM) with retrieval-augmented generation (RAG) grounded in curated documentation synchronized from the knowledge management system Sabio. Based on operational data and an ISO 9001-embedded PDCA process, we show that user feedback alone is too incomplete and sometimes biased to serve as a reliable quality indicator, making continuous human quality assurance essential. From these findings, we derive key requirements for federated deployment—multi-tenancy, local content ownership, and quality assurance as a core function—and propose a reference architecture separating a shared platform layer from the university-specific layer. We further outline three deployment scenarios (all-in-one, fully configurable, and open source) and corresponding cost models.

1 Introduction

In order to meet the increasing demands on university IT service organizations, the IT-ServiceDesk of RWTH Aachen University's IT Center put the AI-supported chatbot Ritchy ("RWTH IT Center Helps You") into productive operation in spring 2025. The primary motivation was to provide fast, low-threshold, and largely continuous support—effectively aiming for 24/7 availability. At the same time, IT service organizations are facing structural challenges: service portfolios continue to expand and increasingly include services that are used across multiple universities (Grzemski & Hengstebeck 2017).

* https://orcid.org/0009-0006-0592-4925
† https://orcid.org/0000-0002-9627-5695
‡ https://orcid.org/0000-0002-3946-4780
§ https://orcid.org/0009-0002-2531-4164

L. Desnos, C. Diaz, J. Mincer-Daszkiewicz, L. Merakos, R. Vogl, S. McLellan and U. Lucke (eds.), EUNIS 2026 (EPiC Series in Computing, vol. 109), pp. 41–51

At RWTH's IT Center, the number of services supported by the IT-ServiceDesk grew from 7 (2010) to 109 (2025) (Bischof et al. 2011, Grzemski et al. 2025a). Simultaneously, the IT-ServiceDesk potentially serves around 55,000 users (approximately 45,000 students and 10,000 employees), as well as additional cooperation partners (RWTH Aachen University 2024). In 2024, this resulted in a support volume of about 65,000 requests across multiple channels (email, phone, in-person, chat, and ticket portal). An initial analysis of incoming requests in 2024 shows that around 1,200 phone calls and approximately 9,000 emails were received outside regular service hours. Since these requests can only be processed on the next working day, they lead—especially at the beginning of the week—to backlogs and a decline in perceived service quality (Grzemski et al. 2025b).

A qualitative review of requests received outside service hours indicates that they predominantly involve recurring standard issues, such as password resets, Wi-Fi configuration (eduroam), VPN access, or questions about software installation. While these issues can often be resolved quickly, they still consume valuable capacity of qualified support staff, reducing availability for more complex problem solving. Against this backdrop, Ritchy was established as an AI-based support augmentation: the chatbot answers standard questions based on documented information and guides users toward self-service resolution (Grzemski et al. 2026).

In the following, we report on insights from roughly one year of productive use and outline scenarios for how an AI support chatbot can be made available to other universities in a federated structure.

2 Ritchy: Technical Implementation

Ritchy is built on a large language model (LLM) combined with retrieval-augmented generation (RAG). The goal is to generate answers exclusively from the underlying documentation. To achieve this, relevant passages are retrieved from a curated knowledge base and used as context for answer generation. Therefore, the answer quality depends directly on the freshness, consistency, and completeness of the documentation. This documentation-bound approach is intended to minimize hallucinations. However, while RAG can reduce hallucinations, it cannot compensate for deficiencies in the knowledge base itself (Lewis et al. 2020). Outdated articles may lead to answers that are formally correct with respect to the documentation but are factually obsolete. If the documentation is contradictory, retrieval may surface conflicting passages without the model being able to reliably determine which version is technically correct. In such cases, Ritchy is designed to make the lack of clarity explicit, refer to sources, and direct users to alternative support channels (Grzemski et al. 2026). These limitations are addressed in operation via the review/PDCA process, which classifies gaps, contradictions, and outdated content as error categories and feeds them back into documentation maintenance and configuration updates.

The knowledge base originates from the Sabio knowledge management system, which maintains the IT Center's public documentation (help.itc.rwth-aachen.de) (approximately 3,334 articles). Following a nightly export, articles are cleaned and structurally segmented, then uploaded to Azure Blob Storage. Embeddings are generated from this content and indexed in Azure AI Search, enabling retrieval to reliably provide matching passages (Grzemski et al. 2026).

To produce traceable and consistent outputs, Ritchy relies on system prompts and guardrails, complemented by conservative generation parameters (e.g., temperature = 0) (Grzemski et al. 2025a). Answer generation is governed by a multi-layer prompt rule set: system instructions define role, tone (including formal address consistent with the user's language), and safety boundaries. A grounding instruction enforces strict source binding to the conversation context and the document passages provided via RAG. If the available evidence is insufficient, Ritchy asks targeted follow-up questions and, if necessary, switches to a standardized fallback that points users to alternative support channels

to contact the support staff of the IT-ServiceDesk. Output is standardized via formatting rules (e.g., paragraphs/lists and step-by-step guidance, without headings) to ensure consistent and easily reviewable responses. Links are restricted to canonical URLs through allowlist/denylist rules. Prompt templates and retrieval parameters are versioned and maintained via the review and release process.

Ritchy is implemented as a public web application. A backend validates and normalizes user inputs, assembles a context-bound prompt, and forwards requests to Azure services. Retrieval and answer generation are performed in Azure using Azure AI Search and Azure OpenAI. Users can rate responses using simple feedback mechanisms (thumbs up/thumbs down) (Grzemski et al. 2026).

Since automated quality indicators (especially pure user feedback) are only of limited reliability in practice, operations are supplemented by a continuous human-driven quality process (PDCA) that systematically identifies errors and feeds improvements back into both documentation and configuration (Grzemski et al. 2025a).

In addition to the public chatbot, the IT Center operates an internal assistant for IT-ServiceDesk staff, called SeKoGPT (Grzemski et al. 2025b). Unlike Ritchy's anonymous access model, SeKoGPT requires authentication and incorporates internal knowledge sources such as policies, troubleshooting procedures and escalation paths. Apart from login and authentication, SeKoGPT uses the same infrastructure as Ritchy. This operating model highlights the key requirements for federated deployments, such as federated identity management as well as role and permission models,

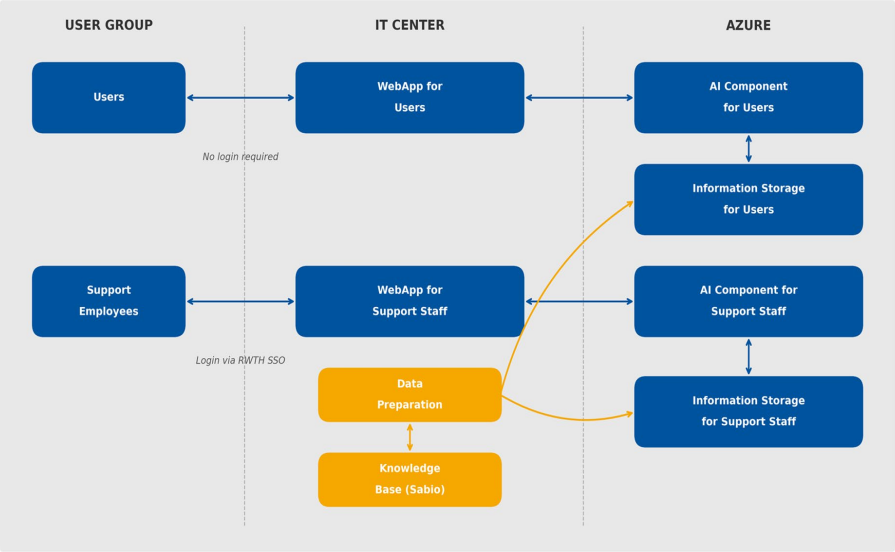

Figure 1: Infrastructure (Grzemski et al. 2026)

particularly to control access to knowledge sources and to separate quality assurance processes on a per-university basis. The largely common technical basis also means that the provision scenarios described in this article can also be applied to SeKoGPT.

The strict separation of user interface, application logic, retrieval, and the LLM simplifies maintenance and evolution of the system. At the same time, it provides a robust foundation for federated operating models, even when participating universities choose to run individual components differently.

3 Ritchy: Operational Experience and Quality Assurance

An analysis of Ritchy usage between February 1, 2025, and January 31, 2026, comprising 12,922 user messages and 12,868 bot messages across 4,586 chats shows that identity management is the dominant topic, followed by high-performance computing (HPC), RWTH email, and software-related questions. The service category "No IT Center service" captures requests that are not handled by the IT-ServiceDesk but by other units at RWTH Aachen University. This distribution analysis supports, among other things, targeted improvements of documentation in those areas where the chatbot is most frequently used in practice.

The following table summarizes the five most frequently requested services. User messages are mapped to services using a Python script that performs basic normalization (e.g., lowercasing and removing special characters) and then matches each message against service-specific keyword lists. If a message matches multiple services, it is assigned to the service with the highest number of keyword matches. Messages with no matches are classified as "Unmatched." The current match rate is approximately 80%, i.e., the share of user messages that can be assigned to at least one service.

Service	User Messages
IdentityManagement	2,521
High Performance Computing	1,036
RWTH E-Mail	901
Software	859
No IT Center Service	404

Table 1: Top 5 Services

Nevertheless, operational experience shows that user feedback alone is not sufficient for reliably measuring answer quality (Kakur et al. 2024). Although users can rate bot responses using a thumbs-up/down mechanism, only about 10% (1,286) of bot messages (out of 12,868 in the observed period) receive any feedback at all. Therefore, there is no immediate user-based quality signal for roughly 90% of responses.

Metric	Value
Number of chats	4,586
Number of bot messages	12,868
User messages per chat (median)	2
Bot messages with feedback	1,286
Positive feedback after QA (corrections and post-hoc ratings of bot messages without user feedback)	78 %
Manually rated bot messages (bot messages without user feedback)	10,346
Manual corrections: positive → negative	30 (33 %)
Manual corrections: negative → positive	59 (67 %)

Table 2: Metrics

Moreover, the submitted ratings are not always reliable. In an evaluation of the feedback provided, significantly more originally negative ratings had to be reclassified by the Quality Assurance team (QA team) as technically correct (59 ratings) than the other way around (30 ratings). "Corrected" in this context means that the QA team independently assesses the factual correctness of the bot's answer and reclassifies the user rating in cases of clear misjudgment. Overall, 89 of 1,286 submitted ratings (≈7%) were categorized differently by the QA team. Without additional controls, such misclassifications would bias the quality reporting (Kakur et al. 2024).

To address this gap, the QA team also reviews and rates conversations without user feedback retrospectively. This supplementary assessment is necessary to derive robust quality statements despite the large share of unrated answers and to detect "silent" incorrect responses that would otherwise remain unmeasured. The following table summarizes the resulting quality metrics.

The 78% rate refers to the share of bot messages in the observation period that were rated positive after QA, combining corrected user ratings and post-hoc ratings for previously unrated messages.

For this reason, continuous human quality control is essential. It is implemented as a PDCA cycle within the IT-ServiceDesk's ISO 9001 quality management system (Pieters et al. 2017): a four-person team invests a total of approximately eight hours per week to systematically review conversations retrospectively, validate user feedback, classify root causes (e.g., documentation gaps, retrieval mismatches, ambiguous terminology), and derive specific improvements. Reviews follow defined selection rules and cover the full set of conversations to detect both acute errors and gradual quality drift. Resulting measures are then implemented either in the knowledge base (e.g., content, structure, synonyms) or in system configuration (e.g., retrieval parameters and prompt rules) and subsequently re-evaluated. In this way, answer quality is not only assessed but also improved in a reproducible and auditable manner.

Figure 2 illustrates the development of positive ratings from February 2025 to January 2026, comparing manual user feedback (green) with overall feedback that combines user feedback and internal quality assurance (blue). Overall feedback is consistently higher and more stable, indicating that user feedback alone is often incomplete and biased as a quality indicator.

A notable outlier occurs in June 2025: manual feedback drops to 37.2% positive, while overall feedback remains largely stable at 75.9%. After targeted optimizations, manual feedback increases substantially and reaches 76.8% by January 2026; overall feedback also improves, reaching 87.2%. Overall, the trend suggests a sustained quality improvement after the initial calibration phase.

In a federated deployment, these quality management tools emerge as a decisive success factor for

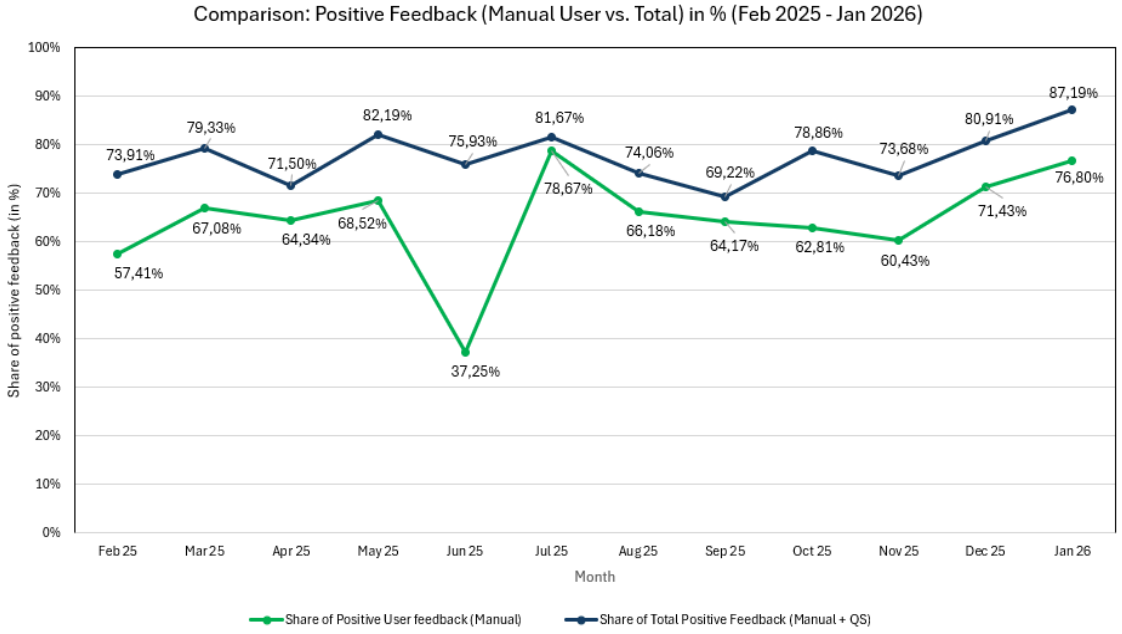

Figure 2: Feedback Ratings

Ritchy. Reliable answers depend not only on the technical RAG setup, but on an established,

repeatable quality process that detects negative trends early and corrects them systematically. For the consortium model to work, participating universities must adopt and actively use the following quality modules: review workflows, error classification, PDCA tracking, and the derivation and implementation of measures in documentation and configuration. Without shared QM tools, local knowledge bases, varying terminologies, and heterogeneous maintenance practices would produce widely divergent answer quality levels, ultimately jeopardizing user acceptance.

4 Benefits and Relief Through a Federated Deployment

The added value of a federated deployment becomes most apparent when universities work on comparable problems in parallel. This often involves similar effort and leads to redundant implementations. A federated service can reduce duplicated work while preserving local responsibility for content, processes, and quality standards. The benefits are multidimensional and address operational support teams, technical operations teams, and quality assurance roles.

At the operational level, a publicly accessible chatbot can answer recurring standard questions immediately and reduce waiting times, especially during off hours when traditional channels are not staffed (Sheth et al. 2020). Relief results not only from fewer standard contacts, but also from improved initial orientation. Users receive step-by-step guidance they can follow independently and learn which information is required for an effective support request. As a consequence, subsequent contacts are more structured and can be handled more efficiently. However, it is not yet possible to reliably verify the reduction in workload for the IT Center based on ticket numbers, as the data basis for this is currently still too small. The exonerative effect will therefore be examined in more detail in future evaluations.

On a technical level, the joint operation of basic functions reduces the workload.

If platform operations, tenant management, security and governance controls, monitoring, and release and approval processes do not need to be designed, implemented, and maintained separately at each university, the implementation and maintenance costs are significantly reduced.

A third, often underestimated aspect is quality assurance. A federated service should not only provide technical access to chatbot capabilities, but also reusable quality tools such as feedback analytics, sample-based reviews, and traceable change histories for prompts and configurations. These tools are independent of local documentation content and can follow standardized review logics. This makes them particularly transferable in a federated deployment.

At the same time, these benefits also define the limits of a federated approach. Content, terminology, approvals, and freshness remain institution specific. Federation therefore does not replace local documentation work, but increases its effectiveness by embedding it in a controlled quality loop. Based on these observations, we derive requirements for architecture and operations in the next section.

5 Requirements for a Federated Deployment

Based on operational experience with Ritchy and the objectives of a federated deployment, a set of scenario-independent requirements can be derived. Central to these requirements is multi-tenancy. Knowledge bases, search indexes, and log and usage data must be stored and processed under strict tenant separation. It is only then that shared operations can be aligned with data protection, governance, and accountability requirements.

Closely related to that is local content ownership. The chatbot should not be treated as a generic AI, but as an interface to an institution’s authoritative documentation. In a federated deployment,

platform functions can be operated jointly, while source repositories and their ongoing maintenance remain the responsibility of each institution.

Finally, quality assurance must be anchored as a core function. Review workflows, comparative testing, and traceable change histories for prompts and configurations should be standard capabilities rather than optional add-ons. Beyond governance and quality requirements, scalability and resilience are essential to handle peak loads without compromising service quality. These requirements are best supported by a clear architectural separation. The following section outlines this separation as a reference architecture.

6 Reference Architecture

To make federated deployment practical, we propose a reference architecture that separates responsibilities and defines clear interfaces between layers. The platform layer consolidates functions that can be operated efficiently as shared services, including a publicly accessible user interface, tenant management, security and governance controls, monitoring, and quality assurance tooling. In contrast, the university layer comprises institution-specific knowledge sources, their preprocessing, and the RAG components that connect these sources to the model, namely indexing and retrieval. These building blocks typically need to remain institution-specific due to differences in documentation, terminology, and approval processes.

To make responsibilities explicit, key artifacts are assigned to the respective layers. On the university layer, this includes knowledge repositories, derived indexes and embeddings, and institution-specific configurations and approvals. On the platform layer, shared operational and governance components are located, such as tenant and role management, policy and rule sets, centralized monitoring and logging capabilities, and quality assurance tools such as review workflows and change tracking. Usage data is collected per tenant and processed under strict separation. This ensures that data protection and accountability requirements are maintained even when parts of the system are operated jointly.

This separation should not be interpreted as a rigid deployment requirement, but as a conceptual model for responsibilities and data flows. Depending on the chosen scenario, components that are assigned to the university layer, such as model access or parts of the RAG pipeline, can also be operated on the platform layer. What matters is that tenant separation, transparency, and accountability always remain clear.

This reference architecture allows the following three deployment scenarios to be clearly classified. They primarily differ in where LLM access and the RAG pipeline are operated and to what extent universities can contribute or replace components. This results in different trade-offs in effort, autonomy, and governance requirements.

7 Deployment Scenarios

7.1 Scenario 1: All-in-One

In the all-in-one scenario, a central operator runs the full technical stack on the platform layer. Participating universities provide their content via upload or defined interfaces, while preprocessing, indexing, retrieval (RAG), and access to the language model are delivered centrally. This model is particularly suitable when a usable service must be introduced quickly and only limited local operational capacity is available.

The main advantages are a short time to launch and low local effort. Platform operations, monitoring, quality assurance tooling, and updates are managed centrally, allowing universities to focus on maintaining and approving their documentation. Implementing a uniform approach also simplifies quality assurance, as configuration changes can be rolled out in a controlled manner and evaluated through comparison and regression testing.

The trade-off is reduced local autonomy, for example with regard to component selection and configuration, coupled with increased governance requirements. When model access and the RAG pipeline are provided centrally, binding rules are needed for how changes are tested, deployed, and rolled back in case of issues. Tenant isolation must also be especially robust at both the technical and organizational level, since failures or data protection incidents would undermine trust across the entire federation.

7.2 Scenario 2: Fully Configurable (Own LLM and RAG Components)

The fully configurable scenario increases degrees of freedom within a standardized framework. The platform layer remains a shared core and provides the user interface, tenant management, security and governance controls, quality assurance tooling, and monitoring. However, universities can operate LLM access and/or RAG components themselves and connect them to the platform via standardized interfaces. This model is particularly aimed at institutions that already run their own infrastructure or require specific operating models due to compliance constraints.

Its main advantage is that it combines local autonomy with shared operational and governance logic. Universities can select and tune model providers, indexing technologies, or preprocessing strategies while still benefiting from a common quality assurance framework. This enables deployment in heterogeneous environments, as long as interface contracts remain stable.

The trade-off is higher integration and coordination effort. Root causes are harder to isolate when parts of the processing chain run outside the platform. Quality variance between tenants may also increase because retrieval and preprocessing strategies can diverge. For this reason, binding benchmarking becomes even more important. It makes quality comparable across universities, supports coordinated evolution, and enables improvements without centralizing technical decisions.

7.3 Scenario 3: Open source Self-Hosting (Scenario 2 as a Docker Distribution)

In the open source scenario, the fully configurable approach is delivered as a packaged stack that universities can operate locally. A central operator or consortium publishes a Docker-based distribution that reproducibly bundles the core capabilities of the platform layer. Universities can run this stack independently and connect or provide LLM and RAG components according to their own requirements.

This model is particularly attractive when maximum autonomy is desired or when dependencies on a central operator should be avoided. At the same time, the consortium can use versioning, reference configurations, and shared comparison and regression tests to ensure that improvements do not remain isolated local changes but are consolidated into a common product.

The trade-off is that self-hosting shifts operational responsibility back to the universities. Updates, security hardening, and monitoring must be implemented locally, which increases effort and raises the risk of diverging software versions and operating practices. To mitigate this, the consortium should agree on clear release cycles, update guidelines, and minimum requirements for logging, telemetry, and quality assurance. This preserves openness without sacrificing comparability or operational reliability.

8 Possible Cost Models

These three possible deployment scenarios imply different cost structures that primarily reflect the degree of centralization and local operational responsibility. Instead of specifying concrete prices, the following cost models outline conceptual approaches that are tailored to the three previous scenarios.

In the all-in-one scenario, a service-based pricing model is appropriate, which separates fixed platform costs from variable usage costs. An annual base fee covers onboarding, security, governance and quality assurance. In addition, a usage-based component is charged for variable cloud costs such as LLM inference (tokens/chats) as well as retrieval and index operation.

In the fully configurable scenario, central platform functions are operated jointly (user interface, client management, policies, monitoring, QM tools), while universities are responsible for LLM access and/or RAG components themselves. Accordingly, the cost model combines a reduced basic contribution for platform operation, governance and quality assurance with an additional contribution for standardization and quality assurance in heterogeneous operations, such as interface maintenance, reference configurations or comparative tests. Variable model and index costs are mainly borne locally, making the model particularly suitable for universities with their own infrastructure or specific compliance requirements, without sacrificing common quality mechanisms.

In the open source scenario, a packaged software stack is provided for local use, with operation, maintenance and security being entirely the responsibility of the universities. Accordingly, an open source plus subscription model is suitable, in which the software is freely available and the consortium finances central value-added services such as release and patch management, reference configurations and joint quality assurance mechanisms through annual subscriptions. Optional support packages can cover additional services such as onboarding or incident support, enabling maximum autonomy while ensuring comparability and operational reliability.

9 Conclusion and Outlook

Operational experience with Ritchy indicates that AI-supported IT support in a university context depends less on the choice of model and more on operational maturity and quality assurance. User feedback alone is too sparse and, in some cases, biased to serve as a reliable indicator for steering answer quality. Robust quality emerges from a continuous quality loop that combines documentation-grounded retrieval (RAG) as the factual basis, conservative generation settings to ensure traceability, and a repeatable review and PDCA process that systematically classifies root causes and feeds corrective actions back into documentation and configuration. In this approach, quality assurance is not an add-on but a core function.

For federated deployment, this translates into scenario-independent requirements, including strict tenant isolation, clear responsibilities, and local content ownership alongside jointly operated platform functions. The proposed reference architecture operationalizes this separation into a platform layer and a university layer, providing a robust basis for governance, data protection, and scalable operations. The three deployment scenarios, all-in-one, fully configurable, and open source, span a spectrum from rapid time to service to maximum autonomy. Regardless of the scenario, the more decentralized operations and component choices become, the more important standardized interfaces, binding benchmarking, release and change processes, and shared quality management tools are to maintain comparable quality across the consortium.

As a next step, we propose a staged, consortium-wide pilot. First, shared minimum standards should be defined, including a unified error taxonomy derived from the review process, a baseline set of metrics such as coverage outside service hours, repeat contact rate, handling times, benchmark scores, and distributions of error types, as well as a standardized test suite of representative inquiries. Building on this foundation, multiple universities can be piloted using identical metrics and review logic to evaluate scenarios and configuration variants in a comparable manner. Moreover, the quality modules should be further developed into a reusable package, including a review workflow with sampling, RAG evidence and logging, action tracking, and versioned release and change gates. This accelerates rollouts and enables early detection of quality drift between tenants through regression tests and trend analyses.

10 References

Bischof, C., Grzemski, S., & Hengstebeck, I. (2011). Introduction of a Service Desk at the Computing and Communication Center of RWTH Aachen University. A practical report. In A. Degkwitz et al. (Eds.), Process-oriented university [General aspects and practical examples] (pp. 181–198). Herchen Publishing House.

Grzemski, S., & Hengstebeck, I. (2017): Future challenges for quality-assured IT support through cooperative structures. In Shaping the Digital Future of Universities. Book of Proceedings Eunis 23rd Annual Congress, 6. https://doi.org/10.17879/21299722960

Grzemski, S., Decker, B., & Hengstebeck, I. (2025a). Implementation of an AI support chatbot based on Microsoft Azure OpenAI with special consideration of quality. In Proceedings of the EUNIS 2025 Annual Congress (pp. 237–246). Belfast, United Kingdom.

Grzemski, S., Hengstebeck, I., & Nohl, M. (2025b). Unermüdlich im Einsatz – IT-Support mit KI. DFN-Verein. https://www.dfn.de/unermuedlich-im-einsatz-it-support-mit-ki/.

Grzemski, S., Decker, B., Hengstebeck, I. & Jakobitz, R. (in press, 2026). Ritchy: Combining AI Automation with Human Quality Assurance in University IT Support.

Hengstebeck, I., & Grzemski, S. (2016). New ways of customer support at the IT- ServiceDesk of the IT Center of RWTH Aachen University. In H. C. Mayr et al. (Eds.), Lecture Notes in Informatics (LNI) (pp. 933–945). Society for Computer Science.

Jo, H., & Park, D.-H. (2023). AI in the Workplace: Examining the Effects of ChatGPT on Information Support and Knowledge Acquisition. International Journal of Human-Computer Interaction, 40(23), 8091–8106. https://doi.org/10.1080/10447318.2023.2278283

Kakur, R., Ritz, H., Hohmann, P. (2024). Analyse-Dashboard mit relevanten Kennzahlen für einen KI-basierten Chatbot im Hochschulbereich. In C. Müller et al. Anwendungen und Konzepte der Wirtschaftsinformatik (pp. 95-96).

Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems (Vol. 33, pp. 9459–9474).

Pieters, M., Hengstebeck, I., Grzemski, S. (2017). „Einführung eines zertifizierten Qualitätsmanagementsystems im IT-ServiceDesk des IT Centers der RWTH Aachen University“, In P. Müller, et al. Lecture Notes in Informatics (LNI) (pp. 77–87). Society for Computer Science. https://dl.gi.de/collections/08e1b8cd-ebb4-486f-9d84-557ee01bf50a

RWTH Aachen University. (n.d.). Facts and figures. Retrieved from https://www.rwth-aachen.de/cms/root/dierwth/profil/~enw/daten-fakten/

Sheth, A., Anantharam, P. and Thirunarayan (2020): “Challenges and Opportunities for AI in Customer Support”, IEEE Intelligent Systems, Vol.35, No. 5, 16-22

Skodowski, M. (2023) 20 Chatbot KPIs – Wie der Erfolg virtueller Assistenten gemessen wird. BOT friends GmbH (Eds.). https://botfriends.de/blog/chatbot-kpi

11 Author biographies

Ingo Hengstebeck M.A. studied Technical Communication. He received his Master’s degree from RWTH Aachen University in 2009. Until 2009, he worked as an employee at the IT Center. Since 2014 he has been the deputy head of the department “Service & Communication”. His work is focused on quality management, process management, and communication in the field of user support. (CRediT: Conceptualization, Investigation, Project Administration, Supervision, Writing – original draft)

Dipl. Inform. Bernd Decker is deputy head of the Department “Process Management and Digitalization in Learning & Teaching” at the IT Center of RWTH Aachen University since 2011. From 2006 to 2009, he worked at the IT Center as a Software Developer, and since 2009 he is leading the development group. His work focuses on IT solutions for processes in the fields of Learning Management Systems, E-Services, and Generative AI. (CRediT: Investigation, Writing – original draft)

Sarah Grzemski M.A. studied Economic Geography, Economics, and Geography, earning her Master's degree from RWTH Aachen University in 2002. Until 2007, she worked as a research assistant in the Department of Economic Geography of Services. Since then, she has been with the IT Center of RWTH Aachen University. In 2010, she became the division head of the IT-ServiceDesk, responsible for first-level IT support. Following organizational changes, her role expanded, and the department was renamed Service & Communication. She now oversees digital (web, social media) and analog (posters, flyers) external presentations, the RWTH printing service, user surveys on IT services, and the IT administration of the IT Center. (CRediT: Conceptualization, Investigation, Writing – original draft)

Sara Erdem M.Sc. studied Marketing Science and received her Master’s degree from Hochschule für Technik und Wirtschaft des Saarlandes in 2022. Since completing her degree, she has been working at the IT Center of RWTH Aachen University in the Service & Communication department. During this time, she worked as a Social Media Manager, but now her work focuses on reporting and analysis of social media channels and other support services. (CRediT: Writing – Review & Editing – original draft)