CoQuMa: Demonstration of the Competency Question Manager for Collaborative Ontology Requirements Engineering

Johannes Riedel[1,*], Jan Martin Keil[1] and Sirko Schindler[1]

[1]German Aerospace Center (DLR), Institute of Data Science, Mälzerstraße 3-5, 07745 Jena, Germany

Abstract

Competency Questions are at the core of most ontology engineering methodologies. Yet, structuring their elicitation has received rather little attention. Competency Questions are developed in a collaboration of Ontology Engineers and Domain Experts over the course of multiple iterations. Especially with an increasing number of participants, managing the process and its results requires more and more time and resources.

The Competency Question Manager (CoQuMa) is a platform to streamline this process for both, Domain Experts and Ontology Engineers, alike. In a web-based system, Domain Experts can document their own requirements and discuss, prioritise or modify existing Competency Questions. Ontology Engineers can ask for clarifications, provide suggestions for rephrasing, and finally arrange them into consolidated Competency Questions and topics. During the entire process, all contributions and actions are recorded within the system, allowing for tracing the results to initial contributions. CoQuMa allows Ontology Engineers to focus on their core task of structuring and formalising knowledge by considerably reducing the organisational overhead of gathering Competency Questions. For Domain Experts, CoQuMa provides an easy-to-use interface to contribute their knowledge, participate in the process, and review the results during the requirements elicitation and consolidation.

Keywords

Ontology Engineering, Competency Questions, Ontology Requirements Specification

1. Introduction

Even in the age of Large Language Models (LLMs), ontologies remain a vital source of formalised and validated information [1]. Over time, various methodologies have been designed to structure the ontology creation process [2, 3, 4]. A common step in most —if not all— of these approaches is the collection of requirements in the form of Competency Questions (CQs). An early definition for CQs describes them as "the requirements for an ontology and as such [...] the mechanism for characterising the ontology design search space" [5]. Later, their role was characterised as providing the "demarcation of the subject domain", "validation", and "alignment of a domain entity to a foundational ontology entity" as well as for the "elucidation or interrogation of ontological characteristics" [6].

Given their importance in the ontology design process, the collection, management, and refinement of CQs has received surprisingly little attention and tooling support. More often than not, CQs are collected via makeshift approaches like plain documents or tables. While this may be adequate for small- to medium-sized projects, it quickly grows out of hand when the number of CQs grows or various stakeholders are involved. In these scenarios, a single Ontology Engineer (OE) or a small group thereof has to coordinate a comparatively larger group of Domain Experts (DEs), gather their individual requirements, and finally aggregate all input into a finalised set of CQs. The authors of this paper themselves experienced this on multiple occasions including a project with 14 experts from 7 different institutes that covered a wide range of domain backgrounds and technical experience [7, 8]. Throughout the process, we encountered various challenges like establishing a uniform workflow suitable for all DEs, developing shared term definitions across different domains for a common vocabulary, grouping CQs

The 25th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2026) - Satellite Events (Workshops, Tutorials, Demos, Posters)

*Corresponding author.

johannes.riedel@dlr.de (J. Riedel); jan_martin.keil@dlr.de (J. M. Keil); sirko.schindler@dlr.de (S. Schindler)

0000-0003-2551-3358 (J. Riedel); 0000-0002-7733-0193 (J. M. Keil); 0000-0002-0964-4457 (S. Schindler)

into different topics to elicit feedback from only the relevant subset of DEs, documenting provenance to keep track of attributions, and many more.

This recurring experience and the lack of suitable tools motivated the design and development of the Competency Question Manager (CoQuMa). It allows joint teams of DEs and OEs to collaboratively work on the creating and refining CQs and to form a consensus about the requirements to a future ontology. In comparison, shared documents require a lot of manual work by the OE to assert agreed structures and to track the provenance of CQs. CoQuMa reduces the administrative overhead and shifts the main focus back to content-related tasks. For DEs, it offers easy-to-use interfaces to contribute new CQs, validate and discuss the wording of existing ones, and make sure that individual inputs are considered in the further process. On the other hand, various tools are provided for OEs to guide and organise the process. Besides access to the features for DEs, this includes, e.g., tools to cluster CQs into topics for organisational purposes, label individual CQs as "out-of-scope", or combine individual input CQs into so-called "consolidated Competency Questions (cCQs)". The latter allows to harmonise across various stakeholder inputs and form the final set of CQs for the design and implementation of an ontology. During the entire process, all contributions are tied to their creators allowing for an easy attribution within the final results. While some of these features are well known from issue trackers, a dedicated tool provides the advantage of guidance and constrains specific to CQs.

In this paper we summarise the capabilities of CoQuMa and provide a glimpse into its design philosophy and interfaces. After a short review of existing activities in the field in Section 2, we provide an overview of the architecture and workflow of CoQuMa in Section 3. Finally, we describe the planned demonstration in Section 4 and outline further developments in Section 5.

2. Related Work

Watkiss-Leek et al. introduce the IDEA2 framework [9], which provides a workflow to generate and improve CQs out of documents or user stories using an LLM. After the initial CQs generation using the LLM, the CQs are continuously improved by providing the feedback of multiple DEs to the LLM until agreement and consensus of the DEs or another break condition was reached. While the framework ensures the correctness of the generated CQs that way, it does not provide mechanisms to ensure their completeness. Beside that, Watkiss-Leek et al. mention the risk of "reviewer fatigue" when the task of DEs is reduced to review generated CQs. Even more critical, we regard the lack of involvement of OEs in the requirement engineering process, as this might later result in a lack of domain knowledge by OEs during the ontology formalisation.

Alharbi et al. conducted a systematic review of methodologies for CQ construction [10]. They identified several approaches for the manual, semi-automatic, and automatic definition of CQs that support in the wording of CQs. However, none of the considered works addressed the issue of managing CQs over the course of an ontology development project.

3. System Overview

CoQuMa is a web application that supports DE and OE to collaboratively specify the requirements on an ontology. Starting from CQs provided by multiple DEs, the goal is to obtain a unified list of cCQs. With CoQuMa, OEs should be enabled (a) to identify overlaps in the requirements of different stakeholders and thereby avoid redundancy in the CQs, (b) to mitigate gaps in the terminology between DEs of different domains, (c) to group CQs by associating them to topics, and (d) to track the provenance of cCQs and the coverage of initial CQs by interlinking CQs. At the same time, rephrasing and discussing CQs remains the responsibility of the OEs, which enables them to improve their own understanding of the domain, which is important for the later ontology formalisation. CoQuMa is publicly available on GitLab.com[1] and Zenodo [11] under the permissive Apache 2.0 license[2].

[1]https://gitlab.com/dlr-dw/CoQuMa [2]https://opensource.org/licenses/Apache-2.0

For the implementation of CoQuMa, we followed a three-layer architecture consisting of a frontend, a backend, and a database. The frontend is a web application implemented in TypeScript based on Svelte[3] and Astro,[4] as well as Tailwind CSS[5] for styling the user interface. It communicates with the backend using a REST API. The backend is also implemented in Typescript and uses Express[6] as a base library and is extended by Prisma,[7] which functions as a object-relational mapping framework. Data are persisted by the backend in a PostgreSQL[8] database. All of these components are containerised using Docker.[9] This architecture ensures modularity and enables automated access to the system through the REST API while facilitating possible replacement of certain technologies or frameworks.

The **intended workflow** is visualised in Figure 1. An ontology requirement analysis using CoQuMa starts with the *registration* of all involved OEs and DEs and the *creation of teams* to group users that represent a specific stakeholder. Inside a team, each user has a dedicated role, i.e. DE or OE, lead, or guest. In case user accounts or teams are already available from previous projects, both steps can be skipped. Next, an OE *creates a project* for the ontology requirements analysis in CoQuMa and *adds all relevant teams* to the project. After this initial project setup, all DEs can contribute requirements to the ontology by *creating initial CQs*. These initial CQs might then be refined by the DEs in collaboration with the OEs to resolve ambiguity. The refinement should involve detailed communication between DEs and OEs, also enabling the OEs to gain the knowledge about the domain required for the later formalisation of the ontology. After finalising the initial CQ, the OEs *create consolidated Competency Questions* (cCQs), aligning terminology, eliminating redundancy, and adjusting their size by merging or splitting CQs. A cCQ is a generalised CQ that links one or multiple related CQs expressing similar requirements into one or more abstract formulations. This approach reduces redundancy and improves the maintainability of the CQs. Consequently, a cCQ provides a structured specification of the functional requirements of an ontology, without compromising the information given by the linked CQs. The cCQs drafted by OEs might then be refined by the DEs in collaboration with the OEs to ensure their terminological correctness and the complete coverage of the initial CQs. The interlinking between initial CQs and cCQs simplifies this verification of the completeness. A possible outcome of this intended workflow is illustrated in the Figure 2.

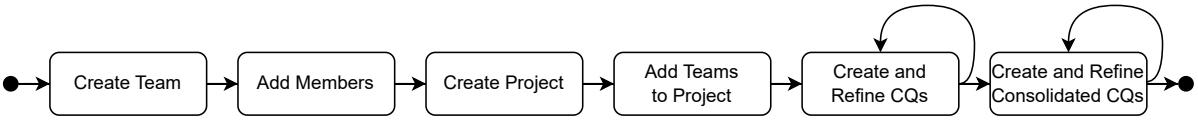

Figure 1: Illustration of the intended workflow for ontology requirements specification using CoQuMa.

The **permission model** restricts activities that users of different roles can perform to avoid overwhelming users with too many options. Currently, there are five roles in CoQuMa: DE, OE, Creator, Lead, and Guest. A screenshot of the user interface for the configuration of a team with users of different permissions is shown in Figure 3. A DE can create an initial CQ and can view or comment on any CQs. An OE can view and comment on any CQ and can create cCQs. DEs or OEs can only delete or update (i.e. change title, description, etc.) CQs they created themselves. The Creator of a project or team and additionally appointed Lead users have full access to all entities within their respective project or team, including viewing, creation, commenting, updating, and deletion. The Creator and the Lead roles are equivalent with respect to permissions. Their differentiation is exclusively representational and serves to identify to other participants who originally created the team or project. Guest users have permission to view all entities within a team or project but can not perform any further action.

To **facilitate collaboration** between the users, especially users of different teams and representing different stakeholders, CoQuMa provides several features for the communication: In a project, multiple teams can be involved representing multiple stakeholders to include all relevant viewpoints of a use case. Comments on CQs can be used to suggest and discuss improvements. A voting system (up- and

[3]https://svelte.dev/ [4]https://astro.build/ [5]https://tailwindcss.com/ [6]https://expressjs.com/ [7]https://www.prisma.io/ [8]https://www.postgresql.org/ [9]https://www.docker.com/

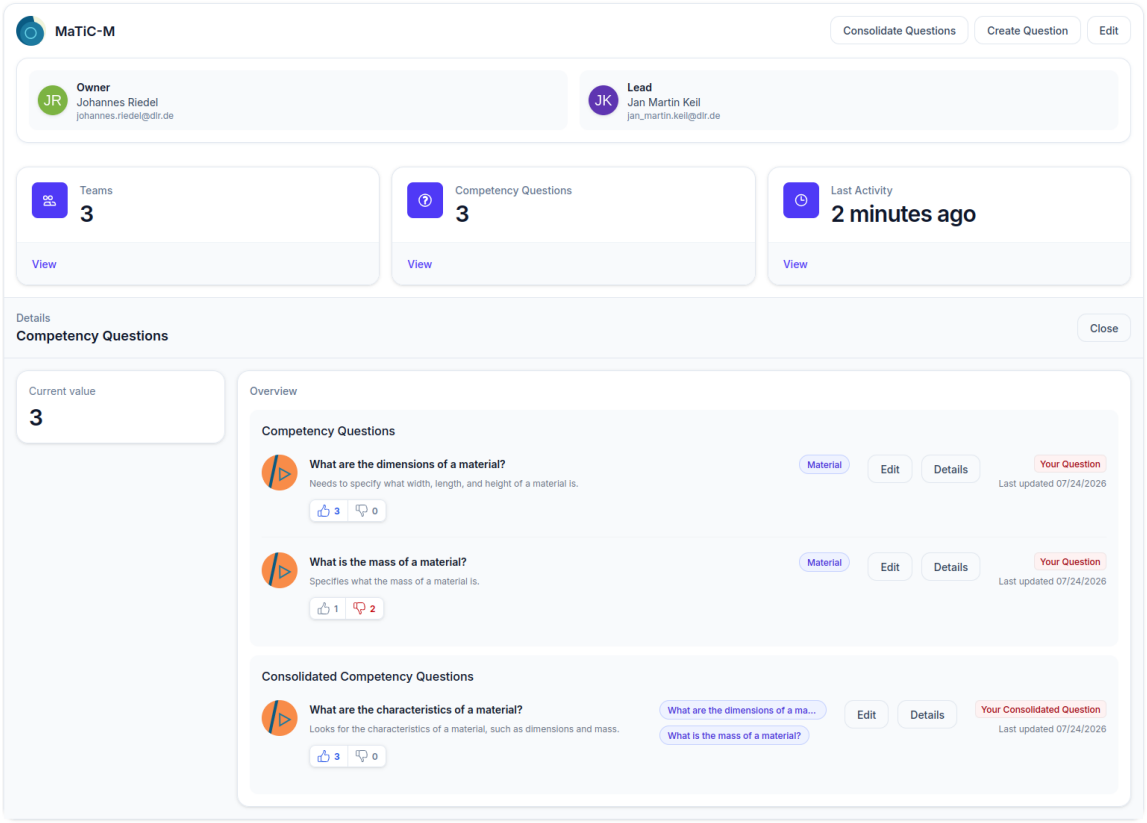

Figure 2: Screenshot of the user interface showing a project containing two CQs with associated voting results, as well as a cCQ that links to two initial CQs.

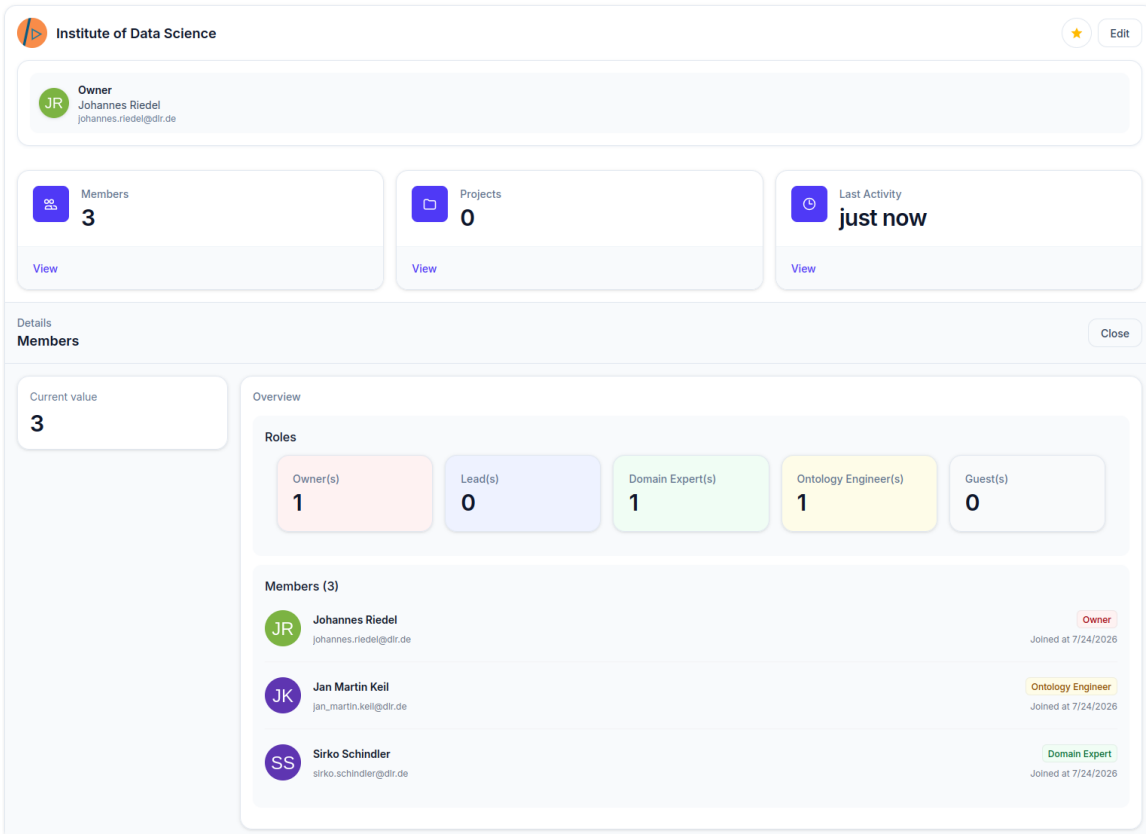

Figure 3: Screenshot of the user interface showing the permissions of different participants in a minimal team configuration of one Domain Expert (DE), one Ontology Engineer (OE), and the creator of the team.

down-votes) can be used to indicate the relevancy of particular CQs. To further structure CQs, they can be clustered into topics allowing to easily manage multiple concurrent threads during development.

4. Demonstration

During the demonstration, users have the opportunity to experience CoQuMa first hand. We will provide prepared sample data as well as the opportunity to create new projects. Either way, visitors can explore CoQuMa in both roles – as Domain Experts or Ontology Engineers – and try out the respective workflows in a live demo system. In particular, this includes the fundamentals of user and team management as well as all stages of a Competency Question's life-cycle from creation and discussion up to the final result in form of consolidated Competency Questions.

During the demonstration, we actively encourage feedback on both the interface as well as the workflows underlying CoQuMa. This will help us to improve CoQuMa and closely align with users' needs. A summary video of CoQuMa's current state is provided online: doi:10.5281/zenodo.21534515.

5. Conclusion and Future Work

In this paper, we introduced CoQuMa – a tool to collaboratively create, manage, and refine Competency Questions (CQs). It allows Domain Experts and Ontology Engineers to jointly gather requirements to an ontology. Users are provided with role-specific interfaces regulating access to certain capabilities and providing for a clean user experience. Users of different backgrounds and stakeholder groups can be organised into individual teams and assigned to different projects. Here, they can add new CQs, discuss or evaluate existing ones, and merge them into consolidated Competency Questions. The result is a curated set of CQs that can be used as the basis for further ontology development.

We are committed to continue the development of CoQuMa and enhance its capabilities. This includes, e.g., a structured export of the provenance making attribution to individual users machine-readable. Other plans include a closer integration into the subsequent ontology development allowing, e.g., the verification of the resulting ontology using previously created CQs, as proposed in the test-driven development for ontology approach [12]. Further, the incorporation of existing methods for the semi-automated definition of CQs [10] could further support DEs.

Acknowledgments

We thank Chiara Tunc, Daniel Motz, Dominik Buschold, and Malte Weber for implementing a preliminary prototype software during a student project at the University of Jena supervised by Susanne M. Hoffmann and Jan Martin Keil. Likewise, we thank Amirhossein Mostafasaraji, Lara Schwarze, Moritz Arnhold, Moritz Ehrhardt and Moritz Martin for implementing a preliminary prototype software during a second student project at the University of Jena supervised by Eleonora Petzold and Johannes Riedel.

Author Contributions

Johannes Riedel: Conceptualisation; Software; Writing - Original Draft, Review, Editing; Validation. **Jan Martin Keil:** Conceptualisation; Methodology; Validation; Writing - Original Draft, Review, Editing; Validation. **Sirko Schindler:** Conceptualisation; Writing - Original Draft, Review, Editing; Validation.

Declaration on Generative AI

During the preparation of this work, the authors have not employed any Generative AI tools.

References

[1] I. Yadav, S. Schindler, D. Peters, R. Klinger, External knowledge integration in large language models: A survey on methods, challenges, and future directions, Semantic Web: – Interoperability, Usability, Applicability 17 (2026). doi:10.1177/22104968261453132.

[2] N. F. Noy, D. L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, Technical Report KSL-01-05/SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics, 2001. URL: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf.

[3] M. C. Suárez-Figueroa, A. Gómez-Pérez, M. Fernández-López, The NeOn Methodology for Ontology Engineering, Springer Berlin Heidelberg, 2011, pp. 9–34. doi:10.1007/978-3-642-24794-1_2.

[4] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, R. García-Castro, LOT: An industrial oriented ontology engineering framework, Engineering Applications of Artificial Intelligence 111 (2022) 104755. doi:10.1016/j.engappai.2022.104755.

[5] M. Uschold, M. Gruninger, Ontologies: principles, methods and applications, The Knowledge Engineering Review 11 (1996) 93–136. doi:10.1017/S0269888900007797.

[6] C. M. Keet, Z. C. Khan, Characterising competency questions for ontologies, in: Proceedings of the Joint Ontology Workshops (JOWO) - Episode XI: The Sicilian Summer under the Etna, co-located with the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), Catania, Italy, September 8-12, 2025, volume 4176 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2025. URL: https://ceur-ws.org/Vol-4176/caos-9.pdf.

[7] J. M. Keil, T. A. Köhler, S. Schindler, Communication in Design-for-Circularity: Requirements to a Knowledge Graph, in: Proceedings of the The 2nd International Workshop on Knowledge Graphs for Sustainability (KG4S 2024) colocated with the 21st Extended Semantic Web Conference (ESWC 2024), volume 3753 of *CEUR Workshop Proceedings*, ceur-ws.org, 2024, pp. 86–92. URL: https://ceur-ws.org/Vol-3753/short2.pdf.

[8] J. M. Keil, M. Abdallah, E. Beeh, K. Bugelnig, J. Haubrich, S. S. Kumtamukkula, T. Lorenz, M. Löbbecke, N. C. Neumann, K. Nottensteiner, D. Peters, R. Reiser, I. V. Rodriguez Brena, R. Sturm, B. Thiele, S. Torstrick-von der Lieth, MaTiC-M Knowledge Graph Competency Questions, 2024. doi:10.5281/zenodo.10730784.

[9] E. Watkiss-Leek, R. Alharbi, H. Rostron, A. Ng, E. Johnson, A. Mitchell, T. R. Payne, V. Tamma, J. de Berardinis, IDEA2: Expert-in-the-loop competency question elicitation for collaborative ontology engineering, in: Proceedings of the Workshop on LLM-driven Knowledge Graph and Ontology Engineering (LLMs4KGOE 2026) co-located with the 23rd European Semantic Web Conference (ESWC 2026), 2026, pp. 88–102. URL: https://ceur-ws.org/Vol-4246/llms4kgoe-8.pdf.

[10] R. Alharbi, V. Tamma, F. Grasso, T. R. Payne, A Review and Comparison of Competency Question Engineering Approaches, Springer Nature Switzerland, 2024, pp. 271–290. doi:10.1007/978-3-031-77792-9_17.

[11] J. Riedel, CoQuMa Software v.1.0.0, 2026. doi:10.5281/zenodo.22280528.

[12] C. M. Keet, A. Ławrynowicz, Test-Driven Development of Ontologies, Springer International Publishing, 2016, pp. 642–657. doi:10.1007/978-3-319-34129-3_39.