

An AI Chatbot for Explaining Deep Reinforcement Learning Decisions of Service-oriented Systems

Andreas Metzger^[0000–0002–4808–8297], Jone Bartel, and Jan Laufer

paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen, Essen, Germany
`firstname.lastname@paluno.uni-due.de`

Abstract. Deep Reinforcement Learning (Deep RL) is increasingly used to cope with the open-world assumption in service-oriented systems. Deep RL was successfully applied to problems such as dynamic service composition, job scheduling, and offloading, as well as service adaptation. While Deep RL offers many benefits, understanding the decision-making of Deep RL is challenging because its learned decision-making policy essentially appears as a black box. Yet, understanding the decision-making of Deep RL is key to help service developers perform debugging, support service providers to comply with relevant legal frameworks, and facilitate service users to build trust. We introduce Chat4XAI to facilitate the understanding of the decision-making of Deep RL by providing natural-language explanations. Compared with visual explanations, the reported benefits of natural-language explanations include better understandability for non-technical users, increased user acceptance and trust, as well as more efficient explanations. Chat4XAI leverages modern AI chatbot technology and dedicated prompt engineering. Compared to earlier work on natural-language explanations using classical software-based dialogue systems, using an AI chatbot eliminates the need for eliciting and defining potential questions and answers up-front. We prototypically realize Chat4XAI using OpenAI’s ChatGPT API and evaluate the fidelity and stability of its explanations using an adaptive service exemplar.

Keywords: chatbot · explainable AI · reinforcement learning · service engineering · service adaptation

1 Introduction

Reinforcement Learning (RL) is increasingly used to cope with the open-world assumption of service-oriented systems, as it helps to address the design-time uncertainty during service and systems engineering [1, 26]. In general, RL aims to learn an optimal decision-making policy for executing a suitable action in a given environment [41]. In a service-oriented system, RL can learn suitable actions via the service-oriented system’s interactions with its initially unknown environment and thereby can make use of information only available at runtime [34]. RL

helped successfully address various problems in service-oriented systems, including dynamic service composition [37], task/job scheduling [12, 14, 43], resource management [13], and service adaptation [32].

Need for Explainability. Recent research on using RL for realizing service-oriented systems leverages Deep RL algorithms, which represent their decision-making policy as a deep neural network (examples include [9, 12, 20, 28]). Deep RL inputs are not limited to elements of finite or discrete sets, and the used neural networks can generalize well over unseen inputs. Deep RL can also natively capture concept drift in a service-oriented system’s environment without the need to explicitly introduce mechanisms to monitor such concept drift [34].

However, one key shortcoming of Deep RL is that the learned decision-making policy is not represented explicitly, but is hidden in the parametrization of the neural network. For service developers and users, the decision-making of Deep RL thus essentially appears as a black box [36]. This means that we require techniques to explain and interpret the internal workings of such black-box systems and how their decisions are made [3, 10, 27, 29].

Explaining the decision-making of Deep RL can help service developers debug the reward function by understanding why Deep RL took certain decisions. The successful application of Deep RL depends on how well the learning problem, and in particular the reward function, is defined [5]. Further, explainability can facilitate regulatory compliance [29]. For example, in the EU, service providers must ensure that their services comply with the relevant legal frameworks, such as the General Data Protection Regulation and the forthcoming AI Act. Third, explanations facilitate service users to build trust. They can understand how the service arrived at its results and thus can accept its results or not [29].

Problem Statement. Two major types of explanation formats can be distinguished [2, 22, 29]: (i) visual explanations, including graphical user interfaces, charts, data visualization, or heatmaps, and (ii) verbal explanations, which, for instance, may take the form of a natural-language dialogue between the explainer and explainee. The chosen presentation method has a direct effect on user comprehension and, therefore, on the success of the explanation [22]. Compared with visual explanations, the benefits of verbal explanations reported in the literature [2, 23] include (1) better understandability for people with diverse backgrounds as well as non-technical users, (2) increased user acceptance and trust, and (3) more efficient explanations.

While the literature on using Deep RL for service-oriented systems provides extensive and systematic evaluations of the performance of Deep RL [9, 12, 20, 28], the problem of how to explain the decision-making of Deep RL using natural language was not yet addressed. In the broader area of explainable AI (XAI), approaches for providing natural-language explanations for machine learning exist [18, 19, 22, 33]. However, these XAI approaches focus on supervised learning and not on RL. Also, they are all built using classical software-based dialogue systems [31], which require the additional engineering step of eliciting and defining potential questions and answers up-front.

Contributions. We introduce **Chat4XAI**, which leverages the capabilities of a modern AI chatbot powered by a large language model to provide natural-language explanations about the decision-making of Deep RL. AI chatbots are intriguing in that they provide natural-language answers to any natural-language question posed to them. However, as a downside of this flexibility, the underlying large language model may "hallucinate", i.e., generate nonsensical text unfaithful to the provided source input [16]. This means AI chatbots may deliver explanations that do not faithfully explain the decision-making of RL, i.e., the explanations may exhibit low fidelity [10]. In addition, AI chatbots may provide different explanations for the very same question asked, i.e., the explanations may exhibit low stability [38].

To deliver natural-language explanations with high fidelity and high stability, **Chat4XAI** uses dedicated prompt engineering for the AI chatbot and careful selection of the hyper-parameters of the underlying large language model. Prompt engineering helps increase the correctness of the answers by providing a set of targeted, initial questions (a.k.a. prompts) before the actual question [40, 42].

We prototypically realize and evaluate **Chat4XAI** using OpenAI’s ChatGPT Completion API. We evaluate the fidelity and stability of the explanations delivered by **Chat4XAI** using an adaptive cloud service exemplar realized using Double DQN as Deep RL algorithm. We assess **Chat4XAI** for different prompting strategies, open as well as closed questions, and different hyper-parameter settings. To contextualize the performance of **Chat4XAI**, we compare **Chat4XAI**’s explanations with how well human software engineers were able to understand the decision-making of Deep RL using visual explanations [25].

Paper Structure. Sect. 2 describes the conceptual architecture and proof-of-concept implementation of **Chat4XAI**. Sect. 3 provides the experiment design and results. Sect. 4 discusses limitations and future enhancements. Sect. 5 relates **Chat4XAI** to existing work.

2 Chat4XAI Architecture and Realization

Fig. 1 shows the main conceptual components of **Chat4XAI** and how they may be embedded to realize an explainable service-oriented system. Fig. 1 also shows the control and data flow among these components and in which order this happens (shown as numbers in black circles). **Chat4XAI** leverages the output of XRL-DINE, a state-of-the-art explanation technique [6], to generate natural-language explanations. Below we explain the different conceptual components of **Chat4XAI** followed by how we prototypically realized **Chat4XAI** via ChatGPT.

2.1 XRL-DINE

XRL-DINE generates different types of so-called *Decomposed Interestingness Elements* (DINEs), which provide insights into the decision-making of Deep RL. In the original approach in [6], DINEs are visualized in the XRL-DINE graphical user interface. Here, we use the information of the DINEs as input to **Chat4XAI**.

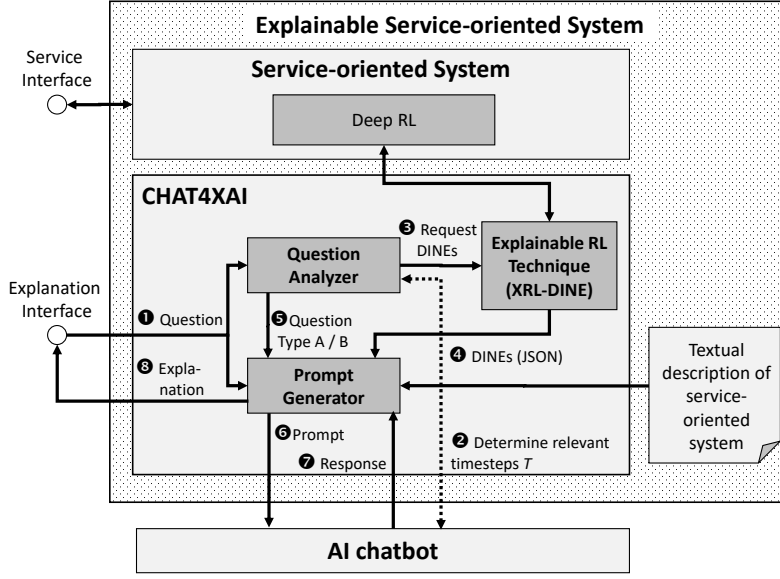


Fig. 1. Chat4XAI conceptual architecture and embedding into service-oriented system

XRL-DINE combines and enhances two explanation techniques: *Interestingness Elements* facilitate identifying situations where the decision-making of Deep RL is uncertain. It thereby helps select interesting decisions among the many decisions taken by Deep RL at runtime [39]. Situations, where Deep RL is uncertain, may point to states which require more training, or where the reward function may need to be re-engineered to provide stronger rewards. *Reward Decomposition* splits the reward function into sub-functions, called reward channels, each of which reflects a different aspect of the learning goal [17]. For each reward channel, a separate decision-making policy is learned. To select a concrete action, the individual decision-making policies are combined and an action is selected from this combined decision-making policy. The trade-offs between the different learning goal become observable via these reward channels.

Technically, the information of the DINES is encoded in JSON format. JSON is an open-standard data interchange format using human-readable text, and thus can serve directly as input to the AI chatbot. To illustrate, the JSON snippet below gives an example of the information contained within a “Relative Reward Channel Dominance” DINE. This type of DINE provides the actual contribution of each reward channel to the aggregated decision.

```
"Action 1": { "Reward Channel A": 0.35,
              "Reward Channel B": 2.61,
              "Reward Channel C": 1.19 },
"Action 2": { "Reward Channel A": 0.13,
              "Reward Channel B": 0.0,
              "Reward Channel C": 1.01 }, ...
```

In this example, *Action 1* is chosen, as it has the highest relative reward with *Reward Channel B* contributing most to the aggregate decision.

2.2 Question Analyzer

The Question Analyzer classifies a given question into one of two question types: **Question Type A** concerns a *single* decision, i.e., it covers the decision taken at a single timestep. One example is to ask why Deep RL decided for adaptation X rather than Y at timestep t .

Question Type B concerns a sequence of decisions, i.e., it covers the decision trajectory of several timesteps. One example is to ask how often, along the concerned timesteps t_k to t_l , Deep RL was uncertain in its decisions.

The reason for classifying a question into one of these two types is twofold. First, while state-of-the-art AI chatbots are capable of directly answering natural-language questions (so-called 'zero-shot' prompting), the quality of the answers can be increased via prompt engineering, i.e., by providing a set of initial questions (a.k.a. prompts) before the actual question to be answered [40, 42]. Differentiating between the two types of questions allows for a more targeted prompt engineering. Second, AI chatbots typically limit the cumulative length of questions and answers per conversation (see Section 2.4). Depending on the question type, we can filter the number and types of relevant DINEs provided as input to the AI chatbot to remain within length limits.

To identify the question type, we ask the AI chatbot to provide us with a list T of all relevant timesteps mentioned in the question. Then, by counting the size of T we can identify whether it is a question of Type A ($|T| = 1$) or Type B ($|T| > 1$). Note that in the case of $|T| = 0$, one may use a default set of timesteps; e.g., the 20 most recent ones, i.e., $T = (t_{now-20}, \dots, t_{now})$.

Depending on the question type, the DINEs for a single time step (Type A questions) or for multiple time steps (Type B questions) are requested from XRL-DINE and forwarded to the Prompt Generator.

2.3 Prompt Generator

The Prompt Generator receives the following pieces of information:

- A textual *description of the service-oriented system* providing relevant concepts, including domain-specific terms, the different actions available, and the learning goals. These concepts provide relevant service-specific knowledge to the AI chatbot, as the DINEs also refer to these concepts. If Chat4XAI is used to deliver an explanation interface to service users (as depicted in Figure 1), this description is provided by the service developer or provider.
- The *question type* as identified by the Question Analyzer.
- The *DINEs* in JSON format for the relevant timesteps T .
- The *actual question*, which is provided by the explainee.

The Prompt Generator generates and issues the following prompts:

Prompt 1 provides relevant concepts to the AI chatbot by providing it with the textual description of the service-oriented system, introduced by the text `The following scenario description will be available...`

Prompt 2 provides the context for the answers to be given by the AI chatbot. It gives the type of data that will follow after this prompt (which depends on the question type) together with a reference to the service-oriented system to allow the AI chatbot to connect to the description given in Prompt 1:

- *Question Type A*: You will be given the state for a single timestep of <name from scenario> as JSON enclosed in ***:
- *Question Type B*: You will be given a trajectory of timesteps for <name> as JSON enclosed in ***:

Prompt 3 provides the actual DINEs in JSON format enclosed in *** to mark the boundaries between the DINEs and the final question in Prompt 4.

Prompt 4 provides the actual question of the explainee.

Typically, an explainee will ask open questions (i.e., without any fixed set of answers to choose from) and expect an answer in concrete terms. Directly asking open Type B questions turned out to be a too challenging for the AI chatbot. We thus employ "chain of thought" prompt engineering, in which the AI chatbot is given smaller subtasks leading up to the final results. In a first prompt, we ask the AI chatbot to provide us with a list of relevant timesteps; e.g., the ones where Deep RL is uncertain. In a second prompt, we then ask the AI chatbot concrete questions about these relevant timesteps; e.g., to count these.

2.4 Proof-of-concept Realization

We developed our proof-of-concept realization of Chat4XAI in Python using the OpenAI ChatGPT Completion API¹. We are using GPT 3.5 turbo as a large language model, which offers a faster generation of responses, and is capable of providing more in-depth answers. The following hyper-parameters are considered in our realization and experiments:

- *n* gives how many responses to generate for each prompt.
- *max_token* limits the length of the answer. The OpenAI API imposes two constraints on the overall length of a request, i.e., a sequence of prompts and responses. First, there is an overall token limit of 4,096 tokens (ca. 3,000 words) per request. Second, there is a limit of 90,000 tokens per minute. *max_token* helps to set a trade-off between these two constraints.
- *temperature* controls text generation behavior. Temperature $\in [0, 2]$ controls the randomness of the text, with a higher temperature resulting in more "creative" text (but with a higher risk of hallucinations). A temperature of 0 leads to deterministic text generation behavior.
- *Top-p sampling* is an alternative to temperature. Instead of considering all possible tokens, only a subset of tokens is considered whose cumulative probability adds up to $\text{top-p} \in [0, 1]$.

Table 1 gives examples of explanations generated via the proof-of-concept realization of Chat4XAI.

¹ <https://platform.openai.com/docs/api-reference>

Table 1. Examples for natural-language explanations generated by Chat4XAI

Question	Explanation
What is the main goal that the service-oriented system wants to achieve?	The main goal that the service-oriented system wants to achieve is to increase User Satisfaction.
Why did Deep RL chose the action “X” instead of “Y” at at timestep t_k ?	Based on the information provided, Deep RL chose the action “X” instead of “Y” because the action “X” has a higher positive impact on the goals User Satisfaction (85.0%) and Running Costs (15.0%) compared to the action “Y”.
Which goal would the service-oriented system achieve when choosing action “A” instead of performing no action at timestep t_i ?	If Deep RL selects “A” instead of performing no action at timestep t_i , the service-oriented system would achieve the goal of decreasing running costs. The “A” action has the highest reward for the Running Costs goal compared to all other actions. However, Deep RL has chosen no action because it has a higher positive impact on the User Satisfaction goal.

3 Experiments

We perform a set of controlled experiments to evaluate the fidelity and stability of Chat4XAI’s explanations depending on different configurations of our explanation technique. We also compare the results of Chat4XAI with the results of an empirical user study from the literature [25]. This user study assessed how well software engineers were able to understand the decision-making of Deep RL by only using explanations in visual form, i.e., using the XRL-DINE graphical user interface. To facilitate reproducibility, relevant background and supplementary material is available from <https://gitlab.com/xrl2/Chat4XAI/>.

3.1 Metrics

We chose the following metrics from the explainable AI literature for assessing Chat4XAI [10, 38]:

Fidelity expresses how well the explanations reflect the behavior of the black-box model. As we use AI chatbots to generate explanations, Chat4XAI exhibits the risk of hallucination, i.e., producing answers not corresponding to the learned Deep RL decision-making policy. We quantify fidelity by measuring the rate of correct explanations for m given questions. Let $x_i = 1$ mean correct explanation, and 0 otherwise, then fidelity is computed as $\sum x_i / m$.

Stability gives the degree to which the same explanation is generated for the same input. Explanations of Chat4XAI for the same input may vary because the AI chatbot may produce non-deterministic results depending on the chosen hyperparameter settings. We measure stability as $1 - \sigma$, with σ being the standard deviation of fidelity across several experiment repetitions for the same input and configuration of independent variables (see Section 3.3).

We seek to directly compare the performance of Chat4XAI with the performance of software engineers using XRL-DINE. We, therefore, selected the following metric from the user study of XRL-DINE [25]:

Effectiveness quantifies the performance of software engineers by measuring the rate of correctly answered questions concerning Deep RL decision-making among n participants. Let $c_i = 1$ mean correct explanation, and 0 otherwise, then the effectiveness is computed as $\sum c_i/n$.

3.2 Experiment Environment

To be able to compare the results for Chat4XAI with the results of the user study reported in [25], we use the same environment as in the user study. In particular, this means we use the same set of DINEs covering the same 21 timesteps concerning the decision-making of the chosen service-oriented system exemplar realized using Double DQN with Experience Replay as Deep RL algorithm.

The service-oriented system exemplar SWIM represents an adaptive multi-tier webshop [30]. SWIM simulates an actual web shop, while allowing to speed up the simulation to cover longer periods of time. The goal of adaptation in SWIM is to maximize a given utility function in the presence of varying workloads. SWIM can be adapted by (1) adding / removing web servers, and (2) changing the proportion of requests for which optional, computationally intensive recommendations are generated. While both types of adaptations have an impact on user satisfaction (due to their influence on throughput and response time), adaptations of type (1) have an impact on costs (due to the costs of more/fewer servers), and adaptations of type (2) have an impact on revenue (due to recommendations leading to potential further purchases in the webshop). We use the same decomposed reward function as introduced in [6], which aims to balance the aforementioned conflicting QoS goals via the weights a , b , and c .

$$R_{\text{total}} = a \cdot R_{\text{user_satisfaction}} + b \cdot R_{\text{revenue}} + c \cdot R_{\text{costs}}$$

User Study. Below, we provide key aspects of the user study from [25] to provide a context for the results reported in Section 3.4. The user study involved 54 software engineers from academia (82%) and industry (8%), most of which held an academic degree in software engineering or related fields (35% Bachelor’s degree, 41% Master’s degree, 11% PhD). The user study was carried out using an online questionnaire. Study participants were asked eight closed questions (see Section 3.3) and were provided with several single-choice answers for each question. Study participants also received a description of the service-oriented system, which included: (1) an explanation of the SWIM exemplar; (2) a list of the QoS goals, among which Deep RL should seek a trade-off; (3) a list of possible adaptations (i.e., RL actions), together with examples of typical effects of these adaptations.

3.3 Experiment Setup

Independent Variables. To analyze the performance of Chat4XAI, we varied the following three main independent variables.

Prompting: We analyze the performance depending on whether (i) zero-shot prompting or (ii) prompt engineering is used. This indicates how much

Table 2. Questions for which explanations should be given²

Type A Questions Focussing on a single timestep	Type B Questions Covering a sequence of timesteps
Q2: At timestep 10, which action did the service choose?	Q1: For timesteps 22,575—22,596, how many adaptations were executed by the service?
Q4: At timestamp 8, why did the service choose the adaptation „Decrease Dimmer“ instead of „Add Server“?	Q3: For timesteps 22,575—22,596, how often is the service uncertain when making a decision?
Q5: At timestamp 15, why did the service choose “No Adaptation” instead of „Add Server“?	Q8: For timesteps 22,575—22,596, what is the main QoS goal that the service wants to achieve?
Q6: At timestamp 13, which QoS goal would the service achieve when selecting „Remove Server“ instead of „No Adaptation“?	
Q7: At timestamp 9, which QoS goal would the service achieve when selecting „Add Server“ instead of „Remove Server“?	

Chat4XAI’s performance is impacted by prompt engineering, resp. how robust the approach would be independent of any specific prompt engineering.

Question Form: We analyze the performance of Chat4XAI in providing explanations for open questions (which would be the typical scenario in a practical setting) and closed questions (like in the user study from [25]).

Hyper-parameters: We analyze the performance of Chat4XAI for different concrete settings of temperature and top.p. We vary temperature $\in \{0, 0.2, 0.5, 1\}$. For each temperature, we vary top.p $\in \{1, 0.8, 0.5\}$ and report the aggregated results per temperature. We use 54 repetitions to get the same number of explanations as in the user study. We split these 54 repetitions into three clusters of 18 repetitions, with each cluster having a different top.p setting. To efficiently execute the experiment, we set $n = 18$, i.e., retrieve 18 answers per prompt. Also, we set $max_token = 350$, delivering a good trade-off between length of answers and throughput.

Questions to be Explained. To assess the performance of Chat4XAI, we have to choose concrete questions that are posed to the AI chatbot to retrieve explanations. As we are interested in comparing the performance of Chat4XAI with that of software engineers from the empirical user study in [25], we use the same set of questions that were formulated there. These questions are shown in Table 2.

3.4 Experiment Results

Table 3 presents the overall results concerning the fidelity and stability of Chat4XAI as well as the effectiveness of software engineers from the user study.

As expected Chat4XAI with prompt engineering outperforms Chat4XAI with zero-shot prompting. For closed questions, zero-shot prompting only achieves an average fidelity of 48% with an average stability of 50%. Prompt engineering achieves an average fidelity of 97% with an average stability of 85%, because the underlying large language model has been provided with better information about the scope and nature of answers expected via initial prompts.

² Questions are numbered in the order they were asked user study participants. Question text adapted from [25] and edited for clarity.

Table 3. Experiment Results

Question Nbr	Question Type	Fidelity of Chat4XAI												Effective-ness of Software Engineers
		Closed Questions								Open Questions				
		Zero-shot Prompting				Prompt Engineering				Prompt Engineering				
		Temperature =				Temperature =				Temperature =				
		0,0	0,2	0,5	1,0	0,0	0,2	0,5	1,0	0,0	0,2	0,5	1,0	
Q1	A	0%	6%	4%	6%	100%	100%	85%	61%	100%	89%	69%	67%	65%
Q2	B	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	91%
Q3	A	100%	100%	100%	98%	100%	100%	100%	100%	100%	98%	100%	98%	94%
Q4	B	33%	96%	78%	50%	100%	100%	98%	96%	100%	98%	78%	78%	61%
Q5	B	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	96%	63%
Q6	B	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	69%	56%	80%
Q7	B	0%	0%	0%	0%	100%	100%	94%	87%	100%	100%	100%	96%	69%
Q8	A	0%	0%	0%	0%	100%	100%	98%	87%	100%	65%	41%	35%	83%
Average		42%	50%	48%	44%	100%	100%	97%	91%	100%	94%	82%	78%	
		46%				97%				88%				76%
		Stability of Chat4XAI												StdDev
		51%	50%	50%	50%	100%	100%	83%	72%	100%	76%	62%	59%	
		50%				89%				74%				44%

Indeed, prompt engineering together with low-temperature settings can lead to a fidelity of 100% with a stability of 100%, because the underlying large language model gives deterministic answers. Here, increasing the temperature leads to a lower fidelity and lower stability, as the chance of wrong answers being created from an already strong baseline of correct explanations increases.

Interestingly, when considering zero-shot prompting, a temperature larger than 0 leads to a higher fidelity albeit with the same low stability. The higher temperature and the resulting more "creative" answers from the AI chatbot appear to increase the chance of providing a correct explanation as we start from a very weak baseline of correct explanations.

Finally, as one might expect, providing correct explanations for the open questions is more challenging. Except for temperature = 0, the fidelity for open questions is generally lower than for closed questions (88% vs. 97% on average) and also exhibits a lower stability (74% vs. 89% on average), indicating a higher randomness of answers.

Comparing the fidelity of Chat4XAI with the effectiveness of software engineers, Chat4XAI was able to outperform the software engineers for 8 out of the 12 experiment configurations and was able to answer all eight questions correctly (100% fidelity) in 3 configurations. According to [25], only 33% of the software engineers were able to answer all eight questions correctly.

3.5 Validity Risks

Concerning internal validity, we addressed the risk that results for Chat4XAI may have been achieved by chance. To this end, we carefully controlled experimental variables as introduced above. In addition, we repeated the experiment multiple times to account for the typical stochastic effects and thus variance of machine learning models [35]. While we chose metrics for Chat4XAI and the user study

that allow numerically comparing them, semantically they are not fully comparable. To do so would require complementing the Chat4XAI results by user studies that assess how well explainees could use the natural-language explanations to understand Deep RL decision making.

Concerning external validity, we chose a concrete service exemplar (SWIM) together with real-world workload traces and an actual subset of the interactions between Deep RL and the service environment. Still, our experiments cover only one concrete problem in service-oriented systems (i.e., service adaptation) and use only one concrete service-oriented system exemplar. We cover different styles of questions ("what/which", "why", "how many") reflecting various insights into the decision-making of Deep RL. Yet, we limited the questions to the ones from the user study in [25] to compare the performance of Chat4XAI with that of software engineers. While we designed Chat4XAI to be as generic as possible and thus applicable to different problems in service-oriented systems, experimental results are limited with respect to generalizability.

4 Current Limitations and Potential Enhancements

Multi-round Question Answering. Currently, Chat4XAI generates a natural-language explanation for a given question. Enhancing Chat4XAI to also allow for follow-up questions thus appears as a natural next step. An interesting further direction for such multi-round question answering is to follow the metaphor of the Socratic dialogue. A Socratic dialogue may take the form of a cooperative argumentative dialogue between the explainer and explainee, where the explainer initiates the dialogue by asking questions to stimulate ideas by the explainee [42]. This may especially help non-technical users to come up with concrete questions concerning the decision-making of Deep RL.

Coping with Missing Insights. Once an explainee is provided with the powerful natural-language interface of an AI chatbot, the explainee may ask questions concerning the decision-making of Deep RL for which the underlying explainable AI technique (here: XRL-DINE) does not provide the required insights. One may check whether the answer given by the AI chatbot is backed by actual insights and inform the explainee accordingly. Here, work on explainable large language models may be leveraged³.

Protecting Sensitive Information. AI chatbots and thus Chat4XAI may be tricked into revealing information that should not be given away [11, 24]. One open question thus is how to leverage AI chatbots for explainable service-oriented systems while protecting sensitive information of the service provider [7].

Explanations for Policy-based Deep RL. Chat4XAI only works for value-based Deep RL because the underlying XRL-DINE technique needs access to the learned decision-making policy in the form of an action-value function $Q(S, A)$, giving the expected cumulative reward when executing action A in state S . In contrast to value-based Deep RL, policy-based Deep RL has the important

³ e.g., see <https://docs.aleph-alpha.com/docs/explainability/explainability/>

advantage that it can naturally cope with concept drifts (e.g., see [12, 28, 34] for its use in service-oriented systems). However, policy-based Deep RL does not use an action-value function $Q(S, A)$, because the fundamental idea is to directly use and optimize a parametrized stochastic action selection policy $\pi_\theta(S, A)$ in the form of a deep artificial neural network.

5 Related Work

As introduced in Sect. 1, RL – and recently Deep RL – was successfully applied to different problems in service-oriented systems [12–14, 32, 37, 43]. While existing papers provide extensive and systematic evaluations of the performance of Deep RL for service-oriented systems, they did not yet address the problem of how to explain the decision-making of Deep RL using natural language.

There exists related work on natural-language explanations in the general area of explainable AI (XAI), which can be grouped in two main clusters.

Requirements Elicitation. Jentzsch et al. perform an empirical study to elicit user expectations towards chatbots for XAI [15]. Results indicate that questions of users differ concerning the question form (open vs. closed), the level of abstraction, and the temporal scope (e.g., past, current or future outcomes). Kuzba and Biecek elicit typical questions a human would ask a chatbot [18]. From around 600 collected dialogues, they distill the 12 most common types of questions. Liao et al. construct a corpus of questions via literature review, as well as expert reviews and interviews [19]. The resulting corpus contains 73 types of questions in 10 categories. Gao et al. introduce a chatbot-based explanation framework built using IBM Watson Assistant [8]. Similar to Kuzba and Biecek, they use this framework to understand what users would like to know about AI and elicit concrete user requests about AI-generated results.

All these works provide interesting insights into what questions may be asked, but are limited to supervised learning. Also, in contrast to these works, **Chat4XAI** does not require the up-front elicitation and fine-grained classification of questions. **Chat4XAI** works with two types of questions only because more specific aspects are handled naturally by the underlying powerful large language model.

Design and Realization. Carneiro et al. suggest combining a chatbot with an explainable model serving as a less complex surrogate than the actual black-box prediction model [4]. They use natural language understanding to extract structured information from user questions concerning the user intent and the entity concerned with the question. Nguyen et al. leverage classical conversational agent architectures for question-answer dialogues [33]. They (1) construct a question phrase bank, (2) establish a mapping between questions and explainable AI techniques, and (3) use template-based natural language generation to create explanations. Malandri et al. consider the knowledge and experience of the users when generating explanations [22]. They explicitly introduce "clarification" as a further dialogue type on top of an earlier explanation framework [21]. Their user study indicated that different user groups perceive explanations dif-

ferently, that all user groups prefer textual explanations over graphical ones, and that clarifications can enhance the usefulness of explanations.

All these works provide evidence for the benefits of using natural-language explanations. However, they do not deliver explanations for Deep RL. Also, they are all built using classical dialogue systems and natural language processing, and thus do not leverage the capabilities of modern large language models.

6 Conclusion

We took a first step towards natural-language explanations of Deep RL used for realizing service-oriented systems. We introduced **Chat4XAI**, an explainable AI technique powered by modern AI chatbot technology built on top of large language models. We performed a proof-of-concept implementation for **Chat4XAI** using ChatGPT. Experimental results suggest that **Chat4XAI** can provide explanations with high fidelity and stability.

In future work, we will work on the potential enhancements of **Chat4XAI** described above. We will also extend our experiments to other service-oriented system exemplars, which cover additional problems, such as service composition. These experiments will be complemented by user studies to assess how useful and easy-to-use natural-language explanations are.

Acknowledgments. We cordially thank Xhulja Shahini for her comments on earlier drafts of the paper. Research leading to these results received funding from the EU’s Horizon Europe R&I programme under grant 101070455 (DynaBIC).

References

1. Baresi, L., Nitto, E.D., Ghezzi, C.: Toward open-world software: Issue and challenges. *Computer* **39**(10), 36–43 (2006)
2. Cambria, E., Malandri, L., Mercorio, F., Mezzanzanica, M., Nobani, N.: A survey on XAI and natural language explanations. *Inf. Process. Manag.* **60**(1) (2023)
3. Camilli, M., Mirandola, R., Scandurra, P.: XSA: explainable self-adaptation. In: 37th Intl Conf on Automated Software Engineering (ASE 2022). ACM (2022)
4. Carneiro, D., Veloso, P., Guimarães, M., Baptista, J., Sousa, M.: A conversational interface for interacting with machine learning models. In: 4th Intl Workshop on eXplainable and Responsible AI and Law. CEUR Workshop Proceedings, vol. 3168. CEUR-WS.org (2021)
5. Dewey, D.: Reinforcement learning and the reward engineering principle. In: 2014 AAAI Spring Symposium, Stanford University, Palo Alto, California, USA, March 24–26, 2014. AAAI Press (2014)
6. Feit, F., Metzger, A., Pohl, K.: Explaining online reinforcement learning decisions of self-adaptive systems. In: Intl Conf on Autonomic Computing and Self-Organizing Systems, ACSOS 2022. IEEE (2022)
7. Følstad, A. et al.: Future directions for chatbot research: an interdisciplinary research agenda. *Computing* **103**(12), 2915–2942 (2021)
8. Gao, M., Liu, X., Xu, A., Akkiraju, R.: Chat-xai: A new chatbot to explain artificial intelligence. In: Arai, K. (ed.) *Intelligent Systems Conference (IntelliSys 2021)*. Lecture Notes in Networks and Systems, vol. 296. Springer (2021)

9. Ghanadbashi, S., Safavifar, Z., Taebi, F., Golpayegani, F.: Handling uncertainty in self-adaptive systems: an ontology-based reinforcement learning model. *Journal of Reliable Intelligent Environments* (2023)
10. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. *ACM Comput. Surv.* **51**(5) (2019)
11. Hasal, M., Nowaková, J., Saghair, K.A., Abdulla, H.M.D., Snásel, V., Ogiela, L.: Chatbots: Security, privacy, data protection, and social aspects. *Concurr. Comput. Pract. Exp.* **33**(19) (2021)
12. Huang, V., Wang, C., Ma, H., Chen, G., Christopher, K.: Cost-aware dynamic multi-workflow scheduling in cloud data center using evolutionary reinforcement learning. In: *ICSOC 2022. LNCS*, vol. 13740. Springer (2022)
13. Iftikhar, S. et al.: AI-based fog and edge computing: A systematic review, taxonomy and future directions. *Internet Things* **21** (2023)
14. Jamil, B., Ijaz, H., Shojafar, M., Munir, K., Buyya, R.: Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. *ACM Comput. Surv.* **54**(11s) (2022)
15. Jentzsch, S.F., Höhn, S., Hochgeschwender, N.: Conversational interfaces for explainable AI: A human-centred approach. In: *Intl Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems. LNCS*, vol. 11763. Springer (2019)
16. Ji, Z. et al.: Survey of hallucination in natural language generation. *ACM Comput. Surv.* **55**(12) (2023)
17. Juozapaitis, Z., Koul, A., Fern, A., Erwig, M., Doshi-Velez, F.: Explainable reinforcement learning via reward decomposition. In: *IJCAI/ECAI Workshop on Explainable Artificial Intelligence* (2019)
18. Kuzba, M., Biecek, P.: What would you ask the machine learning model? identification of user needs for model explanations based on human-model conversations. In: Koprinska, I., et al. (eds.) *ECML PKDD 2020 Workshops. Communications in Computer and Information Science*, vol. 1323. Springer (2020)
19. Liao, Q.V., Gruen, D.M., Miller, S.: Questioning the AI: informing design practices for explainable AI user experiences. In: *Conference on Human Factors in Computing Systems (CHI '20)*. ACM (2020)
20. Ma, W., Xu, H.: Skyline-enhanced deep reinforcement learning approach for energy-efficient and qos-guaranteed multi-cloud service composition. *Applied Sciences* **13**(11) (2023)
21. Madumal, P., Miller, T., Sonenberg, L., Vetere, F.: A grounded interaction protocol for explainable artificial intelligence. In: *18th Intl Conf on Autonomous Agents and MultiAgent Systems, AAMAS19. Intl Found. for Autonomous Agents and Multiagent Systems* (2019)
22. Malandri, L., Mercorio, F., Mezzanzanica, M., Nobani, N.: Convxai: a system for multimodal interaction with any black-box explainer. *Cogn. Comput.* **15**(2), 613–644 (2023)
23. Mariotti, E., Alonso, J.M., Gatt, A.: Towards harnessing natural language generation to explain black-box models. In: *2nd Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence. ACL* (2020)
24. Maslej, P. et al.: The AI index 2023 annual report. Tech. rep., AI Index Steering Committee, Institute for Human-Centered AI, Stanford University (April 2023)
25. Metzger, A., Laufer, J., Feit, F., Pohl, K.: A user study on explainable online reinforcement learning for adaptive systems. *CoRR* **abs/2307.04098** (2023)

26. Metzger, A., Quinton, C., Mann, Z.Á., Baresi, L., Pohl, K.: Realizing self-adaptive systems via online reinforcement learning and feature-model-guided exploration. *Computing* (March, 2022)
27. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* **267** (2019)
28. Mo, R., Xu, X., Zhang, X., Qi, L., Liu, Q.: Computation offloading and resource management for energy and cost trade-offs with deep reinforcement learning in mobile edge computing. In: *ICSOC 2021. LNCS*, vol. 13121. Springer (2021)
29. Mohseni, S., Zarei, N., Ragan, E.D.: A multidisciplinary survey and framework for design and evaluation of explainable AI systems. *ACM Trans. Interact. Intell. Syst.* **11**(3-4), 24:1–24:45 (2021)
30. Moreno, G.A., Schmerl, B.R., Garlan, D.: SWIM: an exemplar for evaluation and comparison of self-adaptation approaches for web applications. In: *13th Intl Conf on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2018. ACM* (2018)
31. Motger, Q., Franch, X., Marco, J.: Software-based dialogue systems: Survey, taxonomy, and challenges. *ACM Comput. Surv.* **55**(5) (2023)
32. Mutanu, L., Kotonya, G.: State of runtime adaptation in service-oriented systems: what, where, when, how and right. *IET Softw.* **13**(1), 14–24 (2019)
33. Nguyen, V.B., Schlötterer, J., Seifert, C.: Explaining machine learning models in natural conversations: Towards a conversational XAI agent. *CoRR abs/2209.02552* (2022)
34. Palm, A., Metzger, A., Pohl, K.: Online reinforcement learning for self-adaptive information systems. In: *Int'l Conference on Advanced Information Systems Engineering, CAiSE'20* (2020)
35. Pham, H.V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., Yu, Y., Nagappan, N.: Problems and opportunities in training deep learning software systems: An analysis of variance. In: *35th Intl Conf on Automated Software Engineering (ASE 2020)*. IEEE (2020)
36. Puiutta, E., Veith, E.M.S.P.: Explainable Reinforcement Learning: A Survey. In: *Machine Learning and Knowledge Extraction - 4th IFIP Cross-Domain Conference, CD-MAKE 2020. LNCS*, vol. 12279. Springer (2020)
37. Razian, M.R., Fathian, M., Bahsoon, R., Toosi, A.N., Buyya, R.: Service composition in dynamic environments: A systematic review and future directions. *J. Syst. Softw.* **188**, 111290 (2022)
38. Robnik-Sikonja, M., Bohanec, M.: Perturbation-based explanations of prediction models. In: Zhou, J., Chen, F. (eds.) *Human and Machine Learning - Visible, Explainable, Trustworthy and Transparent. Human-Computer Interaction Series*, Springer (2018)
39. Sequeira, P., Gervasio, M.T.: Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artif. Intell.* **288** (2020)
40. Strobelt, H., Webson, A., Sanh, V., Hoover, B., Beyer, J., Pfister, H., Rush, A.M.: Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE Trans. Vis. Comput. Graph.* **29**(1), 1146–1156 (2023)
41. Sutton, R.S., Barto, A.G.: *Reinforcement learning: An introduction*. MIT press (2018)
42. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. *CoRR abs/2302.11382* (2023)

43. Yu, Z., Zhao, S., Su, T., Liu, W., Liu, X., Wang, G., Wang, Z., Leung, V.C.M.: DeepSCJD: An online deep learning-based model for secure collaborative job dispatching in edge computing. In: ICSOC 2022. LNCS, vol. 13740. Springer (2022)