Can LLMs Generate Architectural Design Decisions? - An Exploratory Empirical study

Rudra Dhar
Software Engineering Research Centre
IIIT Hyderabad, India
rudra.dhar@research.iiit.ac.in

Karthik Vaidhyanathan
Software Engineering Research Centre
IIIT Hyderabad, India
karthik.vaidhyanathan@iiit.ac.in

Vasudeva Varma
Language Technologies Research Centre
IIIT Hyderabad, India
vv@iiit.ac.in

Abstract—**Architectural Knowledge Management (AKM) involves the organized handling of information related to architectural decisions and design within a project or organization. An essential artefact of AKM is the Architecture Decision Records (ADR), which documents key design decisions. ADRs are documents that capture decision context, decision made and various aspects related to a design decision, thereby promoting transparency, collaboration, and understanding. Despite their benefits, ADR adoption in software development has been slow due to challenges like time constraints and inconsistent uptake. Recent advancements in Large Language Models (LLMs) may help bridge this adoption gap by facilitating ADR generation. However, the effectiveness of LLM for ADR generation or understanding is something that has not been explored. To this end, in this work, we perform an exploratory study which aims to investigate the feasibility of using LLM for the generation of ADRs given the decision context. In our exploratory study, we utilize GPT and T5-based models with 0-shot, few-shot, and fine-tuning approaches to generate the Decision of an ADR given its Context. Our results indicate that in a 0-shot setting, state-of-the-art models such as GPT-4 generate relevant and accurate Design Decisions, although they fall short of human-level performance. Additionally, we observe that more cost-effective models like GPT-3.5 can achieve similar outcomes in a few-shot setting, and smaller models such as Flan-T5 can yield comparable results after fine-tuning. To conclude, this exploratory study suggests that LLM can generate Design Decisions, but further research is required to attain human-level generation and establish standardized widespread adoption.**

Index Terms—**ADR, LLM**

I. INTRODUCTION

Architectural Knowledge Management (AKM) is the organized management of information on architectural decisions and designs within a project or organization. This involves capturing architectural knowledge, styles, design patterns, and quality attributes in a centralized repository. The focus is on ensuring traceability of decisions, promoting collaboration, facilitating knowledge reuse, and offering decision support. AKM aims to enhance communication, learning, and decision-making, contributing to the success of software development projects or other endeavours with complex architectures.

For several decades, the significance of AKM has been widely recognized, and various tools have been developed to assist AKM. [1] Rainer et al. [2] state that while a huge amount of work is done to support AKM activities and capture Architectural Knowledge, it has not been sufficient. This has

been a crucial reason restricting a wider adoption of AKM approaches, and more research is needed for automatically capturing this knowledge [3].

An *Architecture Decision Record (ADR)* is a crucial part of AKM. It entails the idea that software architecture is considered a set of Design Decisions [4]. It is a document used in software development to capture and document important Architecture Design Decisions (ADD), made during the design and development of a software system. A detail explanation is given in Section II Despite the well-established benefits of ADRs, their adoption has been slow to non-existent as described by Georg *et al.* [5]. Unsuccessful adoption of ADRs in software development can occur due to various reasons like, inadequate training, overly complex processes, time constraints, inconsistent adoption, low prioritization, and failure to update and communicate ADRs.

Large Language models (LLMs) excel in comprehending contexts and generating text accordingly. Over the recent years due to advancement of LLMs, text generation has become more accessible. This paper delves into the exploration of whether LLMs can effectively generate Architectural Decision Records (ADRs). While the prospect of generating entire ADRs from a codebase remains a task for future endeavours, the focus of this work is on utilizing LLMs to generate Design Decisions from decisions Contexts as these are recognized as the core components of an ADR.

In the realm of *Natural Language Processing (NLP)*, text generation is typically addressed through three distinct methodologies when utilizing LLMs. These approaches include *zero-shot prompting, few-shot prompting*, and *fine-tuning*. Section II provides a comprehensive exploration of the advantages and disadvantages associated with each approach [6]. It is recommended to initially employ zero-shot prompting, followed by few-shot prompting, and finally, fine-tuning due to the escalating complexity of implementation.

In this exploratory empirical study, we commence by defining the experimental subject, which involves gathering ADRs and choosing the LLMs for the study. Next we perform experiments deploying the 3 approaches mentioned above. In the zero-shot scenario, the model generates Decisions solely based on Context. Whereas in the few-shot approach, the model is *trained in-context* on a set of ADR examples. Lastly, the paper delves into fine-tuning, where a generative model is

trained to produce Decisions from Contexts.

Our results show that LLMs do exhibit noteworthy capabilities in generating Architectural Design Decisions. While GPT-4 excels in 0-shot prompting, smaller models like text-davinci-003 and Flan-T5-base achieve comparable results with few-shot prompts and fine-tuning respectively. Smaller, fine-tuned models, like Flan-T5-base, requiring minimal hosting infrastructure, can prove useful for ADR generation within organizational infrastructure, particularly in privacy or security-sensitive scenarios. We conclude that despite LLMs not being entirely dependable for ADR generation, they can effectively assist architects in documenting ADD.

Our experimental scripts and results are made available with an open-source license, to enable the independent verification and replication of the results presented in this study: https://anonymous.4open.science/r/LLMforADR-401C/README.md

The remainder of this paper is structured as follows. Section II presents background concepts on ADR, LLM, and text generation. Section III presents the related work on tools and technologies for AKM and ADR. Section IV details the overall approach and the study design. Section V describes the results of the different experiments performed, and Section VI presents the related discussion. The threats to the validity of this study are thoroughly analyzed in Section VII. Finally, Section VIII documents our conclusions and future work.

II. BACKGROUND

This section describes some key concepts and ideas that are used in this study, namely, ADRs and LLMs, and approaches of text generation.

A. Architectural Decision Record (ADR)

Software Architecture can be represented as a set of Design Decisions [4]. An Architecture Decision Record (ADR) is a software development document that captures and documents crucial architectural decisions made throughout a project. It encompasses information about the *context* of the problem, the *decision* taken, the anticipated consequences of the decision, and relevant references. ADRs are instrumental in promoting transparency, fostering collaboration, and preserving the historical background of architectural choices. They play a vital role in project documentation and ensuring that decision-making is well-informed. The primary elements of an ADR comprise the problem context and the associated decision.

B. Large Language Models (LLM)

A Language Model is a probabilistic model of a natural language that can generate probabilities of a series of words. Whereas Large Language Models (LLM) are similar probabilistic AI models trained on extensive data for understanding and generating text. They have billions of parameters and long Context Lengths, and are used for various NLP tasks. The Context Length of an LLM is the number of tokens it considers when processing or generating text. Tokens, in turn, are units of text, which might correspond to a word or a sub word. For example, in a sentence like "ChatGPT is great," the tokens are "Chat", "GPT," "is," and "great." Transformers, featuring attention mechanisms are the underlying architecture of these LLMs [7]. Transformers comprise two main components: an Encoder and a Decoder. The Encoder is responsible for handling input text, while the Decoder generates new text based on the encoded information. LLMs can take different forms: they can be Encoder-only models, exemplified by BERT [8], feature an Encoder-Decoder architecture like T5 [9], or be Decoder-only models, such as GPT [10]. GPT functions as a decoder only model in text completion, embodying an auto-regressive or causal LLM. In contrast, T5 is an encoder-decoder model, incorporating both an encoder for processing input text and a decoder for output text generation. In recent years, LLM's like GPT and T5 has revolutionised the world by making Natural Language Generation task easy, robust, and accessible to anyone. Throughout this paper, we use the terms GPT-chat for the models GPT-3.5-turbo and GPT-4, distinguishing them from other GPT models, which are referred to as text-completion models.

C. Zero-shot, Few-shot and Fine-tuning approaches for text generation

Zero-shot learning involves training a model to perform a task without prior exposure to examples of that task. Few-shot learning introduced by Tom *et al.* [11] extends this by allowing the model to be *trained in-context* on a small number of examples for a specific task. *Fine-tuning* entails further training of a pre-existing model on a particular task to enhance its performance on that task. Marius *et al.* [6] studies these approaches in detail. Some important points, as described below should be considered while choosing the right approach.

Zero-shot and few-shot procedures, as defined, do not require training, in contrast to fine-tuning which explicitly demands it. The training process is expensive as it involves substantial investments in both hardware infrastructure and expertise. Additionally, zero-shot prompting operates without dependence on task-specific data, while few-shot prompting employs a modest number of task-specific examples. On the other hand, fine-tuning necessitates a considerable volume of labeled task-specific data for the training process. As a result, zero-shot and few-shot approaches demonstrate practical advantages, and the recommendation is to consider fine-tuning only if these approaches prove ineffective. Theoretically, a few-shot approach is supposed to generate better results than 0-shot prompt, whereas, fine-tuned models are supposed to generate even better response in the domain it's fine-tuned.

Nevertheless, the high-performing models designed for zero-shot and few-shot scenarios are typically huge generic LLMs boasting billions to trillions of parameters. Hosting such models locally can be challenging and, in many cases, impossible, leading to their reliance on cloud infrastructure accessible over the internet. This reliance on external services can give rise to privacy and legal concerns, particularly when private or proprietary data needs to be transmitted to third-party service providers. Conversely, smaller models tailored

to specific tasks can achieve comparable results without the need for extensive resources. These more compact fine-tuned models can be deployed on a company's servers, mitigating some of the privacy and legal considerations associated with utilizing cloud services. Our study aims to explore all these approaches with the different kind of generative LLM's mentioned above.

III. RELATED WORK

For several decades, the significance of Architecture Knowledge Management (AKM) has been widely recognized, and various tools have been developed to assist AKM. [1] The focus on capturing and documenting Architecture Design Decisions (ADD) has grown with the recognition of ADDs' significance [12] [13]. Notable efforts, like Arman *et al.*'s work on retrieving ADDs through the analysis of project history artifacts, have been made [14]. However, these tools were largely deterministic and labour intensive and did not capture much attention from architects. However some Machine Learning based tools have began to emerged in the recent years.

Machine learning for Software Engineering has been a prominent topic in the past few years [15] [16]. Generative AI is gaining popularity recently, particularly in tasks like automatic code documentation and summarization [1]. While April *et al.* worked on Code Documentation in Computational Notebooks [17], Jian *et al.* assembels multiples Foundational models for code summarization [18]. Niche works, such as Intent-Driven Comment Generation by Fangwen *et al.*, have explored specific aspects of this field [19].

Limited research exists in the realm of ML for Software Architecture or ADD. For instance, Xueying *et al.* explored the Identification of Design Decisions from Mailing List using ML and NLP techniques predating the era of LLMs [20]. In a similar vein, Manoj *et al.* focused on extracting Design Decisions from natural language documentation, like Jira tickets [21]. It's noteworthy, however, that neither work did not specifically address ADRs and did not involve the generation of Design Decisions.

Despite these advancements, the field of Software Architecture remains relatively untouched by Generative AI or LLMs. In the context of ADR, Machine Learning applications have been limited, and to the best of our knowledge no works generating Design Decisions using LLMs exists.

IV. STUDY DESIGN AND EXECUTION

In this section, we document the empirical experiment executed for this study in terms of goal (Section IV-A), research questions (Section IV-B), study subject (Section IV-C), experimental procedure (Section IV-D), and experimental Metrics (Section IV-E). A visualization of the process is provided in Figure 1.

A. Goal

This research is about an exploratory investigation to determine the feasibility of effectively utilizing LLMs for the generation of Architectural Design Decisions based on a provided context. The study aims to identify both the advantages and disadvantages associated with this approach. More formally, by utilizing the Goal-Question-Metric approach [22], this objective can be described as follows:

Analyze the effectiveness of Large Language Models
For the purpose of generating Architectural Design Decisions
With respect to contexts of Architecture Decision Records
From the viewpoint of Software Architects
In the context of using Generative AI for Architectural Knowledge Management

B. Research Question

In order to achieve our goal, we address the following three research questions (RQ):

RQ1 *Can LLMs be successfully employed to generate architectural design decisions from a given context in a zero-shot setting?*

Answering this introductory research question will help us determine if an architect can directly utilize a generic, foundational LLM, simply by presenting it with a decision Context, to obtain the corresponding Design Decision.

RQ2 *Does few-shot approach affect or improve a LLM's ability to generate Design Decisions?*

With RQ_2 we try to determine if LLMs can learn ADRs, *in-context* and subsequently generate improved Design Decisions, by incorporating sample Context-Decision pairs into the prompt. We investigate whether smaller and more cost-effective LLMs can effectively generate design decisions in few-approach at par more powerful models in 0-shot setting.

RQ3 *Does Fine-tuning LLMs enhances it's capability of generating architectural Design Decisions based on a provided context?*

While research question RQ_1 and RQ_2 focuses on using foundational LLMs off the shelf, in RQ_3 we investigate if fine-tuning an LLM with Context-Decision pairs improves it's capability of generating Design Decision from Context. Huge LLMs capable of generating excellent text in zero and few-shot are mostly available as cloud service and cannot be trained or hosted loaclly. In RQ_3 we investigate if smaller models that can be trained and hosted locally can generate Design Decisions at par the extensive models after fine-tuning.

C. Experimental Subject

1) ADR Data: We found several repositories from the internet, mostly from github. After reviewing several options, we chose five repositories with a substantial number of sample ADRs that adhered mostly to a standard format [2]. We got 17 ADRs from archane-framework[3], 17 from winery[4], 32 from
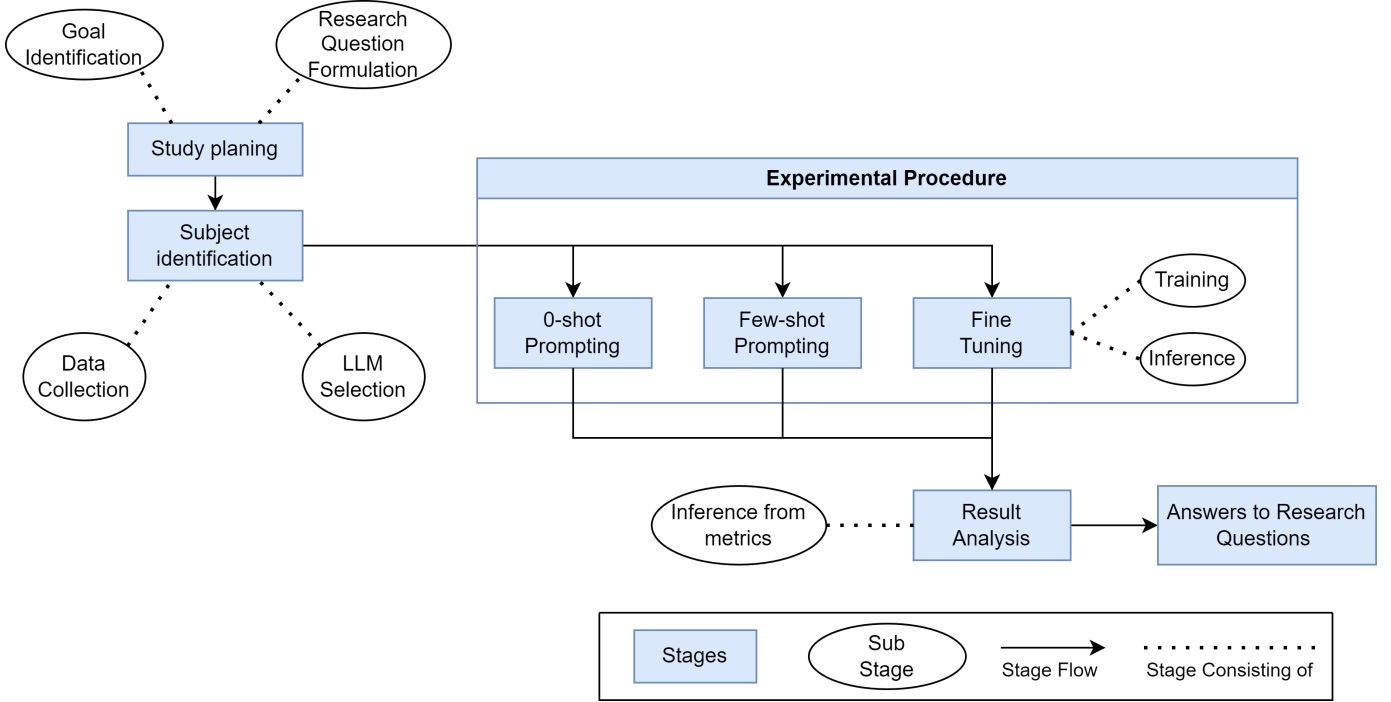

Fig. 1. Study Design

| **Context** |
| We need to decide on whether to use Python as a programming language for our project. Our project involves data analysis, machine learning, and web development. |
| **Decision** |
| We have decided to use Python as our primary programming language for our project. |

Fig. 2. Sample ADR after Extracting Context-Decision

joelparkerhenderson[5] repository, 14 from cardano[6], and 15 from island[7]. We gathered a total of 95 ADRs from these sources or GitHub repositories, we were able to download it directly. However, for other sources, we had to perform web crawling on the master repository to retrieve the corresponding ADRs. To facilitate our experiments, we focused on extracting the Context and Decision components. This extraction process involved a combination of regular expressions and manual intervention. A sample ADR data after extracting Context-Decision is given in Figure 2.

2) LLM: To have a comprehensive representation of LLM's, we selected models from two well-known model series, GPT by OpenAI (representing decoder-only models) and T5 by Google (representing Encoder-Decoder models).

TABLE I
LLMs USED IN THIS STUDY

family	model	size	context length	availability
GPT-2	GPT2 GPT2-medium GPT2-large GPT2-xl	124M 335M 774M 1.5B	1024	local
GPT-3	ada davinci	350M 175B	2048	api
GPT-3.5	text-davinci-003 GPT-3.5-turbo	175B 175B	4000	api
GPT-4	GPT-4	T+	8192	api
T5	T5-small T5-base T5-large T5-3b	60M 223M 738M 3B	infinite	local
T0	T0-3b	3B	infinite	local
Flan-T5	Flan-T5-small Flan-T5-base Flan-T5-large Flan-T5-xl	77M 248M 783M 3B	infinite	local

The models were chosen keeping in mind model sizes, data they were trained on, training techniques, and interaction modes.

Table II presents the essential characteristics of the models employed in this study. "Size" and "Context Length" are explained in Section II. The "Availability" category indicates whether the model can be hosted on a local PC or server, or if an API call is required to access the model from the service

provider.

It is crucial to acknowledge that ADRs are extensive texts, often comprising thousands of characters. However, transformer-based models like GPTs have constraints on context length. Consequently, we selectively utilized data that falls below the context length for each model (refer Table II), ensuring compatibility. Notably, T5-based models do not face this limitation, as T5 can handle infinite context lengths owing to its relative positional encoding. Nevertheless, owing to hardware constraints, we found it necessary to establish a threshold even for T5 models. Additionally, during fine tuning, there was a necessity to exclude data beyond a specific threshold due to hardware constraints.

D. Experimental Procedure

1) 0-shot approach: Here we provided the model with the decision *Context* along with a relevant prompt, anticipating it to produce the desired *Design Decision*. We experimented with several prompt variations on a subset of samples. For instance, in one prompt, we solely provided the *Context* without additional information, while in another, we included the phrase *Architectural Decision Record* within the prompt. Through manual experimentation with various prompts on a subset of samples, we identified the most effective prompts. Some noteworthy observations made in this phase are given in Section VI. For GPT text-completion models, and T5 based models the most effective prompt is given in Figure 3 and for GPT-chat models, and the most effective prompt is given in Figure 4.

Architectural Decision Record
Context:
{context}
Decision:

Fig. 3. 0-shot prompt for GPT text-completion models, and T5 based models

---
{ "role": "system",
"content": "This is an Architectural Decision Record for a software. Give a ## Decision corresponding to the ## Context provided by the User " }
{ "role": "user",
"content": {context} }

Fig. 4. 0-shot prompt for GPT-chat models

2) few-shot approach: In this approach, we engaged in *in-context learning* by manually selecting two ADRs as gold samples. These gold samples were selected by one of the authors who played the role of an expert Software Architect. These gold samples, along with the context of the desired sample, were provided to the model. The expectation was for the model to learn in-context from the gold samples and predict the desired Decision based on the Context from the given sample.

Here also we performed manual experimentation with various prompts on a subset of samples and identified the most effective prompts. For GPT text-completion models, and T5 based models the most effective prompt is given in Figure 5 and for GPT-chat models, and the most effective prompt is given in Figure 6.

Context:
{gold context 1}
Decision:
{gold decision 1}
Context:
{gold context 2}
Decision:
{gold decision 2}
Context:
{context}
Decision:

Fig. 5. few-shot prompt for GPT text-completion models, and T5 based models

---
{ "role": "system",
"content": "These are architecture decision records. Follow the examples to get return Decision based on Context provided by the User" }
{ "role": "user",
"content": "## Context {gold context 1}" }
{ "role": "assistant",
"content": "## Decision {gold decision 1}" }
{ "role": "user",
"content": "## Context {gold context 2}" }
{ "role": "assistant",
"content": "## Decision {gold decision 2}" }
{ "role": "user",
"content": "## Context {context}"

Fig. 6. few-shot prompt for GPT-chat models

3) fine-tuning approach: For fine-tuning we used GPT2 and four T5 based models. GPT2 is a text completion model. Hence for GPT2, we concatenated each context with the corresponding decision and trained the model on generating the full text. However T5 is an encoder-decoder model. Hence the Context was provided to the Encoder, and the Decoder was supposed to generate the Decision. The model was trained on generating the correct Decision.

During training we saved a checkpoint per epoch, and used the checkpoint with least validation loss for inference. The validation loss / epoch curves are given in Figure 7. During inference, we passed the Decision Context to the models and expected it to return the desired Design Decision.

E. Metrics

In accordance with NLP literature, the evaluation of text generation often relies on a combination of metrics rather than a single standard measure. In line with this practice, our
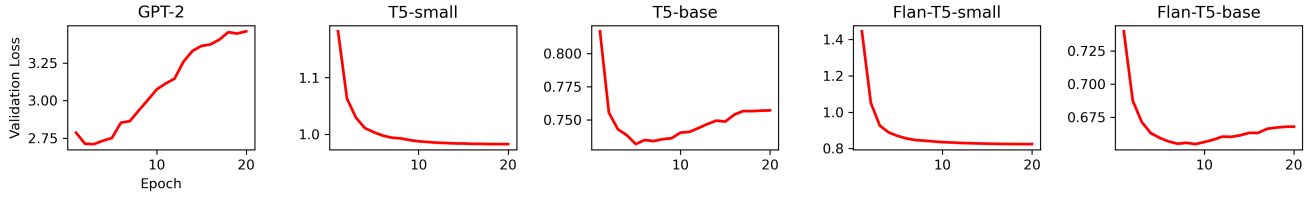
Fig. 7. Training: validation loss / epoch

> **GPT2**
> Generate: {context} {decision}
>
> **T5**
> Read: {context}
> Generate: {decision}

Fig. 8. Training Format

evaluation incorporates ROUGE-1, BLEU score, METEOR, and BERT Score.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [23] is a set of metrics used to evaluate the quality of machine-generated summaries. Here we are using ROUGE-1 that measures the overlap of unigrams (individual words) between the system-generated text and the reference text.

The BLEU (Bilingual Evaluation Understudy) [24] score is a metric used to evaluate the quality of machine-translated text.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) [25] is a metric used for evaluating machine-generated text, particularly in the context of machine translation.

BERT Score [26] is an automatic evaluation metric used to assess the quality of text generation. It leverages pre-trained contextual embeddings from BERT (Bidirectional Encoder Representations from Transformers) [8] and measures the similarity between words in candidate and reference sentences using cosine similarity. BERT Score has been shown to correlate well with human judgment in evaluating text generation outputs. Hence we use it as the primary metric in this study.

Our experimental scripts and details can be found in: https://anonymous.4open.science/r/LLMforADR-401C/README.md

V. RESULTS

In this section, we report the results of our empirical experimentation according to the research questions guiding this study (see Section IV).

Tables IV, V and VI present the metric values for the 0-shot, few-shot, and fine-tuning experiments, respectively, with the highest value for each metric emphasized in bold. The BERT score (F1), which is our primary metric, is graphically represented in Figure 9. Although there are instances where the metrics do not align, the result tables show that they generally

work in tandem and show a correlation with the BERT score (F1).

In addition to computing evaluation metrics, we manually reviewed a subset of the generated samples, focusing on both content and formatting of the generated Decisions. Some sample generated Decision are given in Figure 10, following the same sample from Figure 2.

A. Results RQ_1: Can LLMs be successfully employed to generate architectural design decisions from a given context in a zero-shot setting?

In this introductory research question, we determine whether an architect can effectively employ an LLM by providing it with a decision Context to derive the corresponding Design Decision.

While experimenting with various prompts to find the most effective prompt, we observed including the term 'Architectural Decision Record' in the prompt enhances the overall performance in 0-shot prompting for all models. However, this improvement pertains for adhering to the ADR markdown format rather than impacting the content of the Design Decision the model is required to produce. We also find including the *Title* (e.g., 'MySQL database') of the ADR does not yield notable improvements in the results for 0-shot prompting across all the models. We've also noticed that larger and more descriptive contexts lead to better and more comprehensive decisions.

From Table III and Figure 9 it is evident that the large GPT models and T0 performs well in 0-shot approach, while the T5 models show mediocre results, and the smaller GPT models performs the worst. GPT-4 scores 0.849 as Bert Score (F1) stands out to be the best model with best performance in all metrics other than Blue score and Bert Score (precision).

Upon manually inspecting the generated content, it becomes evident that larger GPT models consistently produce pertinent responses in the correct ADR format, with GPT-4's formatting approaching human level. T0 also generates relevant responses but lacks proper formatting. Conversely, the remaining models demonstrate suboptimal performance in both generating accurate decisions and formatting them correctly, with smaller GPT models frequently producing hallucinatory content.

B. Results RQ_2: Does few-shot approach affect or improve a LLM's ability to generate Design Decisions?

In RQ_2 we aim to assess the capability of LLMs in generating Design Decisons in few-shot approach. Our goal is

TABLE II
RESULTS 0-SHOT

model	rouge-1	blue	meteor	bert score			support
				precision	recall	f1	
GPT2	0.071	0.008	0.109	0.740	0.808	0.772	78
GPT2-medium	0.071	0.007	0.103	0.734	0.81	0.769	78
GPT2-large	0.081	0.009	0.115	0.737	0.815	0.773	78
GPT2-xl	0.084	0.01	0.114	0.749	0.822	0.783	78
GPT3-ada	0.129	0.005	0.137	0.79	0.823	0.805	89
GPT3-davinci	0.154	0.009	0.167	0.799	0.837	0.817	89
GPT3.5-text-davinci-003	0.242	**0.031**	0.198	0.846	0.849	0.847	91
GPT-3.5-turbo	0.235	0.029	0.213	0.836	0.848	0.841	92
GPT-4	**0.259**	0.028	**0.219**	0.847	**0.851**	**0.849**	95
T5-small	0.203	0.017	0.168	0.84	0.831	0.835	91
T5-base	0.164	0.011	0.142	0.829	0.823	0.825	91
T5-large	0.16	0.014	0.137	0.816	0.823	0.819	91
T5-3b	0.088	0.004	0.077	0.794	0.81	0.801	91
t0-3b	0.187	0.005	0.122	**0.856**	0.823	0.839	91
Flan-T5-small	0.142	0.008	0.101	0.823	0.813	0.817	91
Flan-T5-base	0.158	0.012	0.115	0.845	0.822	0.833	91
Flan-T5-large	0.166	0.014	0.105	0.836	0.822	0.827	91
Flan-T5-xl	0.147	0.011	0.089	0.837	0.816	0.825	91

TABLE III
RESULTS FEW-SHOT

model	rouge-1	blue	meteor	bert score			support
				precision	recall	f1	
GPT2	0.087	0.012	0.135	0.778	0.841	0.807	13
GPT2-medium	0.11	0.016	0.017	0.785	0.837	0.809	13
GPT2-large	0.117	0.018	0.176	0.79	0.851	0.818	13
GPT2-xl	0.122	0.018	0.18	0.794	**0.857**	0.823	13
GPT3-ada	0.14	0.008	0.158	0.789	0.827	0.807	86
GPT3-davinci	0.171	0.011	0.181	0.802	0.84	0.82	86
GPT3.5-text-davinci-003	**0.245**	0.028	0.207	**0.849**	0.851	**0.849**	91
GPT-3.5-turbo	0.226	0.027	**0.219**	0.832	0.85	0.84	92
GPT-4	0.23	**0.03**	0.21	0.836	0.851	0.843	93
T5-small	0.151	0.007	0.121	0.824	0.819	0.821	91
T5-base	0.146	0.008	0.12	0.805	0.821	0.812	91
T5-large	0.177	0.014	0.146	0.811	0.825	0.817	91
T5-3b	0.136	0.006	0.1	0.805	0.821	0.812	91
t0-3b	0.152	0.011	0.133	0.808	0.82	0.813	91
Flan-T5-small	0.162	0.005	0.152	0.782	0.824	0.801	91
Flan-T5-base	0.176	0.011	0.158	0.811	0.829	0.819	91
Flan-T5-large	0.168	0.014	0.16	0.81	0.828	0.818	91
Flan-T5-xl	0.17	0.011	0.155	0.805	0.825	0.814	91

to access if smaller, more economical LLMs can outperform larger models in zero-shot scenarios by integrating sample Context-Decision pairs into the prompts.

Based on the data presented in Table IV and the observations from Figure 9, it is apparent that the larger GPT models excel when employing a few-shot approach, whereas the performance of other models remains moderate. Notably, GPT-3.5's text-davinci-003 stands out as the top performer in this approach, attaining the highest scores in rouge-1, BERT-score (precision), and BERT-score (F1). Specifically, it achieves a BERT-score (F1) of 0.849, which matches the performance of GPT-4 in the 0-shot setting.

Upon manually examining the generated content, we observe that the quality of generation closely resembles that of the 0-shot approach, wherein larger GPT models consistently produce relevant Design Decisions with appropriate formatting, in contrast to other models that exhibit shortcomings in this regard. Notably, smaller GPT models stand out as an exception. Although these smaller models frequently generated hallucinatory content in the 0-shot approach, there is a significant reduction in hallucinations when employing the few-shot approach. The rise in metric values, as illustrated in Figure 9, further attests to this observation.

C. Results RQ_3: Does Fine-tuning LLMs enhances it's capability of generating architectural Design Decisions based on a provided context?

RQ_3 examines whether fine-tuning LLMs with Context-Decision pairs improves their ability to generate Design Decisions. We explore if smaller locally trainable and deployable

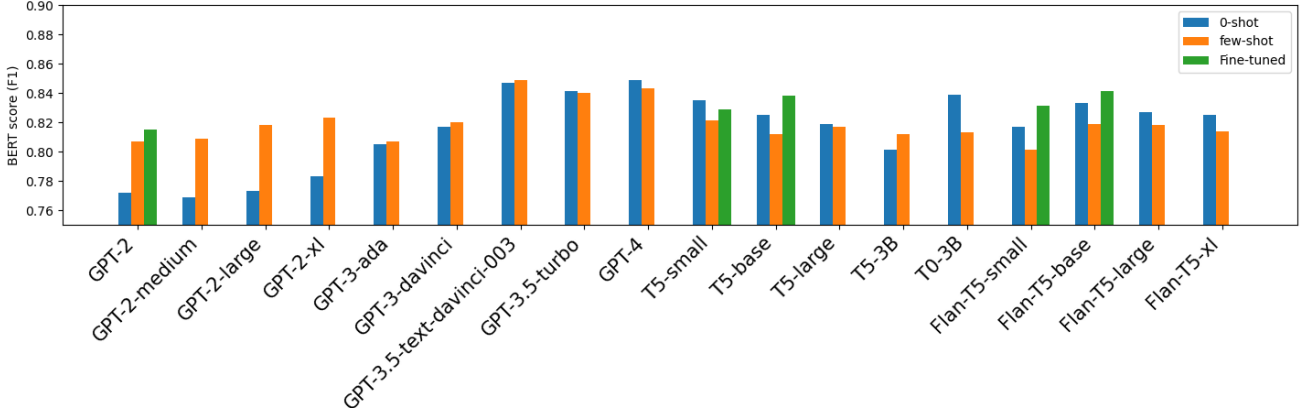| model | rouge-1 | blue | meteor | bert score | | | Train | support |
				precision	recall	f1		Test
GPT2	0.155	0.01	0.176	0.797	0.834	0.815	62	16
T5-small	0.199	0.008	0.156	0.835	0.824	0.829	73	19
T5-base	0.195	0.022	0.168	0.839	0.838	0.838	71	18
Flan-T5-small	0.19	0.018	0.139	0.842	0.822	0.831	73	19
Flan-T5-base	**0.231**	**0.028**	**0.171**	**0.842**	**0.841**	**0.841**	71	18

Fig. 9. BERT score (F1)

fine-tuned models can match the capability of extensive cloud-based models in generating comparable Design Decisions.

Table V and Figure 9 illustrate that all the fine-tuned models outperform their off-the-shelf counterparts, with T5 being the only exception. Flan-T5 emerges as the leading fine-tuned model across all metrics, achieving an impressive score of 0.841 for BERT-score (F1).

On manual examination of the generated content, we find that the T5 models stop hallucinating after fine-tuning. Additionally, the decision content also improves after fine-tuning.

VI. DISCUSSION

This exploratory study provides empirical evidence that LLM's can be leveraged to Generate Architecture Design Decision given the Context, and would definitely help Software Architects in generating ADR's. Below, we address each research question by examining the outcomes of our experiments. Several conclusions are drawn regarding the efficiency of using LLMs to generate ADRs.

A. Can LLMs be successfully employed to generate architectural design decisions from a given context in a zero-shot setting? (RQ_1)

Yes, LLMs can be used to generate Design Decisions in zero-shot setting, but not entirely autonomously. Even the best performing models fails to generate the required Decision comprehensively, necessitating human involvement.

Notably, smaller models such as GPT-2, T5, and Flan-T5, originally not designed for zero-shot capabilities, exhibit sub-par performance. T0, despite being a relatively smaller model, returns relevant decisions but lacks adherence to the ADR format. Contrastingly, smaller GPT models performs better in adhering to correct ADR format than producing correct response. GPT-4 achieves a standard comparable to human performance in ADR format but falls short in generating the required Decision comprehensively. Both manual observation and evaluation metrics support these findings, necessitating further research on leveraging LLMs to reach human-level proficiency in generating Design Decisions.

It's important to highlight that both GPT-3.5 and GPT-4 are cloud-based and offered as 'Software as a Service.' Consequently, to use these models, data must be transmitted to the service provider, like OpenAI[8], potentially raising concerns related to privacy, intellectual security, or legal implications.

Hence we recommend that if security is not a concern, LLMs like GPT-3.5 and GPT-4 can be successfully employed to generate Design Decisions, but not autonomously. Rather it should be used to assist Architects in documenting and making Design Decisions.

B. Does few-shot approach affect or improve a LLM's ability to generate Design Decisions? (RQ_2)

The impact of the few-shot approach compared to the 0-shot approach is inconclusive. As depicted in Figure 9,

[8]https://platform.openai.com/docs/models

Actual Decision
We have decided to use Python as our primary programming language for our project.

Decision Generated by GPT-4 in 0-shot
We will use Python as our programming language for this project. Python has emerged as a reliable, versatile language with a rich ecosystem of libraries and frameworks that cater to our specific needs including data analysis (Pandas, NumPy), machine learning (scikit-learn, TensorFlow), and web development (Django, Flask). Its simplicity and readability make for quicker turnaround times in development phases. Therefore, Python fits perfectly with the requirements of our project, providing us with all the tools necessary for successful development and deployment.

Decision Generated by text-davinci-003 in few-shot
After considering the various criteria discussed, we have decided to go ahead and use Python for our project.
Python is a versatile, powerful language with a wide range of features, libraries, and tools that make it ideal for multiple tasks, including data analysis, machine learning, and web development. It is simple to learn and use, and there is a large community of developers who can provide support and resources. Python is also open-source, free to use, and has a great number of packages available for various tasks.
Python is also a great choice in terms of performance and scalability. With the right packages, it can be used for big data processing and high performance applications. Additionally, Python is an interpreted language, which makes it easier to debug and maintain than compiled languages.
Overall, Python is a great choice for our project, given its features and capabilities, scalability, and community support.

Decision Generated by Flan-T5-small after fine-tuning
It will have a large dataset, which will display a lot of data. This can help us in preparing the application. We need to choose Python as a programming language, because all of our projects will require Python to use it.

Fig. 10. Sample Generated Decision

while the few-shot approach enhances performance for GPT-2 models, in certain models such as GPT-3, GPT-3.5 and GPT-4 the effect is insignificant. Interestingly, for some models like the entire T5 series (excluding T5-3B), the performance actually decreases. This observation is consistent across other metrics. Manual observations suggests that neither adherence to the ADR format nor the content of the generated Decision improves significantly with the few-shot approach for any of the models. The only exception is GPT-2, where the few-shot

Main findings RQ_1 (Generating Design Decisions in 0-shot approach): While LLMs can generate Design Decisions using a 0-shot approach, the quality does not reach human-level. Nonetheless, this method can be employed to support architects in documenting and formulating Design Decisions.

approach reduces hallucinations.

Based on the metrics, it can be inferred that the performance of GPT-3.5's text-davinci-003 using the few-shot approach is equal to that of GPT-4 using the 0-shot approach, both of which exhibit the best metrics in our study. However, it's important to note that GPT-4 is significantly larger (10 times) and expensive (1.5 times) compared text-davinci-003.

Hence we infer from this study that one may use smaller and cheaper models like GPT-3.5's text-davinci-003 in a few shot approach and generate Design Decisions of quality at par with powerful models like GPT-4 at a cheaper rate.

Main findings RQ_2 (Impact of few-shot approach in Generating Design Decisions): While the performance of certain LLMs may show improvement in a few-shot setting, the overall phenomenon lacks generalization and remains inconclusive. Nevertheless, smaller LLMs, when presented with a few-shot prompt, can be employed as substitutes for larger LLMs in certain scenarios.

C. Does Fine-tuning LLMs enhances it's capability of generating architectural Design Decisions based on a provided context? (RQ_3)

It is conclusive from the metrics that fine-tuning does improve the Design Decisions generating capability of LLMs, with T5-small being the only exception. Manual observation also suggests the same both with respect to formatting and content. Though it's observed that even the best performing fine-tuned model can't match the performance of GPT-4 in terms of following ADR markdown format.

The metrics show that after fine-tuning, the top-performing model, Flan-T5-base, with 248 million parameters, achieves results comparable with GPT-3.5 in a few-shot approach, which has 175 billion parameters (700 times more). Additionally, fine-tuned Flan-T5-base also produces results comparable to GPT-4 in a 0-shot approach, boasting over a trillion parameters (7000 times more). This clearly demonstrates that smaller fine-tuned models can effectively substitute larger models for generating design decisions. Due to hardware limitations, our fine-tuning efforts were capped at models with fewer than 400 million parameters. The results suggest that T5-3B or Flan-T5-xl, each with 3 billion parameters (30 times less than GPT-3.5), could potentially match GPT-4 performance through fine-tuning. This indicates that a more extensive study with increased data and hardware capabilities is needed which may yield more promising outcomes.

It is crucial to emphasize that smaller models, such as Flan-T5, can be hosted in-house in most corporate environments, thereby mitigating potential privacy, security, or legal concerns.

Hence, with respect to RQ_3 we conclude, while smaller fine-tuned models like Flan-T5 may not reach the performance level of giants like GPT-4 in generating design decisions, they serve as valuable alternatives in scenarios where privacy or security considerations are paramount.

> **Main findings RQ_3 (Impact of Fine-tuning in Generating Design Decisions):** Fine-tuned LLMs exhibit improved capability in generating design decisions. Compact fine-tuned models, which require minimal infrastructure for hosting, demonstrate results comparable with extensive LLMs and can be utilized as their substitutes in specific scenarios.

VII. THREATS TO VALIDITY

In this section, we discuss threats to the validity of our study, following the categorization provided by Wholin et al. [27].

A. Internal Validity

A threat to internal validity may arise from the selection of metrics, given that evaluating text generation is a challenging and unresolved problem. To mitigate this threat, we have addressed it by adopting a set of metrics commonly utilized by the NLP community. These metrics aim to capture the quality of generated text from various perspectives.

An additional threat to internal validity could be the Inconsistency of ADR writing style, which might lead to inaccurate evaluations. Different organizations or individuals employ different styles for writing ADRs. For example, while some may adopt a precise writing approach, others may opt for a descriptive manner. So even if the LLM generates a Design Decision that matches with the manually written one, the evaluation metrics score may decrease if the wordings, or style doesn't match. To mitigate this threat, we have introduces a set of metrics, which captures the quality of generated text from different perspectives. Additionally, we have introduced the BERT score, which evaluates based on the semantic meaning of words rather than a strict word-to-word comparison, aiming to mitigate the impact of stylistic variations.

B. Construct Validity

A threat to construct validity is the limited amount of data used for training while addressing RQ_3. This threat remains partially mitigated due to unavailability of open source ADR repositories on the internet. Despite this challenge, we have gathered 95 ADRs from 5 different repositories. While fine-tuning, we used 80% of the data for training, and rest 20% for testing. It should be mentioned that we used the same data for validation and testing for the fine-tuning experiments as explained in Section IV due to the low amount of data available. This can be considered a limitation to this study, though it's unlikely that it could change the conclusions.

C. External Validity

A potential threat to the external validity of our study arises from the selection of LLMs. Given the availability of numerous LLMs from various organizations, it is impractical to include all in our study. Thus, we must carefully choose LLMs that are indicative of the broader landscape. To address this concern, we opted for two widely recognized model series: GPT by OpenAI, representing decoder-only models, and T5 by Google, representing Encoder-Decoder models. Within these series, we have incorporated models of varying sizes, training data, training techniques, interaction styles, and other pertinent differences to increase the diversity of our chosen models. For instance, we varied the model sizes from small to extra-large, for T5 and Flan-T5 models. GPT-3 and GPT-3.5 differ in both training data and methods. Additionally, GPT offers text-completion and chat models, both included in this study.

An additional concern regarding External Validity is the potential that the gathered ADRs for this study may not be representative of ADRs in a broader context. To address this concern, we conducted thorough searches for ADRs on the web and also explored GitHub to identify open-source ADR repositories. We selected as many repositories as possible, ensuring adherence to ADR format and rules, and sought diversity in terms of style and source. To the best of our knowledge, the collected ADRs are representative of ADRs in general.

VIII. CONCLUSION AND FUTURE WORK

This study explores the potential of leveraging LLMs, specifically GPT and T5-based models, to automate the generation of ADRs, which are a crucial part of AKM. The investigation involves using 0-shot, few-shot, and fine-tuning approaches to generate ADR Decisions based on their respective Contexts. The study reveals that LLMs demonstrate the capability to generate ADD. Models like GPT-3.5, GPT-4, and T0 effectively generate relevant Design Decisions adhering to ADR markdown format with both 0-shot and few-shot prompting. Fine-tuning enhances the performance of all models significantly. GPT-4 excels in 0-shot prompting, while even smaller models like text-davinci-003 yield similar results using a few-shot approach. Smaller models such as Flan-T5-base demonstrate comparable results after fine-tuning. In summary, LLMs may not be entirely dependable for ADR generation, but they can effectively assist architects in documenting and formulating Design Decisions. Smaller fine-tuned models, requiring minimal hosting, can be locally employed for Decision generation, especially in privacy or security-sensitive scenarios.

Future work involves fine-tuning large LLMs to enhance Design Decision generation, aiming for human-level performance. Additionally, in vivo experiments with architects in live projects will assess the effectiveness of auto-generating ADRs for widespread adoption. Recognizing challenges in fully capturing Design Decisions from decision Context, future efforts will focus on integrating contextual information from diverse sources, including other ADRs, AKM components like

design diagrams, and the codebase, to improve the generation process.

REFERENCES

[1] Tang, Antony & Avgeriou, Paris & Jansen, Anton & Capilla, Rafael & Ali Babar, Muhammad. (2013). A comparative study of architecture knowledge management tools. Journal of Systems and Software. 352-370. 10.1016/j.jss.2009.08.032.

[2] Weinreich, Rainer & Groher, Iris. (2016). Software architecture knowledge management approaches and their support for knowledge management activities: A systematic literature review. Information and Software Technology. 80. 10.1016/j.infsof.2016.09.007.

[3] Capilla, Rafael & Jansen, Anton & Tang, Antony & Avgeriou, Paris & Ali Babar, Muhammad. (2016). 10 years of Software Architecture Knowledge Management: Practice and Future. Journal of Systems and Software. 10.1016/j.jss.2015.08.054.

[4] Jansen, Anton & Bosch, Jan. (2005). Software Architecture as a Set of Architectural Design Decisions. Proceedings - 5th Working IEEE/IFIP Conference on Software Architecture, WICSA 2005. 2005. 109-120. 10.1109/WICSA.2005.61.

[5] Buchgeher, Georg & Schöberl, Stefan & Geist, Verena & Dorninger, Bernhard & Haindl, Philipp & Weinreich, Rainer. (2023). Using Architecture Decision Records in Open Source Projects – An MSR Study on GitHub. IEEE Access. 11. 63725 - 63740. 10.1109/AC-CESS.2023.3287654.

[6] Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation. In Findings of the Association for Computational Linguistics: ACL 2023, pages 12284–12314, Toronto, Canada. Association for Computational Linguistics.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[8] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[9] Raffel, Colin, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." J. Mach. Learn. Res. 21 (2019): 140:1-140:67.

[10] Radford, Alec & Narasimhan, Karthik & Salimans, Tim & Sutskever, Ilya. "Improving language understanding by generative pre-training." (2018):

[11] Brown, Tom & Mann, Benjamin & Ryder, Nick & Subbiah, Melanie & Kaplan, Jared & Dhariwal, Prafulla & Neelakantan, Arvind & Shyam, Pranav & Sastry, Girish & Askell, Amanda & Agarwal, Sandhini & Herbert-Voss, Ariel & Krueger, Gretchen & Henighan, Tom & Child, Rewon & Ramesh, Aditya & Ziegler, Daniel & Wu, Jeffrey & Winter, Clemens & Amodei, Dario. (2020). Language Models are Few-Shot Learners.

[12] Shahin, Mojtaba & Liang, Peng & Khayyambashi, Mohammad. (2009). Architectural Design Decision: Existing Models and Tools. 293-296. 10.1109/WICSA.2009.5290823.

[13] Scheerer, Max & Busch, Axel & Koziolek, Anne. (2017). Automatic evaluation of complex design decisions in component-based software architectures. 67-76. 10.1145/3127041.3127059.

[14] Shahbazian, Arman & Lee, Youn & Le, Duc & Brun, Yuriy & Medvidovic, Nenad. (2018). Recovering Architectural Design Decisions. 95-9509. 10.1109/ICSA.2018.00019.

[15] Shafiq, Saad & Mashkoor, Atif & Dorn, Christoph & Egyed, Alexander. (2020). Machine Learning for Software Engineering: A Systematic Mapping.

[16] Meinke, Karl & Bennaceur, Amel. (2017). Machine Learning for Software Engineering Models, Methods, and Applications. 10.1145/3183440.3183461.

[17] Wang, April & Wang, Dakuo & Drozdal, Jaimie & Muller, Michael & Park, Soya & Weisz, Justin & Liu, Xuye & Wu, Lingfei & Dugan, Casey. (2022). Documentation Matters: Human-Centered AI System to Assist Data Science Code Documentation in Computational Notebooks. ACM Transactions on Computer-Human Interaction. 29. 1-33. 10.1145/3489465.

[18] Gu, Jian & Salza, Pasquale & Gall, Harald. (2022). Assemble Foundation Models for Automatic Code Summarization. 935-946. 10.1109/SANER53432.2022.00112.

[19] Mu, Fangwen & Chen, Xiao & Shi, Lin & Wang, Song & Wang, Qing. (2023). Developer-Intent Driven Code Comment Generation.

[20] Li, Xueying & Liang, Peng & Li, Zengyang. (2020). Automatic Identification of Decisions from the Hibernate Developer Mailing List. 10.1145/3383219.3383225.

[21] Bhat, Manoj & Tinnes, Christof & Shumaiev, Klym & Biesdorf, Andreas & Hohenstein, Uwe & Matthes, Florian. (2019). ADeX: A Tool for Automatic Curation of Design Decision Knowledge for Architectural Decision Recommendations. 10.1109/ICSA-C.2019.00035.

[22] V. R. Basili, G. Caldiera, and D. Rombach, "The Goal Question Metric Approach," in Encyclopedia of Software Engineering. Wiley, 1994, pp. 528–532.

[23] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[24] Papineni, Kishore & Roukos, Salim & Ward, Todd & Zhu, Wei Jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135.

[25] Lavie, Alon & Agarwal, Abhaya. (2007). METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. 228-231.

[26] Zhang, Tianyi & Kishore, Varsha & Wu, Felix & Weinberger, Kilian & Artzi, Yoav. (2019). BERTScore: Evaluating Text Generation with BERT.

[27] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, Experimentation in software engineering. Springer Science & Business Media, 2012