

# Post-Hoc Insights: Natural-Language Explanations for AI-Enhanced/-Integrated Software Systems

Dennis Schiese<sup>1</sup>, Aleksandr Perevalov<sup>1</sup> and Andreas Both<sup>1,2</sup>

<sup>1</sup>Leipzig University of Applied Sciences, Leipzig, Germany

<sup>2</sup>DATEV eG, Nuremberg, Germany

## Abstract

In this paper, we aim to address the problem of explainability as a broad research area for the specific case of component-based systems. Our goal is to create a greater and deeper understanding of such a system and its execution process, which is determined by the observable components' data flows. Since we approach this problem from the data perspective and focus on component-based Question Answering systems, we consider two data types: SPARQL & RDF triples. We present a demonstrator for generating corresponding explanations by using generative AI systems and for comparison based on templates. Our approach is provided as an open-source web application that is freely accessible to all users.

## Keywords

Explainability, Component-based Systems, Natural-Language Generation, Large Language Models

## 1. Introduction

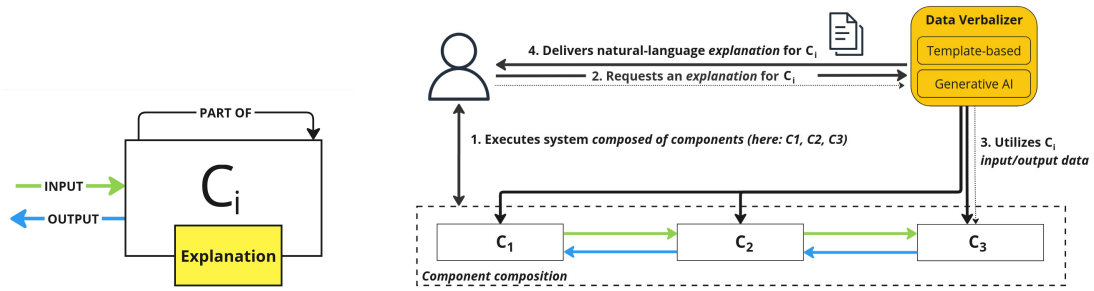
Explanations are of crucial importance in enhancing not only but primarily both, the trustworthiness of a wide range of systems and the traceability of responses. With the ever-increasing complexity of software and the recent and fast progress in artificial intelligence (AI), systems become increasingly opaque to both users and developers [1]. Since these “black-box” systems are problematic for both sides, explainability plays an increasingly important role, especially in the area of AI. There, the term *explainability* primarily describes methods and actions to understand the predictions and thus the decision-making of a certain model, whether it is opaque or transparent. While transparent models are comparably easy to explain, it is difficult to do so for opaque models [2]. Therefore, for such models, explanations are generated post-hoc and with different approaches, e.g., LIME [3] or SHAP [4]. Here, we present an approach for post-hoc explainability within component-based AI-enhanced software systems, i.e., the explainability of the behavior of components in general (not limited to AI models) that uses the system's data that was processed and/or is reflecting the intermediate processing steps. Accordingly, (post-hoc) explanations are created for each component, regardless of their transparency. Our approach is driven by the assumption that probably more general, but also more appropriate explanations for all kinds of components can be generated. We argue that the value of such explanations and the insight into a system can be further enhanced if the system under consideration consists of well-defined components (in particular, having a clear purpose), with the explanations thus being task-oriented. Hence, we want to explore this hypothesis by applying our approach to a real-world component-based system and propose natural-language explanations.

---

SEMANTiCS 2024



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



- (a) General component model with input and output representing the data streams used to create the component's explanations.
- (b) Process of executing the system and requesting an explanation for a specific component (dotted arrow). The *Data Verbalizer* consists of two different approaches to generate explanations: template-based and using generative AI.

**Figure 1:** General component model and process big picture.

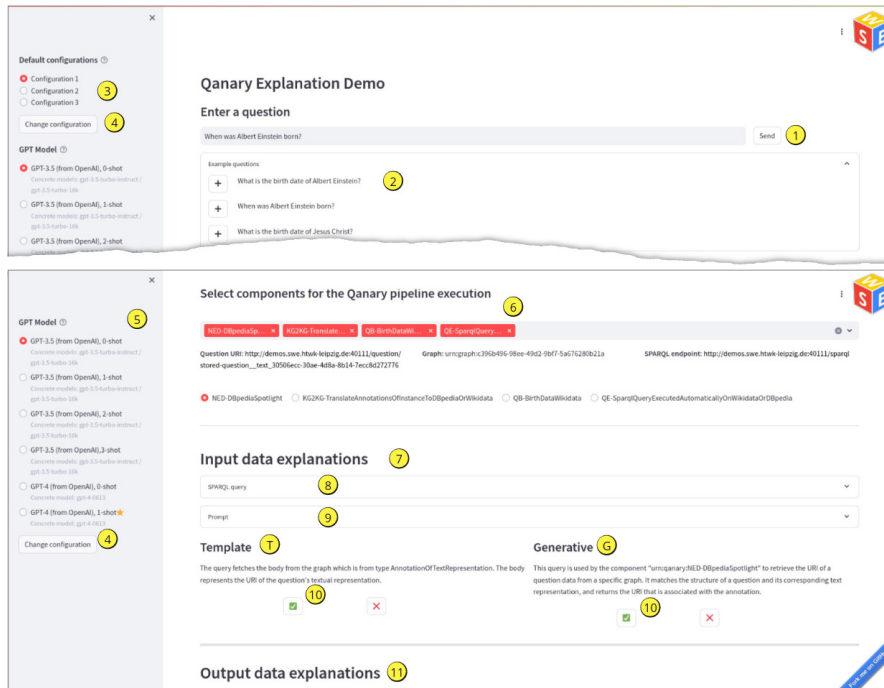
## 2. Related work

Explainability plays a crucial role in gaining a better understanding of a system's behavior. Although the topic is not entirely new, it has become important again with the increasing use of AI in the past few years. Recent research on explainable AI (XAI) [2] has mainly focused on explaining the models and therefore the process. Especially the explanation of opaque models and therefore "black-boxes" remains a great field of future research. Currently, multiple categorizations and corresponding approaches exist for this model type, as outlined in [4, 5, 6]. However, in many cases, these approaches are unsuitable [7]. Therefore, we shift our focus to component-based systems, aiming to generate natural-language explanations post-hoc using real data. To achieve this, we will leverage generative AI, which shows potential in producing explanations due to its proven text generation capabilities [8, 9]. To the best of our knowledge, similar approaches that utilize a system's and more specifically a component's data to create explanations using generative AI, were not known at the time this paper was published.

## 3. Explaining components

Component-based systems utilize the principle of separation of concerns, which we believe enhances explainability. Thus, when assessing explainability in these systems, the focus should be on the individual components. To guide this approach, a brief concept is introduced below.

**Conceptual basis** As the basis for the concept, we developed a generalistic component model which is shown in Figure 1a. It reflects the basic composition of systems where a component  $C$  might call other components which thereby become part of the execution of  $C$ . With this, we describe a minimal, explainable model that can be used to create any (explainable) component-based system. This model can be used to reflect systems composed of components and their data flows. Please note, that this concept is based on the assumption of an orchestrated system and probably only applies to such systems. For the model shown, this means that the data



**Figure 2:** Screenshot of the demonstrator where the input data (represented as SPARQL query) and the output data (represented as RDF triples) are verbalized with different configurations.

stream between a child and its orchestrator is unique. To represent this concept, we derived an ontology<sup>1</sup> which serves as a basis for further extensions. To apply the basis ontology to a use case, the following system information is required: 1 Input and output data streams, 2 Datasets following from data streams, 3 Type of (here: natural-language) explanation, 4 Component integration, and 5 Approach to generate explanations with provided data.

In conclusion, the presented model establishes a basis for explainable component-based systems, characterized by unique data streams between components and their parent. Additionally, an extendable ontology representing this model has been developed.

**Demonstrator** Finally, as a visualization of this concept, applied to a concrete system, the demonstrator shown in Figure 2 has been developed. This web service can perform a Qanary QA process and provides English explanations for each component. The creation of these post-hoc explanations thus follows the workflow shown in Figure 1b. This illustration shows the two existing approaches for the creation of explanations: Via templates and Generative AI. Both are realized and visible in the demonstrator that we used to conduct a preliminary study with about 3000 experiments, where we found that the generatively generated explanations are almost as good as the template-based ones. In particular, in the case of SPARQL query verbalization, the quality of the generatively generated explanations was superior to the quality of the template-based explanations. In view of these results and the resulting findings, we would be grateful for a lively evaluation of the explanations provided by the following demonstrator.

<sup>1</sup>[https://github.com/WSE-research/qanary-explanation-service/blob/main/plain\\_ontology.owl](https://github.com/WSE-research/qanary-explanation-service/blob/main/plain_ontology.owl)

With the demonstrator, it is possible to start a QA process ① by passing a question or ② by selecting a pre-defined one (s.t., an actual execution run can be started). The pre-defined questions are dependent on the selected configuration ③. Here, different component combinations are configured, to easily get started. However, it is also possible to change the configuration ④ and select your very own component combination. When doing so, the order is relevant to the QA process ⑥. When starting the QA process, the explanations are generated afterward. While the template-based explanations will not vary, the generative ones do, due to the characteristics of LLMs. As the quality of these variations may be the most interesting part, the used examples and OpenAI's GPT models<sup>2</sup> can be changed ⑤. When the process is finished, the explanations for all components are available and displayed (for input data: ⑦, for output data: ⑪). For each explanation, the dataset ⑧ and prompt ⑨ can be inspected. The explanations are split up into *Template* and *Generative*. Below every explanation, the quality can be rated as good or poor ⑩. Summarized, our demonstrator supports initiating a QA process via custom or pre-defined questions, influenced by configurable component combinations and the (post-hoc) generated explanations, either template-based or generative, can be reviewed and rated for quality. The application was implemented using the Python library Streamlit. It is available as an online demo<sup>3</sup> as well as published on GitHub<sup>4</sup> and available as a Docker image on Dockerhub<sup>5</sup>. The source code of both applications is released under the MIT license.

**Qanary use-case** Our demonstrator used the Qanary framework<sup>6</sup> [10, 11, 12] as foundation for a component-based system that is manifesting a Question Answering application. Following the Qanary methodology, each component fetches the required data (input) by itself from the Qanary triplestore (centralized process memory) and stores the created data (component output) there after finishing the processing. Hence, this setting is perfectly suited for our use case, as the input and output data of each component is already tracked in the process memory of the Qanary-driven system. Thereafter, we utilized this data to generate the explanations in the demonstrator. Concerning the previously introduced concept we follow, we can declare the access to the triplestore (read/write) as input and output data streams (cf. ①), SPARQL queries (fetching input data) and RDF triples (stored output data) as data (cf. ②), natural language as explanation type (cf. ③), and API-based server/client registration as component integration (cf. ④). Finally, in our demonstrator, two methods for automatic explanation generation per component are integrated: using manually defined templates (cf. ⑤) and using several LLMs (cf. ④ and ⑥). The extended ontology is accessible on GitHub<sup>7</sup>.

## 4. Conclusion

The lack of explainability in many (not only AI-driven) systems is clearly evident. In this paper, we have thus presented an application that addresses this problem for component-based systems,

<sup>2</sup>gpt-3.5-turbo-instruct, gpt-3.5-turbo-16k, gpt-4-0613 | Chat and Chat-Completions API with vanilla settings

<sup>3</sup><http://demos.swe.htwk-leipzig.de:40119/>

<sup>4</sup><https://github.com/WSE-research/qanary-explainability-frontend> and [/qanary-explanation-service](https://github.com/WSE-research/qanary-explanation-service)

<sup>5</sup><https://hub.docker.com/r/wseresearch/qanary-explainability-frontend>

<sup>6</sup><https://github.com/WDAqua/Qanary>

<sup>7</sup>[https://github.com/WSE-research/qanary-explanation-service/blob/main/ontology\\_qanary.owl](https://github.com/WSE-research/qanary-explanation-service/blob/main/ontology_qanary.owl)

as we believe this increases an explanation's potential. The generated explanations were based on the input and output data of each process-related component and were generated automatically using (1) templates and (2) generative AI (using several LLMs). The latter explanations vary more or less depending on the LLM settings used, such as the examples given, the GPT models, or others<sup>8</sup>. Finally, this methodology represents a valuable and general approach to explaining a system by explaining its components. Furthermore, with the increasing performance of Large Language Models (LLMs), the results will become probably increasingly accurate and better.

**Acknowledgments:** This work was partially supported by the German Federal Ministry of Economics and Technology (BMW) under the number 16DTM107B (*ASAGuR*).

## References

- [1] L. Chazette, W. Brunotte, T. Speith, Exploring explainability: A definition, a model, and a knowledge catalogue, in: IEEE 29th Int. Requirements Eng. Conf., 2021, pp. 197–208.
- [2] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, P. M. Atkinson, Explainable artificial intelligence: an analytical review, *WIREs Data Mining and Knowledge Discovery* 11 (2021).
- [3] M. T. Ribeiro, S. Singh, C. Guestrin, "why should I trust you?": Explaining the predictions of any classifier, *CoRR abs/1602.04938* (2016). [arXiv:1602.04938](https://arxiv.org/abs/1602.04938).
- [4] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, *Advances in neural information processing systems* 30 (2017).
- [5] T. Verma, C. Lingenfelder, D. Klakow, Generating natural language explanations for black-box predictions, in: 2022 Fourth International Conference on Transdisciplinary AI (TransAI), IEEE, 2022, pp. 40–46. doi:10.1109/TransAI54797.2022.00013.
- [6] R. Guidotti, A. Monreale, F. Giannotti, D. Pedreschi, S. Ruggieri, F. Turini, Factual and counterfactual explanations for black box decision making, *IEEE Intelligent Systems* 34 (2019) 14–23. doi:10.1109/MIS.2019.2957223.
- [7] H. Lakkaraju, D. Slack, Y. Chen, C. Tan, S. Singh, Rethinking explainability as a dialogue: A practitioner's perspective, *NeurIPS Workshop on Human Centered AI* (2022).
- [8] A. Bhattacharjee, R. Moraffah, J. Garland, H. Liu, Towards LLM-guided causal explainability for black-box text classifiers, in: *AAAI 2024 Workshop on Responsible Lang. Models*, 2024.
- [9] R. Ajwani, S. R. Javaji, F. Rudzicz, Z. Zhu, LLM-generated black-box explanations can be adversarially helpful, 2024. [arXiv:2405.06800](https://arxiv.org/abs/2405.06800) [cs].
- [10] A. Both, D. Diefenbach, K. Singh, S. Shekarpour, D. Cherix, C. Lange, Qanary – a methodology for vocabulary-driven open question answering systems, in: *The Semantic Web. Latest Advances and New Domains*, Springer, 2016, pp. 625–641.
- [11] K. Singh, A. Both, D. Diefenbach, S. Shekarpour, D. Cherix, C. Lange, Qanary – the fast track to creating a question answering system with linked data technology, in: *The Semantic Web*, Springer, 2016, pp. 183–188. doi:10.1007/978-3-319-47602-5\_36.
- [12] D. Diefenbach, K. Singh, A. Both, D. Cherix, C. Lange, S. Auer, The Qanary ecosystem: Getting new insights by composing question answering pipelines, in: *Web Engineering*, Springer, 2017, pp. 171–189. doi:10.1007/978-3-319-60131-1\_10.

---

<sup>8</sup>Depending on the used LLM, there may be settings that impact the randomness or creativity of responses.