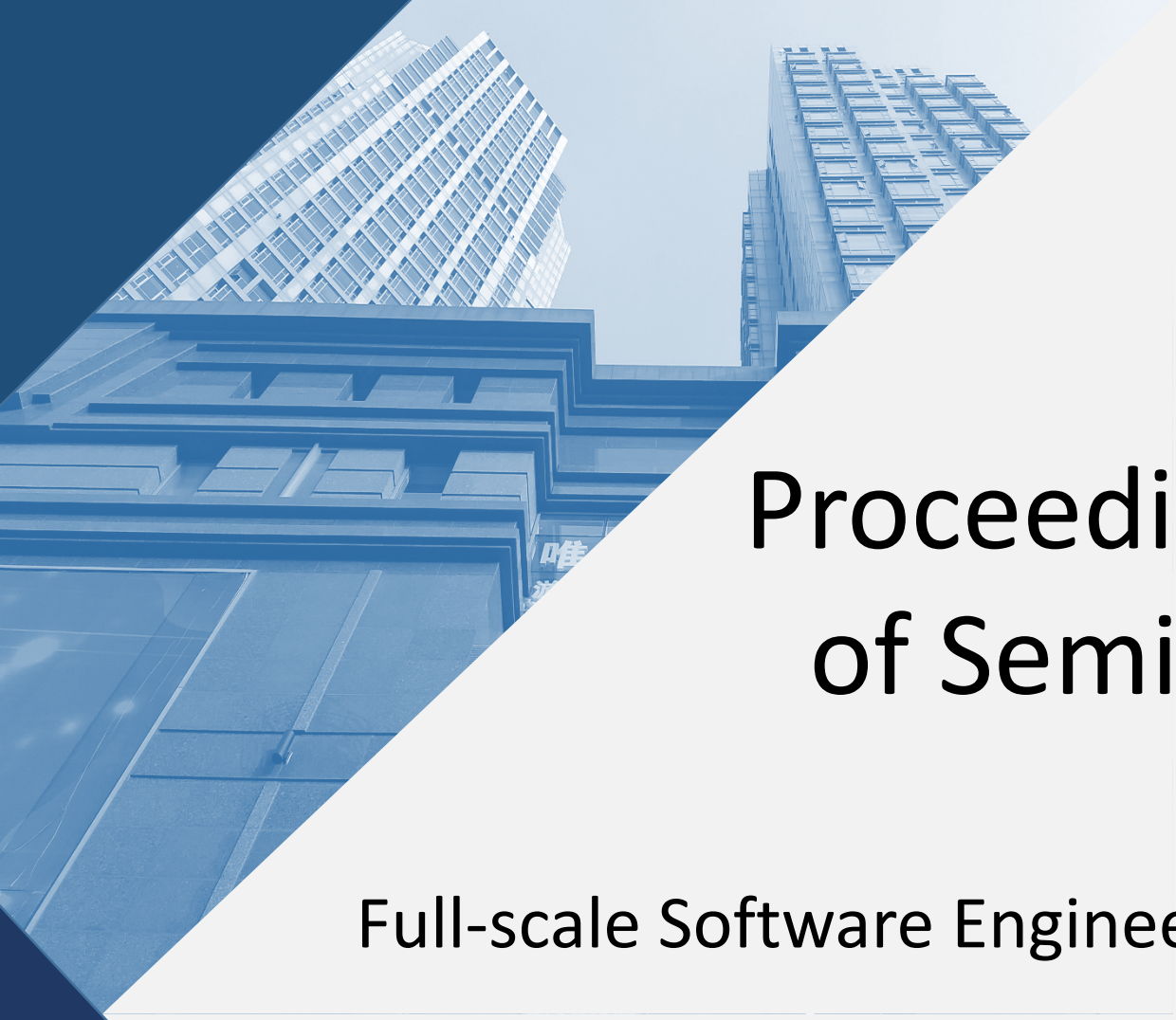


Proceedings of Seminar

Full-scale Software Engineering

2025

Editors: Horst Lichter
Selin Coban
Alex Mattukat
Ada Slupczynski

SWC Software Construction | RWTH AACHEN UNIVERSITY

Table of Contents

Creating an Extensive Overview of Authorization Methods - Insights from an LLM-Assisted Systematic Literature Review

Markus Fleischmann
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
markus.fleischmann@rwth-aachen.de

Marius Kaufmann
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
marius.kaufmann@rwth-aachen.de

ABSTRACT

In today's world, authorization is the central IT security mechanism, for which countless solutions have been developed. Due to the hard-to-grasp diversity of solutions and the absence of a clear compendium for them, it is difficult to get an overview of the zoo of authorization methods. To develop such a compendium, a Systematic Literature Review (SLR) is a suitable methodology, but it comes with the significant drawback of being very time-consuming. Recent advances in Large Language Models (LLMs) offer potential for speeding up this process, yet their effectiveness and precision in key SLR tasks—such as search query formulation, abstract screening, and data extraction—remain underexplored. This study applies a (shortened) LLM-Assisted SLR workflow to develop a broad overview of existing authorization methods. Furthermore, due to missing empirical evidence of the method applied, this study evaluates the applicability of the method to gain insights into the scientific feasibility and performance of LLM-assisted SLR approaches. Therefore, this study makes two important contributions. The results show that while LLMs can speed up the SLR process in some stages, it can pose significant problems in others, both in terms of efficiency and scientific rigor. The resulting overview of authorization methods provides an approachable catalog of state-of-the-art authorization methods.

Keywords

Large Language Models, Research Methodology, Authorization, Access Control, Systematic Literature Review

1. INTRODUCTION

Authorization is a fundamental component of today's IT landscape, ensuring that authenticated users and systems can access only those resources for which they have appropriate permissions. As with many areas in IT, there is no single best solution for implementing authorization. Instead, a diverse set of models and approaches exists, tailored to varying requirements and contexts.

Numerous authorization methods have been proposed and analyzed in the literature. To systematically consolidate these findings from prior studies, a Systematic Literature Review (SLR) provides a rigorous framework for synthesizing existing research. Traditionally, SLRs were first used in medical research, with the goal of supporting evidence-based medicine [5]. The potential of SLRs was also discovered by researchers in other disciplines, and they are now widely used in Software Engineering [5]. During an SLR, at each step of the research synthesis, the process relies on explicit, systematic methods that aim to reduce bias. During the selection of studies, the appraisal of their quality, and when summarizing the selected studies, objective standards are the foundation of any decision taken in an SLR [6]. However, conducting an SLR is highly time- and resource intensive, requiring, for instance, the construction of the initial meta-query and the detailed evaluation of potentially hundreds of sources.

In recent years, the emergence of Large Language Models (LLMs) has sparked interest in their application for research support. One area under active discussion is their role in assisting with SLRs - specifically, how they may reduce the workload while maintaining methodological rigor [11]. While some emphasize the strengths of LLMs in this context [10], others point out their risks and drawbacks when applied in SLRs [12].

Given the variety of authorization methods presented in previous work, there is no comprehensive and up-to-date overview listing and summarizing these models. To contribute to filling this gap, this paper seeks to answer two primary research questions (RQs). First, we aim to provide a comprehensive overview of state-of-the-art authorization methods that can be used as a first reference. Second, we assess, through a case study, how LLMs can assist in an SLR by evaluating the proposed method.

- **RQ1:** What authorization methods are presented in the state-of-the-art literature?
- **RQ2:** How can LLMs support the SLR process?

Based on this, we aim to contribute in two ways to the scientific community. First, our SLR of existing authorization methods and the resulting overview serves as an initial reference for future studies and supports the selection of

SWC Seminar 2025 RWTH Aachen University, Germany.

suitable authorization methods for new projects. Second, through our case study, we provide a practical example of how to integrate LLMs into SLRs and evaluate the strengths, weaknesses, and limitations of LLM usage, offering valuable insights for further work in the area of LLM-assisted SLRs.

2. RELATED WORK

2.1 SLR methodology

2.1.1 Traditional SLR methodology

According to Tingelhoff et al. [11], the process of conducting an SLR consists of four main steps: During the *Design* phase, the researchers familiarize themselves with the domain, identify the need for their SLR in their respective field of research, and define its scope. In the *Discover* phase, researchers define a search meta-query and conduct searches in scientific databases to retrieve relevant literature. This set of papers is then extended by conducting both a forward- and backward search. The third step of an SLR is the *Develop* phase, where the previously discovered papers are evaluated. Then, data from the relevant papers is extracted, with the goal of identifying patterns and gaps in the current research. Lastly, in the *Disseminate* phase, the findings of the SLR and the decisions that were made that led to the findings are presented transparently. [11]

2.1.2 LLM-assisted SLR methodology

As outlined in the introduction, due to the increasing amount of published scientific papers, and the rapid development of LLMs in the last years, a potential solution for the challenge researchers face in handling these large data volumes is to implement LLMs in their SLR process [11]. While this equips them with powerful tools that can streamline their research synthesis, critics expressed concern that the usage of LLMs could increase bias and undermine scientific integrity. To benefit from the advantages without risking an increased bias, Tingelhoff et al. introduce a framework on where and how to use LLMs in the SLR process responsibly [11]. These guidelines are outlined in the following:

1. *Design*: In the first phase of conducting an SLR, LLMs can be used to significantly speed up the domain familiarization process, where the researcher needs to scan through extensive amounts of literature in the designated area. It should, however, be kept in mind, that potential biases of the used LLM, as well as oversimplification of the screened content through the LLM can undermine scientific rigor and integrity. Likewise, LLMs can be used to aid in identifying the need for an SLR, with similar risks and considerations. Finally, these tools can assist the researcher prepare a proposal, although in this phase, the use should be strictly limited to technical assistance (e.g. generating complicated LaTeX code) and refining the linguistic quality of the proposal. [11]

2. *Discover*: During the paper search phase, LLMs offer a way to improve search efficiency significantly, by helping to create and refine a meta-query which is used to find relevant papers in a scientific database, and potentially translating it between the query syntax of different scientific databases. Due to its strength in natural language processing, it is particularly useful in identifying synonyms and related terms, which can improve the quality of the meta-query. In the backward and forward search, connections between papers can be visualized using programmatic tools. LLMs, however, should only be used cautiously, as the models potential bias can have especially strong consequences during this phase, as it could heavily affect the overall direction the SLR takes. [11]

3. *Develop*: It is highly debated whether LLMs can and should be used in the quality assessment of found papers, or whether the potentially induced bias poses too high of a risk to the scientific soundness of the SLR. Given this uncertainty, Tingelhoff et al. [11] recommend not using LLM tools in this step. In the data extraction step, however, the structured nature of using an LLM with the same query to extract data from all papers prevents unequal representation between studies, and therefore even presents a potential advantage over traditional methods. At the same time, the risks for oversimplification and bias have to be taken into account, as well as the intransparent nature of the underlying LLM algorithms and the continuously updated models, which pose a threat to the replicability of the study. Hence, the advantages and risks have to be evaluated cautiously by the researcher. Last but not least, LLMs excel in synthesizing and structuring data. They therefore offer a powerful tool to be used in the data synthesis phase of the SLR, which is mainly due to their ability to process and synthesize extensive data volumes, which by far exceeds human capabilities. However, the researcher should not over-rely on LLMs, as the models tendency to recycle and repeat existing narratives can prevent innovative contributions. [11]

4. *Disseminate*: During the entire SLR process, all decisions should be reported. This especially includes any use of LLMs, which is vital for transparently documenting the research process. In the final step of presenting the results, usage of LLMs to generate content imposes fundamental threats to the scientific soundness of the research, and is therefore strongly discouraged. Researchers are, however, encouraged to make use of LLMs potential to refine writing style and improve language structure, which can increase the impact of the results. [11]

We use these guidelines as a basis for deciding when and how to use LLMs in our shortened SLR.

2.2 Authorization

While previous SLRs have summarized the state-of-the-art of authorization methods, they are now either outdated or focus on a limited selection of methods, lacking a comprehensive overview.

Mohamed et al. [7] propose a classification of access control models into five categories, as well as a distinction between the ambiguously used terms “authorization strategies“ [7] and “access control models“ [7]. According to their distinction, the former describes the *view point* of describing the authorization policies, while the latter defines the *enforcement* of said authorization policies [7]. Focusing solely on the subclass of risk-based access control models, Atlam et al. [2] performed an SLR, in which they, however, introduce a general definition and, to a certain extend, an abstraction of the term “access control“ [2]. They break down access control models into five fundamental elements: subjects, objects, actions, privileges, and access policies [2]. Setting a comparable focus on IoT devices, Qiu et al. [9] give a general overview over access control models and technologies, both classic and recent [9]. Covering a different, but similarly topical field, Ahmad et al. [1] evaluate the state of

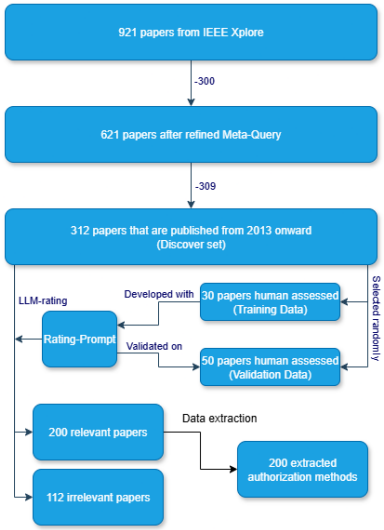


Figure 1: Process Model

the art of access control models used in social networks, and provide a classification of said models based on eight axes [1].

3. METHODOLOGY

We follow the four-step SLR process of Tingelhoff et al. [11], but apply a shortened SLR to fit our scope and resources. In the *Design* step, domain familiarization is conducted via an informal ChatGPT-powered survey of core authorization concepts and keyword derivation without a formal protocol. In the *Discover* phase, backward and forward snowballing are omitted, relying solely on an IEEE Xplore meta-query refined via ChatGPT and supplemented by random title and abstract checks. In the *Develop* step, we extract only four predefined fields (method name, abbreviation, usage, brief summary) for each included paper, omitting a full quality appraisal. We therefore created a preliminary categorization of the found authorization methods.

Figure 1 illustrates our process model, showing how we move from the initial IEEE Xplore result set through abstract screening and classification (*Develop*) to the final extraction of methods.

3.1 Design

Given limited resources of the study, domain familiarity was established through a preliminary study of core authorization principles and techniques. For this preliminary study, the LLM model ChatGPT-4 was utilized to obtain a quick overview over the current authorization landscape. The interaction with ChatGPT-4 was informal and unstructured, relying on spontaneous prompts and exploratory queries rather than a predefined search strategy. Additionally, the DeepResearch functionality of OpenAI was tested experimentally to find and summarize relevant literature. This foundational understanding enabled an informed evaluation and classification of literature identified in subsequent phases.

The primary motivation for conducting the SLR is the lack of a comprehensive and up-to-date catalog of existing authorization methods. Because the time and resources of this study are limited, the scope of the search is restricted to the IEEE Xplore database. IEEE Xplore's subject coverage is estimated at roughly 70% engineering and 22% Computer Science, making it one of the most specialized, high-coverage databases in our field [4]. Focusing solely on IEEE Xplore ensures our shortened SLR captures the vast majority of relevant literature within our resource constraints. The SLR aims to identify authorization methods presented in previous studies; to ensure that the papers are centrally concerned with authorization concepts, the term "Authorization" (or the British spelling "Authorisation", which was suggested by the LLM) must either appear in the title or as a keyword. Including the British spelling yielded an additional nine papers that would otherwise have been missed. To only include papers that present a new method - rather than pure implementations or evaluations of existing methods - the abstract must contain at least one of the following words: "model", "method", "framework" or "scheme". These terms serve as an indicator of a conceptual or technical novelty. "Model" and "method" denote a systematic or structured approach to an authorization related problem. "Framework" and "scheme" signal the introduction of a reusable solution or generalized structure. Requiring the presence of at least one of those words increases the likelihood that a paper introduces a new contribution rather than solely evaluating, reviewing, or implementing an existing approach.

3.2 Discover

The identified keywords from the *Design* phase are used to create the search "meta"-query used for the IEEE-Xplore database. To evaluate the current search query, a series of random title and abstract screenings were carried out on the current result set. A number of papers that were screened described an authorization method evaluation rather than a new method itself. To refine the query, ChatGPT was used to suggest improvements.

To enhance the results of this usage of ChatGPT, prompt patterns based on the work of White et al. [13] were utilized. The 'Persona' pattern is useful when the LLM output should take a certain point of view or perspective. This is important, as "users often may not know what types of outputs or details are important for an LLM to focus on to achieve a given task. They may know, however, the role or type of person that they would normally ask to help with these things" [13]. This is achieved by directly prompting the LLM to act as a given role or take a certain perspective for the rest of the conversation.

The intent of the 'Template' Pattern, on the other hand, is to "ensure an LLMs output follows a precise template in terms of structure" [13]. This can be utilized by providing the template to the model in the prompt, together with the statement that the output must preserve the exact structure of the template. This is especially useful when requiring the output to be in a machine-readable format such as JSON.

'Question Refinement' "helps bridge the gap between the user's knowledge and the LLM's understanding, thereby yielding more efficient and accurate interactions" [13]. This technique involves prompting the LLM to suggest improved or specific questions that the user could ask in place of the original query [13].

As a general procedure for developing the meta query with the help of an LLM, the 'Persona' pattern was used. We formulated our prompt with this pattern, prompting ChatGPT-4 to act as if it is a research assistant for a SLR

that is currently conducted, and that the results of the discover phase must be restricted to a more relevant subset. This resulted in the following additional keywords (searched for in full text or metadata): "access control", "privilege management", "identity management", "access management" or "permission management". The complete query is available in the published dataset [3].

Backward and forward citation searches were not applied due to the limited resources of the study. Computer security and computer science in general is a fast evolving field, meaning older studies risk becoming obsolete [8]. Restricting our review to papers published from 2013 onward ensures that we capture the most recent authorization-methods while keeping the study requirements manageable.

The refined query produced a set of 312 papers. This set is henceforth referred to as the Discover set.

3.3 Develop

3.3.1 Quality assessment of paper

While Tingelhoff et al. [11] advice against using LLMs in the Develop phase to assess the quality and relevance of discovered papers, Schulhoff et al. [10] demonstrated that LLM-based screening can be effective. They integrated an LLM-based screening step into their PRISMA pipeline, using GPT-4 to classify abstracts against explicit inclusion/exclusion criteria. Validation on a set of 100 manually annotated papers yielded 89% precision and 75% recall [10]. To evaluate LLM support (RQ2), we used ChatGPT-o3 to classify papers as either relevant or irrelevant. Following Schulhoff et al.'s screening workflow [10], we defined the following inclusion and exclusion criteria to optimize our ChatGPT-o3 prompt in identifying papers that introduce novel authorization methods:

1. Include a paper if it proposes, analyzes or formally evaluates a new authorization decision or enforcement idea
2. Include a paper if the main contribution is
   - a decision model that decides whether an identified subject may access a resource
   - an enforcement/evaluation mechanism that applies existing rules or policies; or
   - an architecture, protocol or languages that formalizes authorization logic.
3. Exclude a paper if its main focus is
   - authentication or identity proof,
   - cryptography, privacy preservation, or secure communication only,
   - trust/risk/score computation that never feeds a concrete access decision,
   - organizational governance, legal compliance, or policy authoring guidance without enforcement detail,
   - ML model training whose sole outcome is a better model (not an authorization component),
   - implementation frameworks that merely compose standard RBAC/ABAC/XACML components without proposing a new model or algorithm,
   - case studies that only deploy an existing mechanism without modification, or
   - secondary studies (as these require a dedicated treatment, which exceeds the resources available for this study).

These criteria were refined through an iterative review of 30 randomly selected papers from the Discover set. In order to create this criteria, we started to identify recurring characteristics in papers that were clearly relevant or irrelevant for our research. Disagreements were discussed to reach a consensus and the underlying reasoning behind inclusion or exclusion was formalized into explicit rules.

The prompt to classify the papers was, hence, tested and developed using these 30 papers as training data. The basic structure of the prompt was taken from Schulhoff et al. [10]. In order to improve the prompt, the 'Persona' pattern was used again as well as the 'Question Refinement' pattern. We employed a zero-shot prompting strategy, as tests with few-shot prompting resulted in highly biased classifications: the LLM tended to overfit to the specific examples provided, treating them as templates rather than generalizable guidance. The full prompt is archived in the published dataset [3].

For validation of the prompt, another random sample of 50 abstracts was screened manually by both researchers and by ChatGPT-o3.

Papers labeled as "uncertain" by the model were subsequently classified as irrelevant, based on manual validation. In the sample of 50 abstracts, none of the "uncertain"-labeled papers introduced new authorization methods, suggesting that the LLM tends to assign the "uncertain" label to irrelevant papers. Applying the final prompt to all records yielded 200 papers that were classified as relevant.

3.3.2 Extract data from papers

Since the focus of RQ1 is to gather information about the authorization methods that are introduced in the discovered papers, we developed a prompt that extracts a summary comprising of four elements about a given paper:

1. the name of the introduced authorization method
2. its abbreviation (if given)
3. its usage (where/how the method is intended to be deployed)
4. a brief summary of less than 250 words, outlining the methods architecture, its key components and its operational flow

Together, these fields give a reader everything needed at a glance: the method's identity (name + abbreviation), its deployment scope (usage), and its core mechanism (concise summary).

Following White et al.'s prompt-engineering catalogue [13], we combined the 'Persona' and 'Template' patterns.

First, a persona statement (*"You are an AI assistant conducting a systematic literature review on authorization (authZ) methods."*) casts ChatGPT-4o as a domain-aware reviewer, steering it to focus solely on novel authorization contributions that actually introduce a new method.

Second, a rigid JSON skeleton—an instantiation of the Template pattern—forces every answer to appear as one

JSON object wrapped in a single-element array with fixed key order and empty-string fall-backs. This guarantees machine-readable, uniform output that can be concatenated and parsed without post-processing. The full extraction prompt can be found in the published dataset [3].

Tingelhoff et al. [11] state that LLM tools have the potential to assist the researcher in the extraction step, but that it is necessary to thoroughly consider the imposed risks such as oversimplification of complex matters and potentially undermining the scientific integrity of the analysis. Therefore, they advise to rigorously check the accuracy and depth of the data extracted by these tools to ensure it accurately represents the subject's detailed nuances [11].

We therefore manually extracted the authorization method and a concise summary from 10 randomly selected papers, and then compared these results with LLM-generated summaries. Because the LLM's output matched manual summaries in its overall content with acceptable accuracy, we used ChatGPT-4o to process the papers.

In contrast to the caution advised by Tingelhoff et al. [11] regarding LLM-generated content, we assembled the extracted method summaries into a structured catalog via a lightweight, LLM-assisted workflow because taxonomy creation was not the primary focus of this paper. First, ChatGPT-o4-mini-high proposed a fixed set of categories based on all summaries, which a researcher then reviewed and approved. Next, the same model automatically assigned each method summary to one of these approved categories. To drive both steps, our prompts employed the 'Persona' and 'Template' patterns and were executed in a zero-shot fashion to avoid the additional time costs of crafting few-shot examples. Finally, to validate the classification, a random sample of 40 methods was manually checked, yielding an acceptance rate of 92.5%. This resulted in 13 different categories.

4. RESULT

4.1 Evaluation of LLM-assisted SLR

4.1.1 Statistical evaluation of LLM-assisted abstract screening

The validation of the abstract screening and classification of papers from the Discover set to relevant/irrelevant was performed on a set of 50 randomly selected papers from said Discover set. The abstracts of the selected papers were screened by both researchers and the LLM (outlined in Section 3.3.1). The validation yielded the following results. Note that, since a classification as "uncertain" by the LLM was interpreted as an "irrelevant" as mentioned in Section 3.3.1, these cases were also counted as such in the validation.

- The two researchers agree in the classification of 46/50 papers, equaling an agreement of 92%. The 4 papers on which the researchers agree were discarded for the rest of the validation, since their manual classification cannot be compared to the LLM's classification in a meaningful way.
- For the 46 papers on which the researchers agree, their classification matches the LLMs classification in 39/46 papers, corresponding to an accuracy of $\frac{39}{46} \approxeq 84.78\%$.
- In case the researchers classified the paper as relevant, the LLM also classified it as relevant in 28/29 cases, which is a true positive rate (TPR), also known as sensitivity, of TPR $= \frac{28}{29} \approxeq 96.55\%$.
- In case the researchers classified the paper as irrelevant, the LLM also classified it as irrelevant in 11/17 cases, which is a true negative rate (TNR), also known as specificity, of TNR $= \frac{11}{17} \approxeq 64.71\%$.
- Since the number of true positives (TP) is 28, and the number of false positives (FP) is 6, the precision is $\frac{\text{TP}}{\text{TP+FP}} = \frac{28}{28+6} \approxeq 82.35\%$.

4.1.2 Time reduction of LLM-assisted abstract screening

In the abstract screening process, it is necessary to develop objective criteria, which determines whether an abstract classifies a paper as relevant or irrelevant [11]. Since this is necessary regardless of whether an LLM assisted approach is used, this does not make a difference.

Assuming that it takes a researcher a minute on average to read and classify an abstract as relevant or irrelevant for the study, in our case, reading every of the 312 abstracts would have taken us 312 minutes per person, or 624 minutes in total in case both researchers read all abstracts.

Developing, testing and improving the prompt used to classify an abstract as relevant or irrelevant took approximately 360 minutes. Additionally, the prompt had to be validated, for which both researchers manually read and classified 50 abstracts, taking around 50 minutes each, adding up to 100 minutes. Last but not least, all 312 abstracts have to be evaluated by the LLM using the developed prompt. This took around 30 minutes, as they had to be split up into batches of processable size. In total, the LLM-assisted abstract screening therefore took 490 minutes.

Comparing these estimated times, the LLM-assisted abstract screening in our case study led to a speed-up of $\frac{624}{490} - 1 \approxeq 27.35\%$.

4.1.3 Time reduction of LLM-assisted data extraction

In the initial paper screening, it took both researchers 15 to 20 minutes on average to extract and summarize the presented method. Considering this duration, this would result in around 4000 minutes in total to summarize the 200 relevant papers manually. It took about 60 minutes to create and afterwards verify the prompt using the created summaries. The papers were uploaded manually one by one in order to create the summary, as it was not possible to automate the task due to API limitations of the available ChatGPT subscription. This took a little less than 30 seconds of work for each paper to perform the extraction via ChatGPT and save the result. This accumulates to a required time of approximately 96 minutes. The manual extraction of the 10 papers for validation took around 170 min. The usage of the LLM-assisted approach therefore took $60 + 96 + 170 = 326$ mins and thus resulted in a time saving of 3674 minutes. This is a speedup of $\frac{4000}{362} - 1 \approxeq 1004.97\%$ compared to a manual extraction.

4.2 Categorization of Authorization Methods

The preliminary categorization yielded 13 different categories. The different categories with the their respective count are listed in Table 1.

Category	#
Domain-Specific Network and IoT Authorization	29
Blockchain and Decentralized Ledger Authorization	28
Attribute-Based Access Control and Encryption	24
Cryptographic and Formal Methods	20
Service and Microservice Authorization Frameworks	19
Role and Relationship-Based Access Control	17
Trust and Risk-Aware Authorization	14
Privacy-Preserving Authorization	13
Context-Aware and Adaptive Authorization	10
Continuous and Dynamic Authorization	9
Delegation and Multi-Party Access Control	8
ML-Enhanced Authorization	5
Distributed Query Enforcement Models	4

Table 1: Distribution of the 200 authorization methods across categories (# denotes the count of methods in each category).

The following provides a short description for each category:

- **Blockchain and Decentralized Ledger Authorization**: This category includes methods that leverage blockchain or distributed ledger technologies to decentralize trust, ensure tamper-evident logging, and manage authorization decisions via smart contracts.
- **Domain-Specific Network and IoT Authorization**: This category includes methods tailored to specific network domains or IoT environments, such as 5G, automotive CAN-FD, V2G, or edge-cloud integrations.
- **Attribute-Based Access Control and Encryption**: This category encompasses methods that use attribute-based access control models or attribute-based encryption schemes (e.g., CP-ABE) to define fine-grained policies based on user, resource, and contextual attributes.
- **Cryptographic and Formal Methods**: This category encompasses methods grounded in cryptographic key management techniques and formal policy analysis or verification to ensure secure and correct authorization.
- **Service and Microservice Authorization Frameworks**: This category includes methods embedding authorization within service workflows or microservice architectures, including policy sidecars and unified middleware approaches.
- **Role and Relationship-Based Access Control**: This category covers extensions to role-based access control and relationship-based models that introduce hierarchical roles, object relationships, or issuer trust relations to govern access.
- **Trust and Risk-Aware Authorization**: This category captures methods that compute trust scores, perform risk assessments, or adopt zero-trust principles to adapt authorization decisions dynamically based on trust evaluations.
- **Privacy-Preserving Authorization**: This category comprises methods designed to preserve user or data privacy during authorization, using techniques like stealth addresses, searchable encryption, or zero-knowledge proofs.
- **Context-Aware and Adaptive Authorization**: This category includes methods that integrate environmental or contextual parameters (e.g., time, location, device status) to adapt access decisions in real time.
- **Continuous and Dynamic Authorization**: This category covers methods that support continuous policy evaluation, dynamic negotiation, or runtime adaptation of access controls beyond static checks.
- **Delegation and Multi-Party Access Control**: This category comprises frameworks that enable delegation of rights and collaborative authorization decisions among multiple parties or controllers.
- **ML-Enhanced Authorization**: This category covers frameworks that leverage machine learning or data-driven models to infer, predict, or enforce authorization policies or detect anomalies.
- **Distributed Query Enforcement Models**: This category captures methods that enforce authorization at query execution in distributed or peer-to-peer systems by rewriting queries or integrating policy predicates into data requests.

5. DISCUSSION

We identified 200 primary studies from 2013-2025 and extracted the presented authorization methods with LLM support for the SLR process.

5.1 Insights on RQ1: the authorization catalog

Our LLM-assisted shortened SLR yielded a catalog of 200 authorization methods, organized into 13 categories (Table 1). It is important to stress, that our taxonomy was created via a rapid, zero-shot LLM workflow with and automated initial clustering and spot-checking for plausibility. Thus, the catalog represents a preliminary foundation rather than a definitive classification. As categories may overlap or split as domain definitions evolve over time, we view this catalog as a scaffold that future work should refine, validate and extend through deeper manual analysis, multi-database searches and domain expert consensus.

Even as a quick snapshot, the distribution of methods across categories already reveals where authorization research is most active. The two largest categories 'Domain-Specific Network and IoT Authorization' and 'Blockchain and Decentralized Ledger Authorization', with 29 and 28 methods, show the interest in tamper-evident, decentralized trust mechanisms and in securing emerging 5G/6G/IoT/edge-cloud environments.

5.2 Insights on RQ2: LLM-assisted SLR

5.2.1 Meta-query generation

Our results corroborate the findings of Tingelhoff et al. [11]: ChatGPT-assisted refinement of the meta-query helped to reduce the initial IEEE Xplore hit set from 921 to 621 (-32%), filtering out irrelevant literature. Although this shows the potential efficiency gain, every LLM-generated modification of the meta-query must be critically verified because errors introduced at this stage propagate throughout all subsequent stages of the SLR.

5.2.2 Abstract screening

The statistical evaluation presented in Section 4.1.1 shows an overall accuracy of 84.8% and a precision of 82.4%. However, the true–negative rate is only 64.7%, indicating bias: ChatGPT tended to label borderline papers as relevant. This can lead to an unnecessarily increased workload as the set for the later stages may be inflated. A possible reason for this bias is the distribution of relevant to irrelevant papers. Considering the 46 papers where both researchers agreed, they considered 29 as relevant and only 17 as irrelevant. This corresponds to a distribution of 63% to 37%. This imbalance existed as well in the 30 random papers that were used to create and refine the initial prompt. But as these papers were randomly selected from the set from the discovery phase, this imbalance potentially represents the general distribution of relevant to irrelevant papers in the set. Hence, no action was taken to remove the imbalance in either the initial set to refine nor the set to test the prompt.

In order to refine the prompt for the abstract screening, state-of-the-art research results regarding prompt engineering were applied to improve the prompt. When using an LLM-assisted approach for abstract screening, it is very important to carefully craft the prompt to achieve the best possible result. This requires additional knowledge regarding prompt-engineering. This can lead to complications if the SLR is conducted for a topic that is not related to prompt engineering, as the necessary knowledge to perform this task is an additional requirement to the competences of the researcher, in addition to the necessary domain knowledge.

Even as the results are promising with the high accuracy and precision, the poor true-negative rate, which is only slightly better than a random selection, as well as the additional competences required pose a problem for LLM usage to conduct abstract screening and assess the quality of papers. This corroborates the findings of Tingelhoff et al. [11], that LLMs should not be used to assess the quality of papers, as the risk to either miss critical papers or unnecessarily inflate the resulting set with irrelevant papers is too high, as these errors critically influence the overall result and efficiency of the SLR.

5.2.3 Data extraction

The approach to extract and summarize the presented authorization method from the remaining set of the assessment-phase was evaluated based on ten papers. These were summarized by both researchers and the result compared to the result of the LLM generated summary for each of those ten papers. This comparison yield the result, that the core points for each paper was present in the generated summaries. No oversimplification took place and nothing important was left out. Based on this, our result is that LLMs can be used well to support the researcher in the extraction phase, as they provide good results.

5.3 Limitations

Internal validity: We used different models of ChatGPT as our LLMs, which are both continuously updated and might not be available in the future. Therefore, re-runs may produce different screen results or summaries. We logged the model so the prompts support partial replication, yet a variation cannot be ruled out with this.

Construct validity: Only 50 abstracts were double-labeled by human reviewers, which are too few for reliable statistics. Future work should enlarge the gold-standard set for reliable statistics to further evaluate the usage of LLM assistance for abstract screening. In order to verify the extraction, only 10 papers were summarized by hand and compared to the result of the LLM generated summary. These are not enough papers to rule out the possibility that undetected extraction errors may persist in other parts of the created catalog. This is a consequence of limited resources, as this review is conducted as part of an educational event at the university with a limited time frame. The classification of authorization methods into categories was carried out in an unstructured way, complicating replication attempts.

External validity: Only a single database was used in the discover phase and the papers were restricted to english-language papers as well as to a time window of publications between 2013-2025. Authorization methods that are published elsewhere or in another language are therefore missed. Backward/forward snowballing was omitted to reduce the necessary time requirements, again due to the limited resources. Thus, the catalog should not be seen as exhaustive.

Practical limitations: In order to interact with LLM tools in the required frequency e.g. to process all abstracts it is necessary to obtain paid subscriptions. These are also necessary to get access to newer models that promise better results. We reached the daily limit for file uploads during the extraction phase as every full text must be forwarded to the model. This increases the chance that subsequent batches are processes by an already updated model after the limit is reset yielding different - and therefore inconsistent results. It may also slow down the research, as the researchers may have to wait for the "token limit reset" until the research can be continued.

6. CONCLUSIONS

This study conducted an LLM-assisted SLR addressing two research questions regarding authorization methods and LLM-assisted research synthesis. Our key conclusions are:

We developed an up-to-date catalog of summaries of 200 contemporary authorization methods (2013-2025), systematically extracted from IEEE Xplore. This resource provides researchers and practitioners with a structured reference for method selection, featuring method names, deployment contexts, and concise technical summaries. It addresses a critical gap in the field by consolidating fragmented literature into an actionable, although preliminary, taxonomy.

We designed and executed a case study to evaluate the integration of LLMs across three stages of a Systematic Literature Review on authorization methods: meta-query refinement using prompt engineering patterns, abstract screening for relevance assessment, and structured data extraction from primary studies. This methodological experiment systematically applied LLMs to core SLR tasks to assess their viability as research accelerants within a defined authorization domain.

Future efforts should treat our 13-category catalog as a starting point rather than a final taxonomy. Researchers can expand the catalog via multi-database searches with a broader time window for considered publications. Last but not least, mapping found authorization methods to the abstraction presented by Atlam et al. [2] could provide valuable insights, both evaluating its general applicability and establishing a common ground between publications in a research field densely populated with ambiguously used terms.

7. REFERENCES

[1] A. Ahmad and B. Whitworth. Access control taxonomy for social networks. In *2011 7th International Conference on Information Assurance and Security (IAS)*, pages 256–261. IEEE, 2011.

[2] H. F. Atlam, M. A. Azad, M. O. Alassafi, A. A. Alshdadi, and A. Alenezi. Risk-based access control model: A systematic literature review. *Future Internet*, 12(6):103, 2020.

[3] M. Fleischmann and M. Kaufmann. Creating an extensive overview of authorization methods – insights from an llm-assisted systematic literature review. `https://doi.org/10.5281/zenodo.15824023`, July 2025. Dataset.

[4] M. Gusenbauer. Search where you will find most: Comparing the disciplinary coverage of 56 bibliographic databases. *Scientometrics*, 127(5):2683–2745, 2022.

[5] S. Kitchenham, Keele et al. Guidelines for performing systematic literature reviews in software engineering. Technical report, 2007.

[6] G. Lame. Systematic literature reviews: An introduction. In *Proceedings of the design society: international conference on engineering design*, volume 1, pages 1633–1642. Cambridge University Press, 2019.

[7] A. Mohamed, D. Auer, D. Hofer, and J. Küng. Authorization strategies and classification of access control models. In *Future Data and Security Engineering: 8th International Conference, FDSE 2021, Virtual Event, November 24–26, 2021, Proceedings 8*, pages 155–174. Springer, 2021.

[8] B. M. Napoleão, F. Petrillo, and S. Hallé. Continuous systematic literature review: An approach for open science. *arXiv preprint arXiv:2108.12922*, 2021.

[9] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang. A survey on access control in the age of internet of things. *IEEE Internet of Things Journal*, 7(6):4682–4696, 2020.

[10] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, et al. The prompt report: a systematic survey of prompt engineering techniques. *Preprint at https://arxiv.org/abs/2406.06608*, 2024.

[11] F. Tingelhoff, M. Brugger, and J. M. Leimeister. A guide for structured literature reviews in business research: The state-of-the-art and how to integrate generative artificial intelligence. *Journal of Information Technology*, 40(1):77–99, 2025.

[12] G. Wagner, R. Lukyanenko, and G. Paré. Artificial intelligence and the conduct of literature reviews. *Journal of Information Technology*, 37(2):209–226, 2022.

[13] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

Enterprise Architecture Tools for Successful Modernization in Today's Technological Landscape

Johan Kakkallil Mathew
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
johan.mathew@rwth-aachen.de

Yiding Qu
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
yiding.qu@rwth-aachen.de

ABSTRACT

Legacy systems, while often critical to enterprise operations, pose a challenge to scalability, integration, and innovation across evolving business and technology landscapes. While modernization is essential, organizations face persistent issues such as technical debt, fragmented architectures, and operational inefficiencies. Enterprise Architecture (EA) tools have emerged as one of the enablers in this modernization journey, providing structured frameworks to align IT systems with evolving business goals. This paper analyzes and compares EA tools to evaluate their effectiveness in addressing key legacy system modernization challenges. It begins by outlining the specific modernization issues explored in this study—namely technical debt and operational inefficiencies in legacy enterprise systems. It then examines how selected EA tools could play a role in the transformation of legacy systems. A qualitative analysis is conducted using the Gartner EA Tool Framework. The framework was chosen for its structured approach and relevance to modern enterprise architecture practices. The comparison is structured around a set of modernization techniques such as cloud computing integration, AI support, DevOps enablement, and collaboration features—that are essential to legacy system modernization. The evaluation assesses the extent to which each tool supports these domains, highlighting their individual strengths and associated trade-offs. By synthesizing insights from current EA tools and their alignment with modern IT paradigms, the study highlights how these tools could assist in legacy system modernization. The paper concludes by reflecting on how EA tools address legacy system challenges and draws implications for aligning tool capabilities with modernization needs.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*productivity, architecture tools, architecture frameworks* ; D.2.9 [**Software Engineering**]: Management—*productivity, programming teams, software configuration management*; D.2.11 [**Software Engineering**]: Software Architectures—*Domain- specific architectures, Service-oriented architecture*

Keywords

legacy systems, legacy system modernization, systematic literature review, enterprise architecture, enterprise architecture framework, enterprise architecture management tools

SWC Seminar 2025 RWTH Aachen University, Germany.

1. INTRODUCTION

Legacy systems remain a central component in many organizations' software landscapes. These systems, often developed decades ago with technologies now considered obsolete, continue to perform critical tasks and encapsulate valuable domain logic [20, 9]. However, their aging architectures, monolithic design, and poor compatibility with contemporary platforms make them increasingly difficult to maintain and integrate within today's digital ecosystems[25]. As cloud-native infrastructure, microservices, and data-driven operations have become standard across industries, the limitations of legacy systems have grown more pronounced, threatening not only technical scalability but also long-term strategic agility[31]. Consequently, they struggle to accommodate the demands of today's digital enterprises, including real-time data access, cross-platform interoperability, and continuous delivery pipelines[23].

This has led to the development of modernization strategies aimed at evolving legacy systems thoughtfully. Modernization now encompasses a set of strategies aimed at evolving existing systems while preserving their essential components. These strategies include rehosting, refactoring, rearchitecting, replatforming, repurchasing, retiring and retaining[11, 12]. An ongoing challenge lies in the effective selection and integration of tools that support modernization efforts. These tools are critical for tasks such as architecture recovery, dependency analysis, service decomposition, data migration, testing, and system monitoring.

Given the complexity involved in modernizing legacy systems, having a clear, organized approach is essential. EA tools provide this by helping organizations visualize their entire IT landscape—including legacy components—and understand how different systems interact. This insight allows stakeholders to better plan modernization efforts, manage dependencies, and prioritize actions. By bringing structure and clarity to a complicated process, EA tools help reduce uncertainty and improve the chances of successful, smooth transitions to modern environments.

The paper is structured as follows: Section 2 provides background of legacy systems and enterprise modernization ,and reviews key areas of related research; Section 3 outlines the research methodology; Section 4 presents the result and Section 5 provides the discussion; and Section 6 concludes with practical implications and future research directions.

2. BACKGROUND AND RELATED WORK

Legacy systems often struggle to meet the agility, scalability, and integration demands of modern digital ecosystems due to outdated technologies and architectural complexity. Overcoming these challenges requires more than isolated upgrades—it demands a clear understanding of how systems interact, where inefficiencies lie, and how future changes will impact operations. EA provides this structured lens, enabling informed decision-making for modernization. Existing studies have explored EA's role in strategic IT alignment, portfolio rationalization, and transformation planning. Building on this foundation, this paper focuses on how EA tools are applied specifically in legacy modernization contexts, highlighting their capabilities, usage patterns, and contributions to effective transformation.

2.1 Legacy System

There is no universally accepted definition of a legacy system, but most literature characterizes it as a business-critical application that is technologically obsolete[20, 11, 12]. These systems often rely on deprecated programming languages, inflexible architectures, and monolithic structures that make them difficult to maintain or upgrade. Legacy systems are not simply old; they are often deeply embedded in organizational workflows and processes. As Khadka et al. notes, a key distinction between legacy systems and other outdated software lies in their continued importance to mission-critical tasks[20]. Many organizations still depend on them to run core operations, and their failure could result in significant business disruption[23, 16].

Despite their importance, legacy systems are often incompatible with modern platforms, making integration with cloud services, microservices, or APIs difficult[9, 25, 20]. Maintenance becomes increasingly expensive due to architectural complexity and the shrinking availability of professionals skilled in legacy languages such as COBOL[5]. As noted in[20], many organizations also face the risk of vendor lock-in and undocumented system logic, both of which further complicate modernization efforts. Full system replacement is risky, costly, and time-consuming. Moreover, many legacy systems contain irreplaceable business knowledge and historical data that are crucial for continuity[25, 28].

For these reasons, modernization—rather than replacement—has become the preferred strategy for most enterprises. This approach allows organizations to retain the core value of their legacy systems while improving their ability to interact with modern technologies and infrastructure[11]. M'Baya et al. highlights the importance of conducting detailed assessments of system architecture, business impact, and organizational readiness before conducting modernization[22].

2.2 Enterprise Modernization

Enterprise modernization refers to the strategic renewal of organizational software systems to better align with technological advancements and evolving business demands. Unlike routine maintenance, modernization involves more extensive changes—transforming legacy systems to be more agile, maintainable, and compatible with contemporary IT environments[31, 7, 28]. Modernization is not only driven by technical limitations but also by increasing demands for more interactive, user-friendly, and secure applications. Emerging technologies such as cloud computing, mobile platforms, and intelligent automation have raised the baseline for system capabilities, exposing the gap between what legacy systems can deliver and what modern users expect[25, 16]. However, modernization is rarely a purely technical process. It intersects with business processes, organizational culture, and resistance to change. Legacy systems are often tightly coupled with long-standing workflows, and even when new technologies are introduced, old practices and decision models may persist[7, 12]. This means that successful modernization requires not only technical interventions but also organizational alignment and stakeholder engagement[9, 5].

As a strategic discipline for managing complexity and guiding transformation, EA plays a role in modernizing IT to align with business goals. Organizations adopt EA when developing their IT to manage modernization initiatives. One of the aims of EA is to support the modernization of legacy systems by providing structured approaches to analyzing and evolving existing IT assets in alignment with business goals. It ensures that organizations can manage legacy systems' complexity while starting innovation and modernization projects to support long-term operational continuity [19].

2.3 Enterprise Architecture

EA is an important instrument to address transferring new information technology to practice. It is a coherent whole of principles, methods and models that are used in the design and realisation of the enterprise's organisational structure, business processes, information systems, and infrastructure[21]. The field of classical EA that has originally emerged from (and was ever since driven by) the need of organizations to be able to steer changes related to IT, has recently started to extend its scope with the physical domain, including for example manufacturing processes. More concretely, in the past the focus of EA models was on IT infrastructure, software applications, information flows, and business processes, designed at a rather high level of abstraction[6]. Implementing an EA in a software development company is a very important task, which can have a two-fold effect.

2.3.1 Enterprise Architecture Frameworks

With the adoption of EA by different types of organizations in different vertical industries, the selection of the most appropriate EA Framework (EAF) for that organization has become a critical decision when utilizing EA.

An EAF comprises a set of models, principles, and methods that are used to implement EA. The framework provides a means to communicate information about architectural artifacts, their relationships to each other, and to their stakeholders using a common vocabulary [29]. There are five major players commonly recognized in the field of EA frameworks: The Open Group Architecture Framework (TOGAF), the Zachman Framework, the Department of Defense Ar-

chitecture Framework (DoDAF), the Federal Enterprise Architecture Framework (FEAF), and the Gartner Enterprise Architecture Framework. Compared to other frameworks, the **Gartner Framework** approaches EA as a continuous and iterative process, beginning with an assessment of the current architectural state, followed by the definition of business-driven objectives for a desired future state, and the ongoing management of the transformation portfolio through measurable results.

2.3.2 Enterprise Architecture Tools

There are numerous EA tools on the market, that can be broadly categorized into three types that might seem oversimplified, but are useful for the purpose of this paper[18].

- **Diagramming tools**: These tools are used for creating diagrams and visual representations of EA. They include features for drawing flowcharts, UML diagrams, and other types of visual models. Examples include Microsoft Visio, Lucidchart, and Draw.io.

- **Modeling tools**: These tools are designed for creating and managing EA models. They support various modeling languages, such as ArchiMate, BPMN, and UML. Examples include Sparx Systems Enterprise Architect, BiZZdesign, and Orbus Software iServer.

- **Enterprise Architecture Management tools**: These tools provide a wide range of features for managing EA. They often include capabilities for modeling, diagramming, reporting, and collaboration. Examples include MEGA International, Avolution ABACUS, and Software AG ARIS.

3. METHODOLOGY

The study focuses on Gartner's Enterprise Architecture Framework, given Gartner's reputation for providing thorough, market-driven evaluations of EA tools. To ensure alignment with this framework and industry best practices, the selection of EA tools in this research is guided by the "Gartner Magic Quadrant for Enterprise Architecture Tools" (Gartner, 2024), which evaluates 16 leading tools based on their ability to execute and completeness of vision.

To complement the insights from the Magic Quadrant, we also reviewed the official websites of these tools, gathering supplementary information such as product features, and white papers. This multi-source perspective supports a more holistic evaluation, particularly valuable in the context of legacy system modernization, where strategic planning, complexity management, and business alignment are critical.

This approach is suitable for legacy system modernization by highlighting tools that support strategic planning, complexity management, and business alignment—essential elements for effective modernization efforts. This research adopts a qualitative, literature-based approach, drawing upon a range of secondary data sources including peer-reviewed articles, case studies, white papers, and grey literature.

3.1 Search Engines and Criteria

The key sources and repositories utilized also include Google Scholar, IEEE Xplore, ACM Digital Library, ResearchGate, and SpringerLink. These platforms provided access to a wide range of scholarly articles, technical papers, industry reports, and peer-reviewed publications essential for comprehensive research and analysis.

Keywords and search queries included: legacy systems, enterprise modernization, legacy system modernization, application modernization, brownfield projects, tools for enterprise modernization, enterprise architecture tools, and enterprise architecture framework.

The selection of academic and grey literature adhered to the following inclusion criteria:

> **IC1:** The publication must explicitly address legacy system modernization
>
> **IC2:** The context must involve brownfield systems or enterprise-scale modernization efforts

This study examines EA tools based on their support for critical modernization domains central to digital transformation. To capture the complexity, modernization can be framed around six foundational domains of EA. The following analysis explores these domains in detail and evaluates the extent to which leading EA tools support targeted modernization efforts within each area.

The core domains include [27, 14]:

- **Infrastructure**: Enables flexible, scalable, and resilient computing environments through modernization of networks, servers, and deployment platforms—whether on-premise, hybrid, or cloud-native.

- **Business Processes**: Optimizes and automates workflows to eliminate inefficiencies, reduce manual intervention, and increase organizational agility and responsiveness.

- **Collaboration and Communication**: Ensures seamless cross-functional teamwork, transparent decision-making, and knowledge sharing across geographically dispersed teams.

- **Cybersecurity and Compliance**: Strengthens security posture and governance by addressing outdated security models, managing risk, and meeting evolving regulatory requirements.

- **Data Integration and Interoperability**: Breaks down data silos, enables real-time access and analytics, and ensures consistency across systems through standardized data exchange.

- **Software Development and Deployment**: Enhances speed, quality, and reliability of software delivery through agile methodologies, CI/CD pipelines, automated testing, and modern toolchains.

The modernization techniques include [2, 8, 17]:

- **Cloud Computing**: Provides scalable, elastic, and cost-efficient infrastructure that supports rapid deployment and high availability of applications and services.

- **Artificial Intelligence**: Automates repetitive tasks, enhances decision-making, and enables intelligent systems through machine learning, NLP, and predictive analytics.

- **Platform Engineering and DevOps**: Integrates development and operations through self-service infrastructure, automation, and standardized toolchains to accelerate deployment and reduce human error.

- **System Architecture**: Adopts modular architectural patterns such as microservices, APIs, and event-driven systems to promote decoupling, scalability, and system resilience.

Each domain and technique reflects both a technical necessity and a business enabler, ensuring modernization efforts are comprehensive, secure, data-driven, and aligned with modern enterprise goals

4. RESULT

The results present a detailed comparative analysis of eight EA tools, all recognized as leaders in Gartner's Magic Quadrant, focusing on their role in supporting legacy system modernization. Through a systematic evaluation of tool capabilities, we identify how each product addresses key modernization priorities, revealing a range of strengths, weaknesses, and specialization patterns. The comparison highlights both broad functional coverage and niche focus areas, offering insight into how different tools align with various modernization strategies and enterprise needs.

Table 1 provides an aggregated view of the tools' support across the evaluated dimensions, allowing for a clear side-by-side comparison of their overall modernization readiness. It serves as a quick reference for understanding which tools offer comprehensive versus selective support. Table 2 complements this by offering a more detailed breakdown, mapping specific features and capabilities to distinct modernization domains. This enables a nuanced understanding of how well each tool supports emerging practices and technologies, such as cloud adoption, AI integration, and continuous delivery. Together, these tables form the basis for identifying the most suitable EA tools for organizations aiming to modernize their legacy systems in a strategic and scalable manner.

4.1 Core Domains

4.1.1 Infrastructure

Top EA tools such as Bizzdesign, OrbusInfinity, MEGA HOPEX, and SAP LeanIX provide detailed visualization and modeling of IT infrastructure, enabling organizations to assess legacy systems and plan modern alternatives. These tools help modernization by exposing outdated hardware, software, and network dependencies, which can then be prioritized for upgrade, cloud migration, or retirement. For example, MEGA HOPEX automates cloud migration recommendations, and Bizzdesign aligns IT systems with evolving business needs. Through these capabilities, EA tools promote agile, future-proof infrastructure that supports digital transformation. Importantly, they support legacy system modernization by offering transformation roadmaps, dependency analysis, and phased decommissioning strategies. Vendors like Bizzdesign have announced future support for digital infrastructure twins and real-time telemetry from cloud platforms, which will enhance infrastructure visibility and lifecycle planning.

4.1.2 Business Processes

Modern EA tools embed business process modeling to link operations directly to technology assets. BOC ADOIT and OrbusInfinity enable enterprises to visualize end-to-end processes and align them with applications and data. This supports modernization by allowing organizations to reengineer inefficient workflows, eliminate redundancy, and digitize manual operations. By tying processes to capabilities and systems, EA tools guide strategic process transformation that aligns with cloud, AI, and digital service goals. Legacy business processes can be assessed and restructured using these tools to align with modern ERP or SaaS platforms. Upcoming features in ADOIT include low-code BPMN automation and tighter integration with robotic process automation (RPA) platforms to further accelerate legacy process modernization.

4.1.3 Collaboration and Communication

EA tools foster collaborative ecosystems by offering role-based access, shared dashboards, and integration with platforms like Teams, Confluence, and SharePoint. OrbusInfinity and LeanIX promote cross-functional alignment, essential for modernization initiatives which require input from business, IT, and compliance stakeholders. Modernization is accelerated through real-time collaboration, transparent change management, and feedback loops enabled by these features, creating a unified architectural vision that drives coherent transformation. Legacy system modernization is improved through stakeholder engagement and shared decision-making, ensuring consensus and visibility during complex change programs. Some vendors are planning AI-powered collaboration agents and multilingual support to further democratize architectural knowledge sharing and accelerate modernization at global scale.

4.1.4 Cybersecurity and Compliance

Platforms like OrbusInfinity and MEGA HOPEX integrate risk, security, and compliance into the EA landscape. They map security controls, track regulatory compliance, and support audit trails. These tools help modernization by embedding security-by-design into transformation initiatives. As systems evolve toward cloud, AI, and microservices, the EA ensures that compliance and cybersecurity architectures scale accordingly. EA tools reduce modernization risk and ensure governance over rapidly changing tech environments. For legacy systems, this includes the identification of outdated or vulnerable components, mapping them to compliance frameworks (e.g., NIST, ISO 27001), and planning secure replacements. Future enhancements include automated threat modeling and integration with zero-trust architecture principles, enabling secure legacy transitions.

4.1.5 Data Integration and Interoperability

Integration is central to tools like SAP LeanIX and OrbusInfinity, which offer native connectors to ITSM, CMDB, cloud platforms, and project management systems. These integrations support modernization by enabling real-time, data-driven architecture. Interoperability with agile tools and CI/CD systems ensures that the EA remains aligned with operational realities. This dynamic architecture, enriched with external data, accelerates insight, transformation, and iterative modernization at scale. Legacy systems

often operate in silos; EA tools help expose and bridge these silos through data lineage and standardized APIs. Planned features include enhanced semantic data mapping and data fabric integrations to better unify legacy and cloud-native ecosystems.

4.1.6 Software Development and Deployment

LeanIX and OrbusInfinity bridge EA with software engineering by integrating with DevOps pipelines, CI/CD workflows, and source control systems. EA repositories automatically update based on code changes, microservice manifests, or deployment artifacts. This alignment helps modernization by eliminating architectural drift, enabling faster deployment cycles, and maintaining architectural integrity. It ensures that modernization is not just planned but continuously realized through ongoing development. For legacy applications, EA tools support refactoring planning, dependency refactoring, and microservice segmentation. Looking forward, vendors are building enhanced DevSecOps integrations and support for architecture-as-code to streamline modernization efforts within development lifecycles.

4.2 Modernization techniques

4.2.1 Cloud Computing

Cloud transformation is embedded in EA tools via migration modeling, cloud-native architecture views, integration with platforms like AWS, Azure, and GCP, and planning tools based on the 6R framework (rehost, replatform, refactor, etc.). SAP LeanIX enables cloud strategy formulation and scenario modeling, while MEGA HOPEX offers migration wave planning and cost-risk optimization. These features modernize architecture by deconstructing monoliths, phasing legacy retirement, and fostering hybrid or multicloud strategies. EA tools help organizations move to scalable, flexible, and cost-effective cloud infrastructures. For legacy systems, this involves mapping current dependencies, modeling workload portability, and selecting appropriate modernization strategies such as containerization, virtualization, or serverless deployment. Future plans include real-time cloud cost modeling, carbon footprint tracking, and sustainability scoring of cloud environments to guide green modernization strategies that align with ESG mandates.

4.2.2 Artificial Intelligence

Vendors like SAP LeanIX, BOC Group, and MEGA integrate AI and ML for recommendation engines, anomaly detection, auto-classification, and automated modeling. AI supports modernization by accelerating decision-making, providing predictive insights, and reducing manual modeling overhead. For instance, AI can suggest optimal migration paths, detect obsolete components, surface emerging architecture patterns, and prioritize system updates. These capabilities enhance agility and future-readiness, allowing EA teams to lead AI adoption with structured oversight and repeatable frameworks. In the context of legacy systems, AI can aid in reverse engineering business rules, identifying redundant logic, clustering applications for rationalization, and recommending target-state transitions. Future AI capabilities include generative design of architecture states, LLM-assisted documentation, AI-powered impact analysis, and integration with code-level intelligence systems to drive holistic modernization.

4.2.3 DevOps and Platform Engineering

Modern EA tools are increasingly integrated with DevOps environments and platform engineering workflows. SAP LeanIX supports automatic discovery of microservices, service catalogs, and runtime metadata via CI/CD pipelines, ensuring that architecture models remain up-to-date. OrbusInfinity connects architectural views with agile project tracking, deployment pipelines, and runtime observability platforms. These integrations help modernization by linking architecture to execution, enabling faster time to value, and supporting continuous delivery of architectural change. Platform engineering is further enabled through architecture-as-code, Kubernetes-native modeling, and declarative deployment modeling. Legacy modernization is facilitated by mapping legacy workloads to platform-ready environments, modularizing system components, and phasing out rigid monoliths through gradual, risk-managed modernization sprints. Future developments include tighter integration with internal developer platforms (IDPs), automated drift correction, runtime-to-model reconciliation, and feedback loops between telemetry and architectural governance.

4.2.4 System Design and Architecture

EA tools such as ADOIT and OrbusInfinity provide multilayered system modeling using industry standards like ArchiMate, BPMN, UML, and C4. They support modernization by enabling agile architecture iterations, creating digital twin scenarios, and aligning IT systems with business capabilities and services. This technique ensures that new digital platforms are cohesive, reusable, and strategically aligned. Design-driven modernization leads to robust, scalable ecosystems and reduced architectural debt. For legacy systems, these models enable what-if scenario planning, visualization of technical debt hotspots, evaluation of alternative design paths, and structured transformation planning. Vendors are planning new features such as support for edge computing architectures, dynamic architectural heatmaps, and cloud-native pattern libraries to better manage the design and modernization of legacy environments.

5. DISCUSSION

This section presents and interprets the comparative findings, focusing on how various EA tools support legacy system modernization and differ in their capabilities across six key domains. Rather than listing features, the discussion emphasizes how these tools are applied in practice and what strategic value they provide in real-world modernization contexts.

5.1 EA tool support across core domains

SAP LeanIX emerges as a leader in infrastructure modernization. Its automated workload discovery and support for 6R migration planning enable organizations to rapidly generate cloud transformation roadmaps and significantly reduce manual planning effort. LeanIX's wave-planning and cost modeling features streamline cloud decision-making, minimize legacy downtime, and mitigate risk during transitions. Ardoq is also highly effective in the software development domain, offering real-time synchronization with Git repositories and enabling architecture-as-code—crucial in agile, fast-evolving environments.

Tool	Domain / Technique	Legacy System Modernization Support
SAP LeanIX[13]	Cloud Computing	Facilitates migration of legacy applications to cloud by mapping workloads and dependencies, supporting hybrid and multi-cloud strategies.
	AI	Applies AI to identify technical debt and prioritize modernization opportunities within legacy portfolios.
	DevOps and Platform Engineering	Integrates with CI/CD pipelines enabling automated legacy application lifecycle updates.
	Data Integration	Connects with CMDBs and cloud platforms to maintain up-to-date inventories critical for planning modernization.
	System Architecture	Provides time-based views highlighting aging legacy components and suggesting phased replacement paths.
OrbusInfinity[4]	Business Processes	Maps outdated legacy processes enabling redesign aligned with modern capabilities.
	Collaboration and Communication	Offers stakeholder dashboards fostering alignment and consensus on modernization priorities.
	Cybersecurity and Compliance	Detects compliance gaps in legacy systems and supports mitigation planning.
	System Architecture	Visualizes legacy architecture decay and enables scenario planning for modernization.
	Cloud Computing	Builds target-state roadmaps supporting migration of legacy components to cloud infrastructure.
MEGA HOPEX[3]	Infrastructure	Identifies legacy infrastructure dependencies, enabling phased migration plans to reduce modernization risk.
	Cybersecurity and Compliance	Maps outdated security practices and helps align with modern frameworks (e.g., NIST, GDPR).
	AI	Uses AI to analyze usage, cost, and risk to recommend optimization and modernization strategies.
	Cloud Computing	Assesses cloud readiness of legacy components to prioritize migration candidates.
	System Architecture	Offers layered, time-based views facilitating legacy component decommissioning and replacement.
Bizzdesign[30, 26]	System Architecture	Enables capability-based modernization by identifying gaps between legacy and current systems.
	Business Processes	Connects legacy processes with updated business objectives to drive value-driven modernization.
	Cloud Computing	Uses dependency mapping to guide smooth legacy app migration to cloud-native environments.
	AI (Planned)	Developing AI-powered tools to auto-generate optimized modernization options.
BOC ADOIT[1]	Business Processes	Identifies inefficiencies in legacy workflows enabling automation and process modernization.
	System Architecture	Provides comprehensive “as-is” and “to-be” models guiding stepwise legacy system transformation.
	AI (Planned)	Future AI features aim to auto-generate modernization scenarios.
	Collaboration and Communication	Facilitates shared modeling improving coordination among modernization teams.
Ardoq[15]	DevOps and Platform Engineering	Auto-syncs with Git and CI/CD pipelines, enabling continuous delivery of legacy software updates.
	Data Integration	Uses graph-based modeling to reveal hidden dependencies critical for safe modernization.
	System Architecture	Maintains real-time architectural views tracking modernization progress.
QualiWare[24]	Collaboration and Communication	Provides role-based views engaging technical and business stakeholders in modernization projects.
	Cybersecurity and Compliance	Audits outdated controls and supports remediation for compliance modernization.
Capsifi[10]	Business Processes	Aligns legacy operations with updated capability and value chains to maximize modernization ROI.
	AI (Planned)	Developing AI-driven transformation strategy recommendations for legacy-heavy environments.

Table 1: Enterprise Architecture Tools and Legacy System Modernization Support Across Core Domains and techniques

Tool	Infrastructure	Business Processes	Collaboration & Communication	Cybersecurity & Compliance	Data Integration & Interoperability	Software Development and Deployment
SAP LeanIX	•		•	•	•	•
OrbusInfinity	•	•	•	•	•	•
MEGA HOPEX	•		•	•	•	•
Bizzdesign	•	•	•	•	•	
BOC ADOIT	•	•	•	•	•	
Ardoq	•		•	•	•	•
QualiWare	•		•	•	•	•
Capsifi	•	•	•	•	•	

Table 2: EA Tools and Their Support for Core Modernization Domains

In the Business Processes domain, Orbus Infinity and Bizzdesign demonstrate strong modeling and alignment capabilities. Orbus Infinity's integration with Microsoft Teams and Power BI fosters real-time stakeholder engagement, reducing delays and promoting IT-business alignment. Bizzdesign offers in-depth BPMN, ArchiMate, and value stream modeling to simulate and optimize workflows. Capsifi, although requiring more customization, excels in strategic modeling and business capability maps.

Collaboration and communication features vary among tools. Orbus Infinity stands out for its M365 integration, while LeanIX and Ardoq offer dynamic, role-based dashboards. QualiWare also supports governance and stakeholder-specific views, though with a steeper learning curve. In the compliance domain, MEGA HOPEX clearly leads with built-in GRC frameworks, automated mapping to standards like ISO 27001 and NIST, and audit support. BOC ADOIT and Bizzdesign offer GRC capabilities but require more manual configuration. SAP LeanIX and Ardoq provide certifications and documentation tools, though they lack full compliance modules.

Regarding data integration, SAP LeanIX, Orbus Infinity, and Ardoq provide the most extensive support, with broad API access, webhook functionality, and connectors for platforms like CMDB, ITSM, and cloud services. This facilitates automation and reduces data entry burdens. While MEGA HOPEX and BOC ADOIT support integration, they often require add-ons or consulting. For software development and deployment, LeanIX and Ardoq again lead the field, supporting CI/CD pipelines, GitOps, and automated service mapping. Orbus Infinity and QualiWare offer some integration, while MEGA HOPEX and BOC ADOIT currently lag behind.

In summary, tool selection should depend not only on domain coverage but also on the operational depth of each capability. LeanIX suits organizations emphasizing speed and automation in cloud migration. Orbus Infinity is ideal for enterprises seeking business-IT alignment. MEGA HOPEX offers robust compliance support, while Ardoq is best for agile development. Additionally, Bizzdesign and BOC ADOIT provide strong modeling for governance-focused organizations. Capsifi and QualiWare deliver strategic planning strengths, but it may require greater investment in setup and training. These distinctions highlight the need to match tools to specific modernization goals and organizational contexts.

5.2 Scope and Limitations

This research is based exclusively on secondary data sources and does not incorporate primary data collection such as interviews, surveys, or direct observations. Consequently, internal organizational dynamics and contextual nuances influencing tool adoption may not be fully represented. Furthermore, the applicability of findings may vary depending on organizational scale, industry sector, technological maturity, and geographical context. Despite these limitations, the study offers a structured and comparative perspective on tool-assisted modernization and provides a foundation for future empirical studies, benchmarking efforts, and practical decision-making frameworks.

6. CONCLUSIONS

In conclusion, the leading EA tools highlighted in Gartner's Magic Quadrant 2024 each offer distinctive capabilities that collectively support comprehensive legacy system modernization. SAP LeanIX excels in integrating cloud computing, AI, and DevOps, providing dynamic, data-driven insights that enable organizations to map, prioritize, and execute modernization initiatives with agility and precision. OrbusInfinity stands out for its strong alignment between business processes and IT architecture, ensuring modernization efforts deliver clear business value while managing complexity and risk effectively. MEGA HOPEX is particularly valuable in highly regulated sectors, embedding cybersecurity, compliance, and risk management into legacy transitions to safeguard governance and continuity.

Tools such as Bizzdesign, ADOIT, and Ardoq contribute significantly through advanced architectural modeling, dependency mapping, and scenario simulation, which are critical for decomposing legacy monoliths and planning phased migrations with minimal disruption. Their ongoing support for continuous architecture monitoring and collaboration empowers organizations to maintain control over complex modernization journeys.

By leveraging these EA platforms, enterprises can reduce technical debt, accelerate cloud adoption, and foster more scalable, and secure IT ecosystems. Their ability to bridge legacy and modern architectures through actionable insights and integrated workflows is essential for navigating today's digital transformation challenges. Selecting the right EA tool—and exploiting its full modernization capabilities—remains a strategic imperative for businesses committed to sustainable, future-ready legacy system renewal.

7. REFERENCES

[1] Boc adoit. https://www.boc-group.com/en/blog/ea/ea-study-2024, 2025.

[2] Gartner, 2025. URL: https://gartner.com.

[3] Mega hopex, 2025. URL: https://www.mega.com/blog/application-modernization-introduction-and-best-practices#:~:text=,based%20on%20ROI%2C%20risk%2C%20and.

[4] Orbusinfinity, 2025. URL: https://www.orbussoftware.com/resources/blog/detail/how-enterprise-architecture-powers-federal-modernization#:~:text=According%20to%20a%20Forrester%20Total,OrbusInfinity%20can%20help%20agencies%20achieve.

[5] Issam Al-Azzoni, Lei Zhang, and Douglas G. Down. Performance evaluation for software migration. In *Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering*, ICPE '11, page 323–328, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1958746.1958792.

[6] Adina Aldea, Maria-Eugenia Iacob, Andreas Wombacher, Marlon Hiralal, and Thijs Franck. Enterprise architecture 4.0 – a vision, an approach and software tool support. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, pages 1–10, 2018. doi:10.1109/EDOC.2018.00011.

[7] Assia Alexandrova, Lucia Rapanotti, and Ivan Horrocks. The legacy problem in government agencies: an exploratory study. In *Proceedings of the 16th Annual International Conference on Digital Government Research*, dg.o '15, page 150–159, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2757401.2757406.

[8] Zainab Asimiyu. Modernizing enterprise it systems the intersection of cloud computing, ai, and knowledge management. 12 2023.

[9] Wesley Assunção, Luciano Marchezan, Alexander Egyed, and Rudolf Ramler. Contemporary software modernization: Perspectives and challenges to deal with legacy systems, 07 2024. doi:10.48550/arXiv.2407.04017.

[10] Madhushi Bandara, Fethi A Rabhi, Rouzbeh Meymandpour, and Onur Demirors. A digital interaction framework for managing knowledge intensive business processes. In *Australian Symposium on Service Research and Innovation*, pages 108–122. Springer, 2018.

[11] Santiago Comella-Dorda, Kurt Wallnau, Robert Seacord, and John Robert. A survey of legacy system modernization approaches. page 30, 04 2000.

[12] Adenekan Dedeke. Improving legacy-system sustainability: A systematic approach. *IT Professional*, 14(1):38–43, 2012. doi:10.1109/MITP.2012.10.

[13] Alexander Ettinger. Enterprise architecture as a dynamic capability for scalable and sustainable generative ai adoption: Bridging innovation and governance in large organisations. *arXiv preprint arXiv:2505.06326*, 2025.

[14] Maulahikmah Galinium and Negar Shahbaz. Case studies: Business and technical perspectives in migration of legacy systems to soa. *arXiv preprint arXiv:1412.7959*, 2014. URL: https://arxiv.org/abs/1412.7959, doi:10.48550/arXiv.1412.7959.

[15] Hong Guo and Shang Gao. Ai application in enterprise architecture management: From the perspective of tools. In *Proceedings of the International Conference on Electronic Business (ICEB)*, number 67, 2024. URL: https://aisel.aisnet.org/iceb2024/67.

[16] Chandrasekaran K. H. SeetharamaTantry, Murulidhar N. N. Implications of legacy software system modernization -a survey in a changed scenario., 2017.

[17] Zeshan Haider and John Yang. Revolutionizing enterprise architecture: Harnessing ai and cloud synergy with devops integration, 11 2024. doi:10.13140/RG.2.2.33117.63200.

[18] Federico Heras. A comparison of enterprise architecture tools. In *Proceedings of the 20th International Conference on Smart Business Technologies - Volume 1: ICSBT*, pages 186–192. INSTICC, SciTePress, 2023. doi:10.5220/0012121500003552.

[19] Rabie Khabouze. *Modernization of Legacy Information Technology Systems*. PhD thesis, Walden University, 2022.

[20] Ravi Khadka, Belfrit Batlajery, Amir Saeidi, Slinger Jansen, and Jurriaan Hage. How do professionals perceive legacy systems and software modernization?, 05 2014. doi:10.1145/2568225.2568318.

[21] Marc M. Lankhorst. Enterprise architecture modelling—the issue of integration. *Advanced Engineering Informatics*, 18(4):205–216, 2004. Enterprise Modelling and System Support. URL: https://www.sciencedirect.com/science/article/pii/S1474034605000054, doi:https://doi.org/10.1016/j.aei.2005.01.005.

[22] Abir M'baya, Jannik Laval, and Nejib Moalla. An assessment conceptual framework for the modernization of legacy systems. In *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pages 1–11, 2017. doi:10.1109/SKIMA.2017.8294120.

[23] Adya Mishra. Legacy system modernization: Effective strategies and best practices, 11 2020. doi:https://doi.org/10.5281/zenodo.14769544.

[24] Marco Nardello. Enterprise architecture for digital manufacturing: Ea models and an automated ea modelling method to support industry 4.0 transformation. 2019.

[25] Olufunmilayo Ogunwole, Ejuma Adaga, and Augustine Ibeh. Modernizing legacy systems: A scalable approach to next-generation data architectures and seamless integration. *International Journal of Multidisciplinary Research and Growth Evaluation.*, 4, 04 2025. doi:10.54660/.IJMRGE.2023.4.1.901-909.

[26] Anastasios Papazoglou. Capability-based planning with togaf® and archimate®. Master's thesis, University of Twente, July 2014. URL: http://essay.utwente.nl/65421/.

[27] Christophe Ponsard. Assessing it architecture evolution using enriched enterprise architecture

models. *arXiv preprint arXiv:2204.06226*, 2022. URL: https://arxiv.org/abs/2204.06226, doi:10.48550/arXiv.2204.06226.

[28] Maria Raksi. Modernizing web application: case study, 2017.

[29] Jaap Schekkerman. *How to survive in the jungle of enterprise architecture frameworks: Creating or choosing an enterprise architecture framework.* Trafford Publishing, 2004.

[30] Emmanouil D. Tritsiniotis. Get ready for the cloud:tailoring enterprise architecture for cloud ecosystems, November 2013. URL: http://essay.utwente.nl/64258/.

[31] Daniele Wolfart, Wesley Assunção, Ivonei Silva, Diogo Domingos, Ederson Schmeing, Guilherme Villaca, and Diogo Paza. Modernizing legacy systems with microservices: A roadmap. pages 149–159, 06 2021. doi:10.1145/3463274.3463334.

A Comparative Evaluation of Traditional REST API Testing Tools and LLM-based Methods

Cristian-Mihai Stratulat
RWTH Aachen University
Software Construction
52074 Aachen, Germany
cristian-mihai.stratulat@rwth-aachen.de

Keven Hu
RWTH Aachen University
Software Construction
52074 Aachen, Germany
keven.hu@rwth-aachen.de

ABSTRACT

In recent years, the accelerating trend of digitalization has led to many critical services - such as banking and insurance - to migrate to modern web applications. REST API represents one of the most common interfaces between these services. As many critical functionalities depend on these APIs, the need for testing to ensure correctness and reliability is greater than ever. In particular, testing against formal definitions such as OpenAPI specifications has gained traction with tools like EvoMaster, Restler and Schemathesis. However, the exploration of potentially better tools remains a focus of research. This paper conducts a comparison in terms of code coverage of the testing suites between the automated testing tool EvoMaster and different widespread large language models (LLMs) like ChatGPT and Google Gemini. Different prompting techniques are tested and analyzed on how they impact the quality of the output. For the scope of the study, two microservices have been chosen that expose an REST API, with the source code available. Instrumentation was performed to measure the code coverage accurately. The LLMs were prompted to produce only test case inputs and corresponding assertions, due to their limit in response, and these test cases were manually executed.

The results of the study seem to indicate that LLMs can outperform EvoMaster while providing more semantically meaningful inputs than the simple fuzzing technique used by the tool. Our study shows that testers can benefit from almost the same efficiency in terms of code coverage when creating test cases using LLMs as they would achieve using an automated testing tool.

Keywords

Software Testing, REST API Testing, Black-Box Testing, Artificial Intelligence, Large Language Models, Prompt Engineering

SWC Seminar 2018/19 RWTH Aachen University, Germany.

1. INTRODUCTION

Microservices are a popular architecture choice for software applications and services. Especially among web services, the microservice architecture is widely spread. They power popular online services like Amazon in online shopping or Netflix[6] and Spotify[15] in streaming. In fact, a survey of around 4300 respondents that work as software developers, engineers, project managers, or other software-related professions reveals that 71% of those respondents use microservices at least partially in their organization[9].

Although there is no universally accepted definition for microservices, several properties characterize them such as modularity, loose coupling, communication over networks, and the ability to be deployed independently of others. For communication over networks, microservices most commonly use REST APIs for an average of 66.8% of messages[16]. Therefore, tools and approaches that support the software development of REST APIs are of great interest. In particular, the area of software testing is important to ensure the correctness and reliability of REST APIs. Tools that support developers in finding test cases, oracles and implementations can be useful since they may save time and shorten the required development time. There are already tools that attempt to automate the testing process based on an OpenAPI specification of the REST API, such as EvoMaster[2]. However, these tools may have drawbacks such as redundant test cases or low code coverage [14]. Thus, new tools and approaches that support software testing of REST APIs should be researched and explored.

In recent years, a new approach to supporting activities from many different fields such as medicine, cybersecurity, or entertainment has become popular. The approach is the use of artificial intelligence, especially in the form of machine learning, in various tasks such as diagnostics[3], threat detection[24] or recommendation systems[26]. Another aspect in which machine learning made improvements possible is natural language processing. Especially the paper "Attention Is All You Need" by Vasqani et al.[25] significantly contributed to improvements in natural language processing. It did so by proposing a new neural network architecture called Transformer that exclusively used attention mechanisms instead of recurrence and convolutions like in traditional recurrent neural networks and convolutional neural networks. So-called large language models, or LLMs for short, are built using the Transformer architecture and are trained on very large data sets in order to solve natural language processing tasks. Among LLMs, some have even found widespread use in the general public, such as ChatGPT[12].

In the case of software engineering, LLMs have been a point of interest for their ability to generate text based on user prompts. For example, the generation of black-box test cases for REST APIs using LLMs has been a focus of research[13]. However, studies conducted in this field have not evaluated how LLMs available in the chat-based context in the form of web interfaces in the browser perform. This is an interesting point of focus, since chat-based web interfaces of LLMs are very accessible to software test developers. We, therefore, studied the capabilities of the LLMs Google Gemini 2.5 Flash, ChatGPT-4o mini and Llama 3.3-70B-Instruct in order to determine how well the chat-based LLMs perform in the task of black-box test case generation for REST APIs. For that purpose, the following research questions are addressed in this paper:

- **RQ1:** Do LLMs have the potential to perform better than classical tools with minimal human interaction in terms of code coverage?
- **RQ2:** How do variations in prompting techniques and choice of LLMs influence the quality of the generated test cases?
  - **RQ 2.1:** Which prompting techniques perform the best?
  - **RQ 2.2:** How do different LLM models influence the quality of the outputted test cases?

2. BACKGROUND

2.1 REST API

Representational State Transfer, or REST for short, is an architectural style for distributed systems, especially web services. It focuses on the scalability of services, encapsulation of components, and the design of application programming interfaces, or API for short. Data or files are resources that have a representation that captures the current or intended state of that resource. Software components can perform actions on the resource by sending the intended representation of that resource using a RESTful API. Some core concepts of REST are for example the separation of client and server which act independently and communicate through requests from the client and responses from the server, uniform interfaces where requests always contain a unique resource identifier and resources are manipulated through representations or the server not remembering information regarding a client[17].

A client includes a verb that describes one of the CRUD (create, read, update, delete) operations and a unique resource identifier. Responses of a server include an HTTP status code and a response body that may contain a representation of a resource that was requested or a description of the response. Status codes are three-digit long numbers and have a prefix from 1 to 5. The most common codes have prefixes 2, 4 and 5, where 2 signals a successful request, 4 indicates an error from the client side, and 5 means an error from the server[17].

In order to describe a REST-API in a formal and standardized way, the OpenAPI specification can be used[19]. It is a sort of contract by which the API is bound to ensure that requests and responses have a standardized form.

2.2 Prompting Techniques for LLMs

Large language models operate on the basis of user inputs. These inputs are called prompts and their structure or content may impact the quality of the output produced by the LLM. For example, the level of detail of the instructions in the prompt can impact the quality of the output. There are different techniques that aim to improve the quality of the LLM output beyond a simple instruction to perform a task.

0-shot Prompting
0-shot prompting refers to prompts that do not contain any examples of the expected output[22].

Few-shot Prompting
The technique is called few-shot prompting if the prompt contains at least one or more examples of the expected output. The aim is for the LLM to learn to complete a task based on the provided examples[22].

Persona Prompting
In persona- or role prompting, the LLM is assigned a specific role. This may produce more fitting outputs for open-ended tasks[22].

Chain-of-Thought
In chain-of-thought prompting (COT), the LLM is instructed to express its thought process before giving a final answer. In the special case of combining COT with 0-shot prompting, a phrase like "Let's think step by step,..." is added to the prompt in order to induce the generation of the thought process[22].

3. RELATED WORK

The quest for robust and efficient automated testing has been a task that has drawn attention for some time resulting in a diverse landscape of tools and methodologies. This chapter will briefly present the most relevant developments that have been made and their current limitations.

One of the primary tasks that one such tool needs to fulfill is the generation of relevant inputs and sequences to exercise the system under test. Exhaustion of the entire input space is never feasible [23], and it is common sense to assume that any trivial algorithm that would try to generate a combination with this exhaustion purpose will ultimately fail. However, various tools have adopted more intelligent and targeted philosophies. These approaches move beyond brute force and leverage OpenAPI specification parsing and behavioral feedback. Popular tools in the field are Schemathesis[11], which excels at generating diverse inputs and finding edge cases, RESTler [4], developed by Microsoft, which has a focus on producer-consumer dependencies among requests and RestTestGen [7], an extensible framework to implement new automated black-box strategies, while still enforcing care for dependency problems via their Operation Dependency Graph component.

One tool that also deserves a mention, being one of the first pioneers in the automated testing for REST API world as well as harness algorithms inspired by machine learning principles is EvoMaster [2]. It operates on the paradigm of Search-Based Software Testing, employing an Evolutionary Algorithm, the main idea being that test cases generated are continuously refined based on the responses from the system.

Another challenging task in the field of software testing is the problem of test oracle generation. This problem is sometimes considered an important bottleneck in the current development of better, automated testing tools [5]. Though not the main focus of the paper, it is worth mentioning that research has begun to explore ways in which they can further improve this problem, AGORA being one such example [1]

The tools described above have been evaluated and their limitations have been discovered. For example, as observed by Kim et al. [14] they have issues in producing well-adjusted input values specifying certain constraints and they struggle to satisfy dependencies among requests. Without a test oracle generation tool, they often time produce rather simple assertions, on status codes and schema format hence making them prone to miss failures.

With the rise of powerful LLMs, there has been as well a trend in research to find ways to bring their qualities to improve further the broad field of software testing[27] [21]. Specifically talking about our niche of REST API testing, we have seen some prototypes developed such as RESTGPT [13], but even though they look promising, they are still in a rather early phase, lacking the robustness and user-friendliness typically required for widespread, off-the-shelf adoption by practitioners.

Despite these advancements and the promising early prototypes, a significant research gap remains in empirically evaluating the real-world performance of LLMs—even when accessed via a chat-based interface—in direct comparison to classical, established automated testing tools. Our work aims to contribute precisely by gathering quantitative data in this under-explored direction.

4. RESEARCH DESIGN

This chapter will describe in detail our design and the reasons for our empirical investigation. Specifically, we explain the choices made regarding the systems under test selection, LLM models, evaluation metrics, prompt crafting techniques and tools used in our study. It is important to mention from the beginning that our study does not represent a controlled experiment with its classical semantics. Unlike traditional experimental designs that aim to establish causal relationships, our approach is primarily observational. Our objective is to conduct a series of data-gathering empirical evaluations designed to characterize the performance of LLMs in a specific domain.

We start by presenting what is the chosen system that the generated test suites will be tested against and the reasons for choosing it. Previously in our paper we have named some tools that will be used (i.e.: LLMs, classical tools for REST API black box testing) but now we will provide a justification why we have picked exactly these instruments and not others.

We continue by showing how we concretely measure the quality factor previously mentioned: code coverage, and end with a brief description of our prompt crafting strategies

4.1 System and Tool selections

The first major decision that had to be made was in deciding what the system under test would be. The following requirements guided our search:

1. The system must be open source to be able to instrument it to measure code coverage
2. The system must expose a REST API interface, both with trivial and complex endpoints featuring parameters and interdependencies (e.g., requiring an ID generated from a prior POST request).
3. The system should provide an OpenAPI specification that can be used as input for test generation.

Matching all of the above criteria, we have decided to use the PiggyBank project [28] as our target system. This project contains multiple microservices and we have chosen the microservices *transfer-gateway* and *account-twin-services* to be tested. The reason for choosing the two of them is that the transfer-gateway service exposes a rather simple interface, while the account-twin-service presents endpoints parameters and interdependency, making it interesting to observe the behavior under test.

When it comes to picking a traditional automated test generation tool, as previously presented in the related work section there are a number of tools that we can choose from. Our choice was EvoMaster as it was shown by Kim et al. [14] that it is a black-box testing tool that performs well.

Selecting from the vast majority of LLMs was again a crucial decision. The field is moving very fast, and there are many new models released in the near future, potentially delivering stronger results. These models have fundamental differences: some are proprietary and closed-source, while others are open-source, with the possibility of even self-hosting them. Another characteristic would be the usage possibility: as access is price-based and subject to limitations (e.g.: rate limits, context window size).

We included in our study only models that were freely accessible to us — either directly through their official websites or indirectly via third-party platforms provided by our university. The decision leads to the limitation that not the most powerful models were tested, accessibility being preferred. Given these conditions, we selected the following three models:

- ChatGPT-4o mini - provided by OpenAI, a company with a good reputation in the industry, a wide-spread model
- Gemini 2.5 Flash - provided by Google, one of their most powerful models on the free tier
- Llama 3.3-70B-Instruct - developed by Meta, being an open-source model, with self-hosting possibilities

We accessed the LLMs ChatGPT-4o mini and Llama 3.3-70B-Instruct via RWTHgpt [20], allowing us to use the models consistently up to a certain token limit. We preferred this option since the ChatGPT normal interface, on the free tier, does not allow the setting of a specific model (the model might change in between messages), and we could not self-host Llama hence RWTHgpt was a convenient solution. Gemini web interface provides [10] the option to select the model, and the token window was large enough to allow us to conduct our tests.

4.2 Evaluation Metrics

As already mentioned before in the research question, the main focus of our study is code coverage.

When measuring it, various tools can be used, provided access to the application's source code is available. In our

setup, it is important to mention that both microservices include functionality beyond their REST API endpoints. For instance, they also utilize RabbitMQ as a communication protocol. This introduces a limitation: no matter how thorough the REST API tests are, they cannot achieve full application coverage due to the presence of logic that is triggered through other means.

The code base of each microservice is structured into multiple packages, each package having a different responsibility within the application. Among these, two packages are of particular interest for our study: the *controller* package and the *service* package. To obtain a more granular understanding of the generated test suites, we measured code coverage at three distinct levels:

- **Application-wide coverage**, showing the overall test coverage across the entire microservice
- **Controller-level coverage**, focusing on the code contained in the *controller* package. This layer is responsible for handling the REST API interface
- **Service-level coverage**, which focuses on the code in the *service* package. This layer implements the core business logic and operations exposed via the REST API

This separation allows us to better understand the impact of REST API testing on different parts of the application and to interpret the results with more granularity.

4.3 Prompt crafting

To analyze our second research question, we tried out three different prompting techniques, each representing commonly used approaches in prompt-based interactions [22]:

- 0-shot prompting: "We aim to conduct comprehensive REST API testing. Generate enough black-box test cases in natural language to achieve reliable testing..."
- Persona prompting: "You are a QA engineer experienced in REST API testing. Your goal is to conduct comprehensive REST API testing..."
- Chain of thought prompting: "Your goal is to conduct comprehensive REST API testing. First, think step by step about what endpoints exist and what tests you would need..."

The LLMs were asked to provide the test cases in a natural language focusing on the quality of the tests and the assertions, without trying to produce test runners.

We have analyzed these prompt techniques both in a low-information setting, providing generic instructions, as well as in a high-information setting, being more specific about the task (i.e.: asking to pay attention at endpoints dependency, asking to provide happy paths, corner cases). This resulted in six test suites for every model we tested.

Initially, we considered applying few-shot prompting and multi-turn (interactive) prompting. However, we ultimately decided against it. Few-shot prompting did not noticeably improve output quality, likely because the LLMs already demonstrated strong general knowledge of REST API testing. Moreover, providing too many specific examples risked undermining the generalization aspect of the evaluation. As for interactive prompting, the systems under test were relatively simple, and we found that a single prompt was sufficient for the models to understand the task context.

4.4 Testbed setup

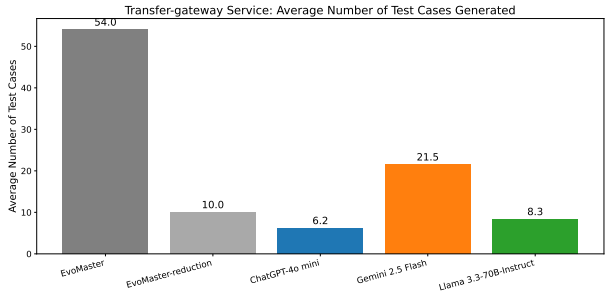
We have run the microservices inside a dockerized environment, where we have installed JaCoCo version 0.8.11 as our code coverage tool. EvoMaster was run inside a container as well, following their official documentation [8]. The time budget that EvoMaster received was two minutes for both microservices. Test cases generated by the LLMs were parsed into a JSON format and were ultimately consumed by a Python script that acted as our test runner.

5. DATA COLLECTED

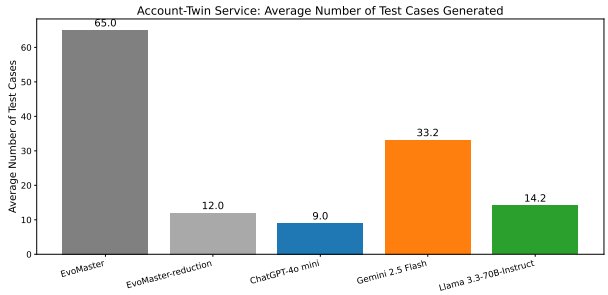
In this chapter, we will present the data obtained from our generated test suites and present some limitations that have to be taken into account.

5.1 Generated tests and coverage achieved

Figure 1 presents the first important observation, presenting the total number of test cases generated by each tool for their corresponding microservice. An important detail to highlight is that EvoMaster initially generates a large test suite, all of which are executed against the system. Subsequently, a reduction step is applied to retain only the most cost-efficient subset of tests based on its internal heuristics. We included both full and reduced results in our analysis. In Figure 2, we can observe the main comparison of this paper. Six test suites were generated for each LLM configuration, corresponding to the different prompting techniques explored. The values presented in both figures for LLMs represent the average across these six variations.



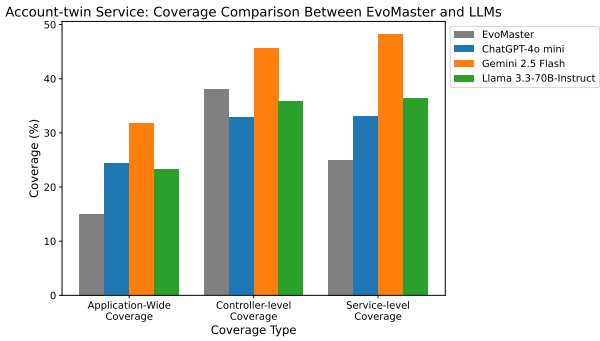
(a) Number of test cases generated for Transfer-gateway Service



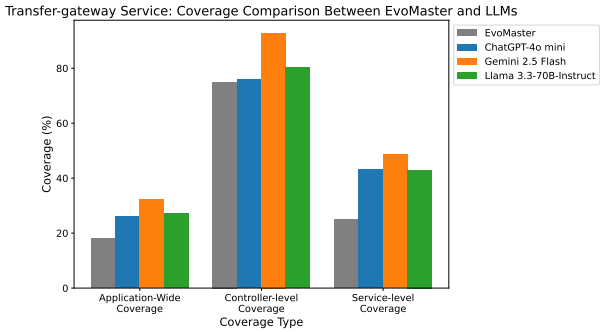
(b) Number of test cases generated for Account-twin Service

Figure 1: Number of test cases generated by LLMs and EvoMaster for both microservices

In Figure 3 and Figure 4, we have provided the individual data of each test suite that was generated and observe how they compare with each other across both microservices.

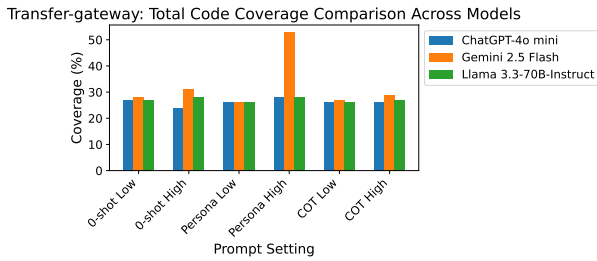


(a) Performance comparison for Account-twin Service

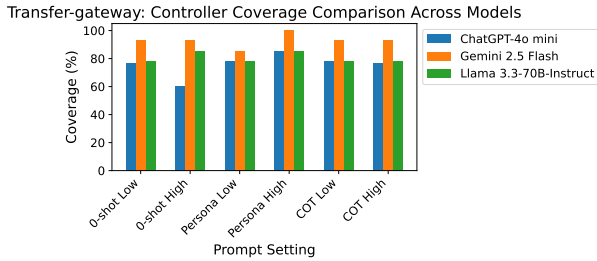


(b) Performance comparison for Transfer-gateway Service

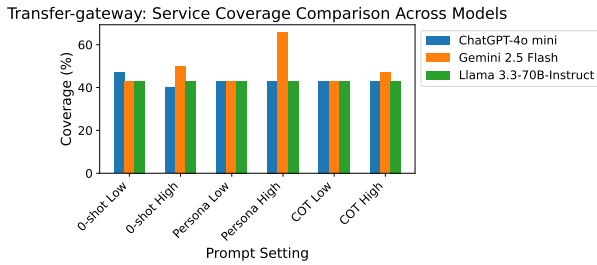
Figure 2: Evomaster's performance compared with LLMs across different microservices



(a) Total code coverage comparison



(b) Controller code coverage comparison



(c) Service code coverage comparison

Figure 3: LLM test suites coverage for Transfer-gateway

The coverage was observed at different levels as explained in the design, to provide a better understanding of the performance achieved.

6. DISCUSSION

6.1 Research Question 1

Based on the research data gathered, it is apparent that all LLMs that were examined in this study outperformed the traditional tool EvoMaster in terms of code coverage in all cases except for the code coverage for the controller of the Account-twin service. It is also notable that, despite achieving a higher code coverage in almost all cases, the LLMs did so with around 49% fewer test cases than those generated by EvoMaster. This shows that LLMs are not only more effective but also more efficient in terms of the number of test cases. A potential reason for the significantly better performance of LLMs may be their reasoning capabilities. It seems that the LLMs are able to find more paths with only the OpenAPI specification. Also, LLMs may be able to find dependencies between endpoints and therefore consider them in their test cases due to their ability to reason. Furthermore, reasoning abilities may explain the efficiency of the sets of test cases. LLMs seem to be able to understand which test cases are not redundant.

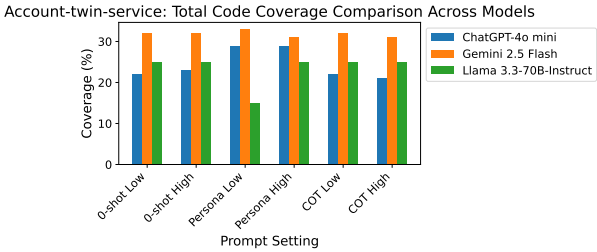
6.2 Research Question 2

6.2.1 *Research Question 2.1*

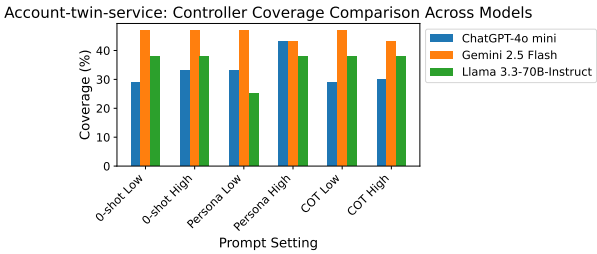
In terms of code coverage, neither the level of detail of the prompt nor the prompting technique used appears to have a definite impact on the quality of the generated test cases. There is no apparent pattern that makes it possible to determine whether a prompting technique has, in general, a positive or negative impact on the quality of the generation. For example, based on Figure 4c about the transfer-gateway service with the Google Gemini 2.5 Flash model, one may assume that using highly detailed persona prompting yields higher quality test cases than other prompting techniques. However, this is refuted by Figure 4a where other prompting techniques such as low detailed persona prompting or highly detailed simple prompting performed better for Google Gemini 2.5 Flash in the case of the service of account-twin service. Only in the case of ChatGPT-4o mini may a pattern be visible. Persona prompting may have a positive impact on the code coverage of the generated test cases for ChatGPT-4o mini.

6.2.2 *Research Question 2.2*

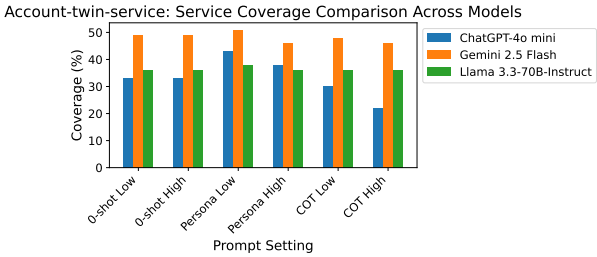
The choice of LLM has the most significant impact on the quality of the generated test cases. Across all comparisons, Google Gemini 2.5 Flash performed the best with a wide margin of up to more than 20% in terms of code coverage. Llama 3.3-70B-Instruct performs about the same or outperforms ChatGPT-4o mini in most cases, with the exception being the persona prompting for the account-twin service case. Considering that ChatGPT-4o mini is a small



(a) Total code coverage comparison



(b) Controller code coverage comparison



(c) Service code coverage comparison

Figure 4: LLM test suites coverage for Account-twin-service

model focused on cost efficiency and fast responses[18], it performed well compared to Llama 3.3-70B-Instruct. Considering that Google Gemini 2.5 Flash is the most recent model among all three with a release date of 20.05.2025, its performance exceeding the other models meets our expectations. However, ChatGPT-4o mini outperforming Llama 3.3-70B-Instruct in some cases is an unexpected result, since ChatGPT-4o mini is an older model released on 18.07.2024 than Llama 3.3-70B-Instruct which was released in December of 2024.

6.2.3 Further Discussions

During the comparison of the test cases between the LLMs and EvoMaster, we noticed that the values that were used as input in the LLM-generated test cases made sense in the context of bank accounts, while EvoMasters values were only a sequence of random characters. This contextual semantic understanding that LLMs exhibit may be helpful for users since the generated test cases become easier to understand. Another notable aspect is the generation time of the examined LLMs. In order to keep the test cases from the LLMs comparable to the ones from EvoMaster, we measured the generation time of the LLMs and made sure that all test cases were generated and output in the chat window within two minutes or less. We chose two minutes because that was the time that EvoMaster had available to generate test cases. Despite delays caused by uncontrollable factors, such as an internet connection with varying stability, the generated test cases of the LLMs were all displayed in significantly less than two minutes with ChatGPT-4o mini even displaying the generation in less than ten seconds in some cases. This further highlights the efficiency of LLMs not only in code coverage compared to the number of test cases but also in the time needed to generate the test cases.

6.3 Limitations

While the results of this study suggest certain patterns and trends, they should not be overgeneralized. Several limitations in the experimental design and setup may have influenced the outcomes, and these factors should be carefully considered when interpreting the findings.

The first limitation we have encountered is the quality of the models. We picked the models that provided us with enough tokens to conduct our testing, however, better results could likely be presented by a stronger model.

The second most important limitation of the study is the amount of context we were able to provide to LLMs. RWTHgpt still offered only a limited number of tokens that we were able to use, both our prompt as well as the response consuming from this budget. Hence, as we wanted to be able to conduct all of our test cases, we had to be careful about how much input information we provided as this could have drastically shortened the length of the test suite generated.

The third and final limitation concerns the number and complexity of the microservices used for evaluation. We chose two microsystems: one relatively simple and one more involved. However, we did not conduct tests across a broader set of systems, which limits our ability to draw consistent or generalizable conclusions about the tools' performance across diverse contexts. Additionally, while the more complex system presented greater challenges, it remained small enough to be processed in full by the LLM within a single prompt. Consequently, it remains unclear how the approach would scale to significantly larger systems that might require multi-prompt strategies or more advanced prompt engineering.

7. CONCLUSION

In this study, we used one traditional tool and three LLMs to generate black box test cases for two endpoints of the REST-API of an open-source software project based on the OpenAPI specifications of the endpoints. We have set up a docker container for using EvoMaster, executing the LLM test cases and monitoring the code coverage of the executed test cases. Our results show that LLMs outperform EvoMaster significantly in terms of code coverage. Furthermore, prompting techniques in general do not seem to have an impact on the quality of the generated test cases. Instead, the most important factor in test case quality seems to be the chosen model, with Google Gemini 2.5 Flash generating the best test cases among all three LLMs.

Considering our obtained data, LLMs have the potential to play an important role in REST API testing in the future and maybe even replace some traditional tools for test case generation. This is mainly due to the reasoning capabilities of LLMs that make for example the discovery of dependencies between endpoints possible. However, the notion of replacing software developers remains unreasonable since knowledge of software testing and REST APIs is still required for the generation of black box test cases for REST

APIs. In addition to that, other quality metrics for software testing need to be examined as well for LLMs to verify our results further. Another direction for further research is training a LLM specifically on a data set curated for software test cases.

8. REFERENCES

[1] J. C. Alonso, S. Segura, and A. Ruiz-Cortés. Agora: Automated generation of test oracles for rest apis. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2023, page 1018–1030, New York, NY, USA, 2023. Association for Computing Machinery.

[2] A. Arcuri. RESTful API Automated Test Case Generation. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 9–20, Prague, Czech Republic, July 2017. IEEE.

[3] S. Asif, Y. Wenhui, S. ur Rehman, Q. ul ain, K. Amjad, Y. Yueyang, S. Jinhai, and M. Awais. Advancements and prospects of machine learning in medical diagnostics: Unveiling the future of diagnostic precision. *Archives of Computational Methods in Engineering*, 32:853–883, 2025.

[4] V. Atlidakis, P. Godefroid, and M. Polishchuk. Restler: Stateful rest api fuzzing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 748–758, 2019.

[5] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, 2015.

[6] N. T. Blog. Rebuilding netflix video processing pipeline with microservices. `https://netflixtechblog.com/rebuilding-netflix-video-processing-pipeline-with-microservices-4e5e6310e359`.

[7] D. Corradini, A. Zampieri, M. Pasqua, and M. Ceccato. Resttestgen: An extensible framework for automated black-box testing of restful apis. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 504–508, 2022.

[8] Webfuzzing/evomaster. `https://github.com/WebFuzzing/EvoMaster`. Accessed: 2025-06-11.

[9] GitLab. A maturing devsecops landscape - 2021 global survey results. `https://about.gitlab.com/images/developer-survey/gitlab-devsecops-2021-survey-results.pdf`.

[10] Google gemini. `https://gemini.google.com/app`. Accessed: 2025-06-11.

[11] Z. Hatfield-Dodds and D. Dygalo. Deriving semantics-aware fuzzers from web api schemas, 2021.

[12] K. Hu. Chatgpt sets record for fastest-growing user base. `https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/`, 2023.

[13] M. Kim, T. Stennett, D. Shah, S. Sinha, and A. Orso. Leveraging Large Language Models to Improve REST API Testing. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ICSE-NIER'24, pages 37–41, New York, NY, USA, May 2024. Association for Computing Machinery.

[14] M. Kim, Q. Xin, S. Sinha, and A. Orso. Automated test generation for REST APIs: no time to rest yet. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2022, pages 289–301, New York, NY, USA, July 2022. Association for Computing Machinery.

[15] Kubernetes. Case study: Spotify, spotify: An early adopter of containers, spotify is migrating from homegrown orchestration to kubernetes. `https://kubernetes.io/case-studies/spotify/#:~:text=An%20early%20adopter%20of%20microservices%20and%20Docker%2C,on%20premise%20data%20centers%20to%20Google%20Cloud.`

[16] A. Lercher, J. Glock, C. Macho, and M. Pinzger. Microservice api evolution in practice: A study on strategies and challenges. *Journal of Systems and Software*, 215:112110, 2024.

[17] M. Masse. *REST API Design Rulebook*. O'Reilly Media, Inc., 1st edition, 2011.

[18] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`.

[19] Webfuzzing/evomaster. `https://swagger.io/specification/`. Accessed: 2025-06-14.

[20] Rwthgpt. `https://genai.rwth-aachen.de/app`. Accessed: 2025-06-11.

[21] R. Santos, I. Santos, C. Magalhaes, and R. de Souza Santos. Are We Testing or Being Tested? Exploring the Practical Applications of Large Language Models in Software Testing. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 353–360, May 2024. ISSN: 2159-4848.

[22] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, and P. Resnik. The Prompt Report: A Systematic Survey of Prompt Engineering Techniques, Feb. 2025. arXiv:2406.06608 [cs].

[23] A. Spillner, T. Linz, and H. Schaefer. *Software Testing Foundations: A Study Guide for the Certified Tester Exam*. Rocky Nook, 3rd edition, 2011.

[24] L. Stanham. Machine learning (ml) & cybersecurity how is ml used in cybersecurity? `https://www.crowdstrike.com/en-us/cybersecurity-101/artificial-intelligence/machine-learning/`.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[26] V. Vorotilov and I. Shugaepov. Scaling the instagram explore recommendations system. `https://engineering.fb.com/2023/08/09/ml-applications/scaling-instagram-explore-recommendations-system/`.

[27] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang. Software Testing With Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering*, 50(4):911–936, Apr. 2024.

[28] N. Wild. Nilswild/piggy-bank. `https://github.com/NilsWild/Piggy-Bank/tree/master`. Accessed: 2025-06-11.

LLMs in Research Methodology: Opportunities and Challenges

Lennart Hüsing
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
lennart.huesing@rwth-aachen.de

Tim Schupp
RWTH Aachen University
Ahornstr. 55
52074 Aachen, Germany
tim@timschupp.de

ABSTRACT

Large Language Models (LLMs) are increasingly applied across a variety of real-world applications, offering notable efficiency gains in several domains. Their potential to support and accelerate scientific research is promising. However, their integration into research workflows also raises challenges and concerns around correctness, ethics, and transparency.

This paper presents a systematic literature review that examines how LLMs are being applied in different areas of research methodology and challenges that arise when using LLM tools in research. A large number of scientific papers were categorized based on similar research use cases.
Findings highlight this topic as very fast-growing, with most studies found being from the last two years.

Potential use cases of LLMs in research methodologies include systematic reviews, surveys, data analysis, and many more. Recurring challenges are hallucinations, lack of depth, and general errors in LLM responses. Additionally, ethical questions arise, and scientific integrity is to be questioned when using LLMs in the research process.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering; D.2.9 [**Software Engineering**]: Management—*research, large language model, research methodologies, research process*

Keywords

Large Language Models, Research Methodology

1. INTRODUCTION

Large Language Models (LLMs) have experienced rapid improvements in recent years, allowing the generation of human-like responses and texts [99]. Additionally, LLMs, like the GPT-series, are now broadly accessible and promise efficiency and quality improvements for specific tasks [74]. This led to the usage of LLMs in various domains, including software engineering, education, and medical tasks [99]. Despite their advantages, LLMs are also seen critically, due to their lack of transparency, environmental impact, and inability to create complex reasoning [39].
Given the vast number of opportunities and significant challenges of LLM tooling, the usage of LLMs in research should be investigated, since they are already being applied to various areas of research [112].
Particularly interesting is the automation of certain processes or approaches in research methodologies. While multiple studies have already investigated how LLMs are integrated into the research process, none have given a partial picture of LLM usage in different areas of research methodologies [59, 34, 66, 11, 15].

Research methodology is a fundamental aspect of scientific work, describing the approaches, techniques, and frameworks to organize and apply research and to gather information systematically [31]. Utilizing these methodologies allows researchers to plan, execute, and analyze their research in order to answer research questions [31].

This work aims to create an overview of areas of research methodologies, where LLMs can be applied, and what challenges arise when doing so. To create an overview, this study makes use of a scientific literature review, aimed at answering the following two research questions.

- **RQ1:** In which areas of research methodology is LLM-based tooling applied and how?
- **RQ2:** What are the challenges of using LLMs in research methodology?

The structure of this work is as follows. First, the methodology is defined in Section 2. Then the results found in the SLR are grouped and presented in Section 3 and eventually discussed in Section 4. This is followed by a discussion of threats to validity in Section 5, and finally, this paper is concluded in Section 6.

2. METHODOLOGY

To answer the specified research questions, a systematic literature review according to Kitchenham is performed [54].

The following sections describe the process used for identifying relevant literature according to the stages of an SLR: (i) formulation of a search string, (ii) selection of scientific databases, and (iii) definition and application of inclusion and exclusion criteria.

SWC Seminar 2018/19 RWTH Aachen University, Germany.

2.1 Search Strings

The search string combines our two essential concepts using *AND*: (1) terms related to research methodologies and (2) terms associated with LLMs. Specific research methodologies are used in the search string because initial tests showed that the title of papers often did not include umbrella terms like *research methodology* but instead a specific automated methodology itself. The second part consists of synonyms and abbreviations of LLMs.

The second research question also mentions challenges in LLM usage in research methodologies. This has not been included in the search string because papers including this third part would have been included in a search string with only the two mentioned parts anyway.

This resulted in the following search string.

```
("research methodolog*" OR "academic research"
    OR "research syntesis" OR "research activit*"
    OR "research techniques"
    OR "research approaches"
    OR "research process" OR "research question"
    OR "research questions" OR "data analysis"
    OR "literature review" OR "systematic review"
    OR "meta-analysis" OR "interview*"
    OR "survey*" OR "focus group*"
    OR "observation*" OR "case study*"
    OR "questionnaire*" OR "experiment*"
    OR "quantitative research"
    OR "qualitative research" OR "mixed method")
AND ("LLM" OR "LLMs"
    OR "large language model"
    OR "large language models" OR "generative ai")
```

2.2 Database Selection

This study uses the following well-known scientific databases to apply the search string to. Selection was made based on the ease of access and the amount of available papers on our topic.

- IEEE Xplore Digital Library
- ACM Digital Library
- SpringerLink
- arXiv
- Scopus

Many of our terms used in the search string are widely used across academics and thus in many texts and especially abstracts. Therefore, to keep the number of results manageable, the query was performed on only the titles of the papers.

2.3 Inclusion and Exclusion Criteria

Inclusion and exclusion criteria are used to filter the studies found by the query for their usefulness in answering research questions [54]. Studies to be excluded were about general research on LLMs rather than the usage of LLMs in research disciplines. Others were not available to us and therefore also excluded. The ones included addressed at least partially one of the two research questions.

The inclusion and exclusion criteria are the following.

- IC1: The paper addresses the use of LLM-tooling for a research methodology.
- IC2: The paper explains how LLMs are applied in a research methodology.
- IC3: The paper addresses a challenge when using LLMs in research.
- EX1: The paper is about research on LLMs.
- EX2: A Generative AI other than an LLM is used.
- EX3: The paper is not available to us.

2.4 Study Selection

After applying the specified inclusion and exclusion criteria and removing all duplicates 254 papers remain, the access time to all databases was May 2025. Due to the large number of papers, each paper was only examined by one author of this paper.

Database	# Results	# Final
SpringerLink	152	17
Arxive	1046	115
IEEE	241	32
ACM	93	5
Scopus	1021	85
Total	2553	254

Table 1: Findings per database

3. RESULTS

In this section, the collected studies are categorized, summarized, and analyzed. A total of 254 relevant papers were identified and included in this review, with the majority published within the last two years, as shown in Table 1.

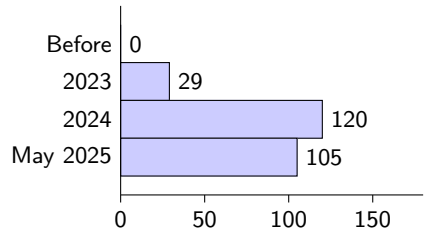


Figure 1: Number of papers published by year

To structure the findings in this paper, a categorization scheme is introduced. While a perfect categorization of LLM applications in research methodologies is inherently difficult due to overlapping boundaries and the interdisciplinary nature of research in general, we intentionally designed our sections to best reflect the diverse LLM use cases. These are also grouped in the Table 2. This table is meant to present an overview of existing LLM approaches in research methodology.

The classification is based on the 3 foundational paradigms in research methodology: qualitative, quantitative, and mixed methods [31]. These paradigms form the backbone of this paper's framework, and all identified LLM use cases are aligned accordingly. Within each of these top categories, we introduced relevant subcategories (*Literature Review* 3.1,

Meta–Analysis 3.2, *Surveys* 3.3,*Data Analysis* 3.4, *Experiments* 3.5, *Research Process* 3.6) that further specify where and how LLMs are applied in research.

This categorization aims to provide clarity but also allows for a structured presentation of LLM applications across the landscape of all methodological approaches. It allows for highlighting recurring tools, trends, and challenges unique to each methodology.

The following sections explore the subcategories in detail, illustrating the scope of LLM applications, the tasks they support, and the challenges researchers face when integrating LLMs.

3.1 Literature Review

Systematic literature reviews (& 'systematic literature surveys') are methodologies for identifying, evaluating, and synthesizing relevant research on a topic. Note that systematic literature surveys are significantly different from general *surveys*; the use of LLMs in those is described in Subsection 3.3.

Recent studies explore various uses of LLMs in automating and supporting literature reviews. We group these into six categories below.

3.1.1 Automated Review Generation

LLMs are used to generate literature reviews either in full (end-to-end) or in stages. Some work suggest that stepwise prompting improves coherence [55], while others suggest long running workflows, that integrate retrieval-augmented generation (*RAG*) to synthesize content from multiple documents [3, 6]. LLM but also word frequency-based methods are also evaluated [7].

Especially for literature surveys, LLM tooling promises automation near human performance in content and citation quality [98, 58]. To achieve this the LLM pulls papers relevant to a certain topic from a database. Using information from the identified papers, an outline, content, and finally a complete paper are generated.

3.1.2 LLM-Assisted Systematic Reviews

Many studies apply LLMs in systematic review workflows supporting stages like search, screening, data extraction, and synthesis [108, 83, 69, 85]. Multiple studies suggest *Human-in-the-loop* frameworks improve accuracy, reliability and correct mistakes [85, 105]. Two studies apply LLMs for reviews in medical domains in sentiment analysis of clinical abstracts [38, 68].

Lai et al. and Qi et al. make use of LLM-Chatbots to generate different parts of a literature survey, including the titles of sections, abstracts, and content generation [56, 79]. They describe the results generated, similar to humans, but criticize inaccuracies and hallucinations for citations in the LLM responses. To counter the hallucination problem Wen at al. make use of a different approach [100]. Here the researcher describes the research topic to the LLM, which then generates queries to retrieve scientific papers from a database. These are then summarized and correctly cited in the survey.

3.1.3 Query Generation

To assist literature reviews LLMs are used to generate Boolean queries for literature search, using structured prompts and seed studies [97, 1], with iterative query refinement and comparing results between LLM models [61].

3.1.4 Definition and Information Extraction

LLMs can help extract definitions, concepts, and structured data from scientific text [91, 110], showing high accuracy in specific domains but limited performance on embedded formats, like for example tables.

3.1.5 Evaluation and Benchmarking

Studies compare LLMs against human reviewers, revealing strengths in efficiency but weaknesses in critical analysis and citation reliability [93]. Mingye et al. suggest Multi-model consensus approaches [107], improve annotation accuracy by combining multiple LLM outputs with selective human review. While designed for code classification, the approach is also proposed for literature reviews. Tools like *Elicit* and *Scite.ai* are benchmarked across SLR tasks [89].

3.1.6 Human-AI Collaboration

Several works emphasize mixed workflows combining LLMs with human judgment [26, 86]. Applications include peer review checklist validation [37] and medical knowledge workflows [71].

In all studies, LLMs offer speed, scalability, and structural guidance but require careful prompt design and oversight to mitigate hallucinations, bias, and domain limitations. Hybrid approaches are the most effective for high-stakes review tasks.

Systems like *InteractiveSurvey* aid literature reviews by generating structured, personalized surveys [101].

3.2 Meta–Analysis

Meta-analysis is a well-known research methodology that statistically combines results from multiple studies to identify overall trends or effects. Recent work demonstrates diverse roles for LLMs in automating meta-analyses. We categorize these into five functional domains.

3.2.1 Data Extraction & Structuring

Several studies propose pipelines that use LLMs to extract structured data from scientific papers. One approach integrates GPT-3.5 with XML parsing and SQL generation [87], while another combines GPT with *SBERT* for outcome alignment across studies [88]. These methods reduce manual effort but depend heavily on input formatting and prompt design.

In [16], LLMs are used to aggregate performance metrics and evaluations across studies. The papers also show that the accuracy and reproducibility highly depends on prompt structure and human oversight.

3.2.2 Full Meta-Analysis Generation

There are so-called End-to-end systems that employ LLMs to synthesize findings and generate structured abstracts. Two studies use retrieval-augmented generation and a novel loss function to fine tune LLMs for improved meta–analysis synthesis, using data from large scientific datasets [4, 5]. A broader pipeline explores LLMs in all steps of review production, including protocol design and manuscript drafting [65]. Reason et al. automate a network meta-analysis for comparing different patient treatments by utilizing LLMs [80]. First, the data is extracted from relevant papers discussing a treatment. Then, a Python script is generated,

Main Methodology	Method	Use Case	References
Qualitative	**Survey**	Survey Creation	[50, 2, 75]
		Survey Response Simulation	[52, 47, 25, 20, 43, 21, 33, 51, 96]
		Moderation of Participants	[45, 111, 57, 27]
		Survey Analysis	[9, 49, 76, 103, 12, 32, 27]
	Literature Review	Automated Review Generation	[55, 3, 6, 7, 98, 58]
		LLM-Assisted Systematic Reviews	[108, 83, 69, 85, 105, 38, 68, 56, 79, 100]
		Query Generation	[97, 1, 61]
		Definition and Information Extraction	[91, 110]
		Evaluation and Benchmarking	[93, 107, 89]
		Human-AI Collaboration	[26, 86, 37, 71]
Quantitative	**Meta–Analysis**	Data Extraction & Structuring	[87, 88, 16]
		Full Meta–Analysis Generation	[4, 5, 65, 80]
		Study Screening	[19]
		Policy Evaluation	[10]
Mixed Methods	**Research Process**	Hypothesis Generation, Ideation Support, Experimental Design, Scientific Writing, Multimodal Content Creation, Summarization, and Peer Review Assistance	[34, 66]
		Text Processing Assistance	[48, 60, 64]
	Data Analysis	Code and Query Generation	[114, 115, 44, 104, 72, 90, 116, 42, 84, 70, 13, 18]
		Data Analysis Assistant	[102, 73, 77, 106, 92]
		Qualitative Analysis	[113, 35, 41]
	Experiments	Simulating Experiments	[63, 53, 29, 23, 62, 95, 81, 101]
		Conducting Experiments Automatically	[67, 24]

Table 2: Overview of Research Methodologies with Multiple LLM Use Cases and Corresponding References

which is used for comparing the outcomes of each of the studies. Finally, the results are summarized in a report. These systems improve efficiency but require chunking and human oversight.

3.2.3 Study Screening

LLMs have been applied to automate study inclusion decisions based on title and abstract. A rule-based prompting system (*LARS-GPT*) achieves high recall and reduces workload by 40% [19]. Performance varies with prompt quality and model choice.

3.2.4 Policy Evaluation

In non-medical domains, LLMs have been used for sentiment analysis in policy evaluation. One study processes 2300 abstracts on EU cohesion policy to generate sentiment scores [10]. Results are sensitive to textual ambiguity and bias.

3.3 Surveys

Surveys are a method to systematically collect data from a certain group of people [31]. The goal is to gather participants' beliefs and opinions related to a certain research topic [31].

LLMs show great potential in being used for surveys. This includes the creation process of surveys, the simulation of survey respondents, the moderation of participants in a survey, and finally, the analysis of the collected results. Nevertheless, hallucinations and the inability to understand complex topics limit the LLMs' usage.

3.3.1 Survey Creation

LLMs can be used to create and pretest surveys. Ke et al. developed the Behavioral Research Assistant (*BRASS*) Bot, which interacts with the researcher, helping to create surveys [50]. It allows the researcher to give definitions for desired survey topics and then generates tailored survey questions for the researcher. These can be evaluated and modified if desired. Additionally, *BRASS* allows the researcher to define different personas, representing survey participants. The LLM can answer the survey's questions, mimicking the persona created. This is not meant to represent an actual answer, but rather to test the wording and definition of the survey questions, allowing for quick modifications and adaptations to the survey.

Some surveys are also sensitive to cultural knowledge and thus can not be applied to other cultures than the addressed one [2]. Here, Adhikari et al. used the LLM to automatically adapt said surveys to specified cultures [2].

Noteworthy is that the output of the LLM always needs reviewing by the researcher or at least a human in the loop [75]. For further guidance, provide Parker et al an overview on how to integrate LLMs into the well-known interview protocol from Castillo et al. [75, 22].

3.3.2 Survey Response Simulation

There are different ways LLMs have been used to simulate responses to surveys. However, studies that did not pre-train or pre-advice the LLM found that the responses often did not align with real human responses, like for example in [52]. Here, the model tended to favor the extreme choices, which the participants did not, due to cultural reasons. Further concern is raised by [47], saying that LLMs were trained on data of people who did not consent to partake in an interview. Additionally, LLMs generalize by design and provide no interpretable depth in their answers. On top of that, most LLMs are trained on mostly Western data, leading to Western tendencies in responses.

Because of that, some studies used pretraining or provided more context for a persona in their prompts generated from actual participants, leading to better results in synthetic responses [25, 20].

LLMs have been used to simulate both individual and group responses in various survey contexts, supporting tasks like population estimation, opinion prediction, and personality profiling [43, 21, 33, 51, 96].

3.3.3 Moderation of Participants

LLMs can also be used to moderate and collect information by interacting with the participants of the surveys.

For example, [45] makes use of an LLM-based interview agent. First, the researcher selects participants and creates the survey. These participants are then called by the interviewer, who is an LLM agent. The agent communicates with the participant and asks the questions. Additionally, answers are extracted and stored, which leads to 98% correctness of the stored answers.

Other studies criticized that the LLMs were unable to ask precise follow-up questions, when an answer was unclear [57, 27].

For moderating virtual focus groups, the *Focus Agent* has been developed [111]. It can either simulate discussions or it can manage discussions with human participants. The communication with humans is done via text-to-speech and speech-to-text. Testing has shown that the synthetic discussions reflected common human viewpoints, showing no uniqueness. Usage of the synthetic discussions is therefore limited to gaining common viewpoints before the discussion with humans or to test and refine the discussed topics and questions for a discussion with humans.

Using the tool for managing an actual human discussion revealed helpfulness for the participants and kept the discussion centered around the topic. However, they advise using it as an addition to a human moderator, because the tool lacks intelligence, is viewed as a tool, and does not catch hints.

3.3.4 Survey Analysis

Once all responses are gathered, researchers need to analyze the results. Allamong et al. advise to let an LLM remove spelling mistakes first to improve the following analysis [9]. Other authors use the interview transcripts and extract the answers to each question using two LLMs [49]. First, the survey questions are identified in the transcript by the LLM, and then the answers are extracted, followed by a sentiment analysis. In a similar way, LLMs are used for thematic analyses [76, 103]. In particular, [103] tested the use of an LLM agent framework for thematic analysis in clinical interviews. Here, an expert communicates with multiple agents and can input interview transcripts. These are evaluated by the LLM and transformed into a thematic analysis, which can be fine-tuned by the researcher. This resulted in a high-quality thematic analysis in a fraction of the time a human alone would need. Balt et al. show that LLMs can also be used for (binary) classifying interview responses [12].

LLM chatbots can support interviews through summarization and theme identification, though they are criticized for lacking personal narratives and for issues like hallucinations and prompt reliability [32, 27].

LLMs have also been integrated into chatbot systems to emulate interview scenarios, allowing for more in-depth responses by dynamic follow-up questioning [27, 28]. In addition, LLMs have been used as synthetic participants, generating responses to interview-style prompts. These are noted to be plausible but often lack depth and nuance [48, 46]. GPT-3 has further been explored for the generation of synthetic qualitative datasets within HCI research contexts [40].

3.4 Data Analysis

Data analysis is the process of organizing, visualizing, and interpreting quantitative as well as qualitative data to gain insights about the data and to support suggested conclusions. Multiple papers suggest using LLMs for different research steps in data analysis, including code and query generation, tailored assistants for specific tasks, as well as qualitative data analysis tasks.

3.4.1 Code and Query Generation

Chat2Query and *TiInsight* are two LLM-powered chatbots that take the user's input and create and execute SQL queries based on the textual input. Finally, they visualize the results found [114, 115].

Multiple others discuss the use of existing chatbots, like GPT, to generate code based on the desired analysis and objective provided via text [44, 104, 72]. They achieve this by prompting, but advise to limit the usage for simple and medium data analysis tasks due to errors in code, limited correctness, and hallucinations. Jansen et al. added the option to execute the generated code inside the chat [44].

A more advanced tool is *Flowco*, which allows analysts to create a workflow for the data analysis using a simple UI [36]. Each step created is then analyzed by an LLM and turned into code, which is then executed, and the result is added to the view.

LLMs are also widely used for assisting coding tasks related to research [90, 116]. Some studies deployed LLMs for automated codebook creation and pattern recognition [42, 84]. Tool-based implementations include *QualiGPT* [109], *CHALET* [70], and *LLAMA3* for coded classification in public health interviews [13]. *CAQDAS* tools integrated LLMs for thematic visualization [18].

3.4.2 Data Analysis Assistant

One problem with using an LLM chatbot for data science tasks is that the LLM produces too much information, which leads to analysts spending extra time extracting gained insights [102]. To address this, *InsightLens* has been developed [102]. *InsightLens* is a tool consisting of multiple LLM agents. At the base lies an LLM chatbot for user interactions. Its task is to answer questions regarding the analysis of data and to create insights for the user. The other LLM agents extract the gained insights of the conversation with the LLM-Chatbot and groups them by topic and insight. The gained insights, including generated code, visualizations, and data insights, are visualized in the tool, giving the analyst a clear overview of what insights have been gained.

Nisse et al. utilize the LLM as a connector to call deeper functions and tools [73]. Here, the LLM is used to identify what the researcher would like to find out and how. Then the corresponding methods, triggering other tools, are called in the background. The result is fed to the LLM to be

explained to the user

A similar approach based on two agents is used to see if findings are consistent with current literature [77]. The first agent can be used to query concepts and ask for concept relations inside tabular data. For answering this, the LLM falls back on external tools (i.e. linear regression tools) and then presents statistical findings and their consistency with existing literature. The consistency with the literature is based on the second agent, which summarizes abstracts from a scientific database.

Another tool based on tabular data is *TablePilot* [106]. It is a prompting-based tool, but allows for recommending analysis methods to run on the table, handle user queries, provide and execute the code, and show the results. Furthermore, it can generate a complete report based on the findings from the interaction with the LLM.

An overview of further agents to be used in data science can be found in [92].

3.4.3 Qualitative Data Analysis

Most appliances in this subsection are best suited for quantitative data analysis. However, there are some specifically designed for qualitative data analysis. For example, key point extraction from textual data [113]. Here, keypoints are first generated for each data entry and then evaluated to only keep the best-matching keypoint. Both steps are performed by LLMs. Additionally, LLMs can be used to classify documents, extract information, annotate passages, and correct and summarize text [35]. Hamerlik et al. use the LLM for stance classifications on political texts. The results suggest that the LLM struggled especially with complex political texts, compared to humans [41]. The mentioned results in the survey analysis Subsection 3.3 are also about qualitative data analysis.

Another study proposes a prompt chaining technique for analyzing Reddit data, showcasing how LLMs can support qualitative annotation pipelines with reduced hallucination rates [78].

3.4.4 Critique

It is always advised to validate the data generated by the LLM [104, 30]. The result of the LLM-powered data analysis relies strongly on the model [72]. Some perform poorly. Especially for qualitative data analysis, critique was raised about sharing confidential research data with public LLMs, limited capability of LLMs to interpret complex topics, and predominant Western views in many LLMs [30].

LLM outputs were evaluated via IRR scores, precision/recall metrics, and mixed-methods user feedback [13, 14, 17, 70, 82]. These studies demonstrate both the potential and current limitations of LLMs in supporting, augmenting, or simulating key components of qualitative research workflows.

3.5 Experiment

Experiments are scientific procedures that are used to test hypotheses by observing the results and outcomes. Research investigates the use of LLMs in conducting experiments by either simulating synthetic experiment participants or by automating the experiments' execution.

3.5.1 Simulating Experiments

LLM agents have been tested as synthetic participants in different behavioral/economic experiments [63, 53]. They were used for games where multiple LLM agents could communicate with each other. Although they sometimes match human decisions, they do not manage to do so consistently, leaving room for improvement. Since in many cases the results still somewhat match with human experiments, LLMs can be used for hypothesis testing, refining experimental design, and for exploration [29]. When conducting problematic social experiments, it is not advised to use restricted models like ChatGPT since these hold back on their answers [29]. This is also backed by Chen et al., saying LLMs perform poorly on sensitive topics [23]

Lin et al. proposes LLMs to generate synthetic Electronic Health Records that fit certain statistical properties to be used in healthcare. [62].

Other studies use LLMs to generate synthetic text that reflects real-world data, by guiding it with real examples, simple labels, and post-generation filtering, this data can be used to train classifiers [95].

LLMs are also used to automate parts of research workflows, such as data processing and experiment scripting [81, 101].

3.5.2 Conducting Experiments Automatically

Two papers suggest using LLMs in automating experiments. *AILA* is used in a lab setting and automates certain experiment tasks [67]. First, the researcher creates a request, he wants to be done. The LLM agent then creates sub-tasks from that request and makes use of instrument documentation to generate code for controlling lab equipment. Once the results are there, *AILA* analyses these and reports the findings regarding the initial request. Although this works sometimes, errors still appear in the code. Therefore, the authors raise concerns regarding safety in an LLM automated lab environment.

The second study investigates the use of LLMs in Field Experiments [24]. Here, the LLM is asked to predict the result of already investigated field experiments. It gets the experiment design and three potential results, including the correct outcome, as input. It has to decide between the three options and shows an accuracy of 78%

3.6 Research Process

Research methodology guides the research process. Two existing papers already provide a comprehensive overview of the research process [34, 66], including hypothesis discovery and creation, ideation optimization, experimental design, scientific paper writing, multimodal content generation, summary creation, and peer review. These aspects will therefore not be discussed in detail here. Instead, this section highlights additional findings beyond the existing overviews.

Recent studies show that LLMs can support hypothesis generation and scientific planning through prompting strategies, multi-agent system coordination, and the integration with domain-specific tools [8, 81].

3.6.1 Text Processing Assistance

LLMs assist in summarizing literature, drafting findings, and generating theoretical constructs [48, 60]. Tools like *CoQuest* facilitate the development of research questions through iterative AI–human collaboration [64].

4. DISCUSSION

In this section, the results of the previous section are used to answer the research questions specified in the introduction.

4.1 Areas of Research Methodologies

The area of research is identified as very fast growing, as most papers have been published in the span of the last two years, as shown in Table 3. This is likely to be related to the increased availability of LLM services. Therefore, further development is to be expected soon. Partitioning the results into different areas of research methodologies revealed differences in interest regarding LLM usage. Especially in qualitative research, the interest is very high, as *Literature Reviews*, *Meta-Analyses*, and *Surveys* were the most prominent methodologies found to be using LLMs. Here they were used for collecting data, summarizing existing texts, and assisting with labeling and analyzing qualitative data. This is to be expected as LLMs show their biggest potential in text processing. In quantitative areas, *Data Analysis* was the most prominent part. Here, the LLMs were used to summarize insights and to generate code for data analysis tasks. Finally, LLMs could also be used to enhance the research process in general and for conducting experiments.

Multiple approaches have been seen on how LLMs are utilized. This includes using the LLM as a simple interactive chatbot by using prompts or general means of conversation to extract results. Others have created customized tools, where multiple LLM-Agents were used, depending on the task. These tools often made use of other technologies as well, were the main task of the LLM was to control and initiate the other technologies.

4.2 Challenges

Interestingly, most tools heavily relied on manual human work. Many mentioned that human review is always necessary, and most ideas were implemented with a human-in-the-loop approach. The reasons for that were hallucinations and inaccuracy in LLM responses.

Hallucinations

LLMs frequently generate made-up and inaccurate responses. This issue is especially bad in tasks like literature reviews and meta-analyses, where incorrect citations or unsupported claims can severely compromise research integrity.

Examples: [56], [79], [100], [94]

Lack of Depth

In qualitative research, particularly tasks like interview simulation or thematic analysis, LLMs tend to produce responses that lack personal narrative and depth. The outputs often, even though syntactically correct, miss the quality and detail that real human responses provide.

Examples: [14], [47], [103]

Ethical Concerns

Many papers mention ethical issues with LLM use that also threaten the scientific integrity of work conducted using LLMs. These issues mostly revolve around: non-consensual training data use and cultural bias (western training narrative), which raise serious concerns about inclusivity, representativeness, and data privacy when using LLMs in research settings.

Examples: [30], [47]

Accuracy

Even when LLMs produce text that appears semantically correct and well-structured, the factual correctness of the content can not be ensured. Manual human validation is often required to ensure that results are not misleading or incorrect, especially in data analysis use cases.

Examples: [102], [35], [104]

Reliability

LLMs integrated into automated pipelines, such as code generation or data extraction, create many operational issues in conjunction with error handling or downstream failures based on LLM hallucinations. Due to the still inherently undeterministic nature of LLMs, many scientific workflows using them have issues with reproducibility and trust of the results.

Examples: [36], [44], [67]

5. THREATS TO VALIDITY

While creating the search string, it was noticed that many papers do not include the term *research methodology* or similar ones in the paper. This is the reason why specific research methodologies like *systematic review* were added to the search string. Due to that finding, the possibility exists that some research methodologies were left out because their title did not match our search string.

Additionally, our queries of the database were purely based on titles, due to the terms related to research and those related to LLMs are being used in many papers' abstracts, keywords, etc. If we were to use the query on, for example, abstracts as well, the number of papers would not be manageable anymore. This might result in not including all relevant papers.

Simple Google searches reveal multiple, freely available LLM tools, which allow users to do different research activities. Since this SLR only included scientific databases, there might be some tools missing, which can only be found in gray literature.

Since this SLR investigated a very fast-growing topic, new literature might already be available.

6. CONCLUSION

This paper performed an SLR about the usage and challenges of using LLM tooling for research methodologies.

Findings show that this area of research is very new and of current interest, due to the rising number of new papers suggested in the last two years. The results presented suggest using LLMs in a variety of research methodologies, enhancing quality and efficiency. Nevertheless, researchers are advised to fact-check the LLMs' outputs, due to errors in LLM responses.

Despite current limitations, LLMs are increasingly taking on a role in the research process, and there are strong indications that they will continue to evolve and find a stable, well-defined place within scientific workflows. It is to be expected that very soon, new possibilities will arise, because of the fast rising interest in using LLMs for research. Since in some examples the LLMs are already being used as part of the scientific workflow, it becomes more and more important that usage guides are constructed and applied.

7. REFERENCES

[1] G. P. Adam, J. Deyoung, A. Paul, I. J. Saldanha, E. M. Balk, T. A. Trikalinos, and B. C. Wallace. Literature search sandbox: a large language model that generates search queries for systematic reviews. *JAMIA Open*, 7(3), 2024. Cited by: 1; All Open Access, Gold Open Access.

[2] D. M. Adhikari, V. K. Cannanure, A. Hartland, and I. Weber. Exploring llms for automated pre-testing of cross-cultural surveys. *arXiv preprint arXiv:2501.05985*, 2025.

[3] S. Agarwal, G. Sahu, A. Puri, I. H. Laradji, K. D. Dvijotham, J. Stanley, L. Charlin, and C. Pal. Litllms, llms for literature review: Are we there yet?, 2025.

[4] J. I. Ahad, R. M. Sultan, A. Kaikobad, F. Rahman, M. R. Amin, N. Mohammed, and S. Rahman. Empowering meta-analysis: Leveraging large language models for scientific synthesis. In *2024 IEEE International Conference on Big Data (BigData)*, pages 1615–1624, 2024.

[5] J. I. Ahad, R. M. Sultan, A. Kaikobad, F. Rahman, M. R. Amin, N. Mohammed, and S. Rahman. Empowering meta-analysis: Leveraging large language models for scientific synthesis, 2024.

[6] N. F. Ali, M. M. Mohtasim, S. Mosharrof, and T. G. Krishna. Automated literature review using nlp techniques and llm-based retrieval-augmented generation. In *2024 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 1–6, 2024.

[7] N. F. Ali, M. M. Mohtasim, S. Mosharrof, and T. G. Krishna. Automated literature review using nlp techniques and llm-based retrieval-augmented generation, 2024.

[8] A. K. Alkan, S. Sourav, M. Jablonska, S. Astarita, R. Chakrabarty, N. Garuda, P. Khetarpal, M. Pióro, D. Tanoglidis, K. G. Iyer, M. S. Polimera, M. J. Smith, T. Ghosal, M. Huertas-Company, S. Kruk, K. Schawinski, and I. Ciucă. A survey on hypothesis generation for scientific discovery in the era of large language models, 2025.

[9] M. B. Allamong, J. Jeong, and P. M. Kellstedt. Spelling correction with large language models to reduce measurement error in open-ended survey responses. *Research & Politics*, 12(1):20531680241311510, 2025.

[10] Z. Asatryan, C. Birkholz, and F. Heinemann. Evidence-based policy or beauty contest? an llm-based meta-analysis of eu cohesion policy evaluations. *International Tax and Public Finance*, 32(2):625–655, Dec. 2024.

[11] M. Aubin Le Quéré, H. Schroeder, C. Randazzo, J. Gao, Z. Epstein, S. T. Perrault, D. Mimno, L. Barkhuus, and H. Li. Llms as research tools: Applications and evaluations in hci data work. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA, 2024. Association for Computing Machinery.

[12] E. Balt, S. Salmi, S. Bhulai, S. Vrinzen, M. Eikelenboom, R. Gilissen, D. Creemers, A. Popma, and S. Mérelle. Deductively coding psychosocial autopsy interview data using a few-shot learning large language model. *Frontiers in Public Health*, 13:1512537, 2025.

[13] E. Balt, S. Salmi, S. Bhulai, S. Vrinzen, M. Eikelenboom, R. Gilissen, D. Creemers, A. Popma, and S. Mérelle. Deductively coding psychosocial autopsy interview data using a few-shot learning large language model. *Frontiers in Public Health*, 13, 2025. Cited by: 0; All Open Access, Gold Open Access.

[14] M. Bano, D. Zowghi, and J. Whittle. Exploring qualitative research using llms, 2023.

[15] C. F. Barros, B. B. Azevedo, V. V. G. Neto, M. Kassab, M. Kalinowski, H. A. D. do Nascimento, and M. C. G. S. P. Bandeira. Large language model for qualitative research – a systematic mapping study, 2025.

[16] C. F. Barros, B. B. Azevedo, V. V. G. Neto, M. Kassab, M. Kalinowski, H. A. D. do Nascimento, and M. C. G. S. P. Bandeira. Large language model for qualitative research – a systematic mapping study, 2025.

[17] S. Bhaduri, S. Kapoor, A. Gil, A. Mittal, and R. Mulkar. Reconciling methodological paradigms: Employing large language models as novice qualitative research assistants in talent management research, 2024.

[18] G. Bryda and A. P. Costa. Transformative technologies: Artificial intelligence and large language models in qualitative research; [tecnologias transformativas: InteligÊncia artificial e grandes modelos de linguagem na pesquisa qualitativa]; [tecnologÍas transformadoras: Inteligencia artificial y grandes modelos lingÜÍsticos en la investigaciÓn cualitativa]. *Revista Baiana de Enfermagem*, 38, 2024. Cited by: 0.

[19] X. Cai, Y. Geng, Y. Du, B. Westerman, D. Wang, C. Ma, and J. J. G. Vallejo. Utilizing large language models to select literature for meta-analysis shows workload reduction while maintaining a similar recall level as manual curation. *BMC Medical Research Methodology*, 25(1), 2025. Cited by: 0.

[20] Y. Cao, H. Liu, A. Arora, I. Augenstein, P. Röttger, and D. Hershcovich. Specializing large language models to simulate survey response distributions for global populations. *arXiv preprint arXiv:2502.07068*, 2025.

[21] Y. Cao, H. Liu, A. Arora, I. Augenstein, P. Röttger, and D. Hershcovich. Specializing large language models to simulate survey response distributions for global populations, 2025.

[22] M. Castillo-Montoya. Preparing for interview research: the interview protocol refinement framework. *Qualitative report*, 21(5), 2016.

[23] Y. Chen, Y. Hu, and Y. Lu. Simulating field experiments with large language models. *arXiv preprint arXiv:2408.09682*, 2024.

[24] Y. Chen, Y. Hu, and Y. Lu. Predicting field experiments with large language models. *arXiv preprint arXiv:2504.01167*, 2025.

[25] S. Cho, J. Kim, and J. H. Kim. Llm-based

doppelgänger models: Leveraging synthetic data for human-like responses in survey simulations. *IEEE Access*, 2024.

[26] C. Combrinck. A tutorial for integrating generative ai in mixed methods data analysis. *Discover Education*, 3(1), Aug. 2024.

[27] A. Cuevas, J. V. Scurrell, E. M. Brown, J. Entenmann, and M. I. Daepp. Collecting qualitative data at scale with large language models: A case study. *Proceedings of the ACM on Human-Computer Interaction*, 9(2):1–27, 2025.

[28] A. Cuevas, J. V. Scurrell, E. M. Brown, J. Entenmann, and M. I. G. Daepp. Collecting qualitative data at scale with large language models: A case study, 2024.

[29] Z. Cui, N. Li, and H. Zhou. Can ai replace human subjects? a large-scale replication of psychological experiments with llms. *arXiv preprint arXiv:2409.00128*, 2024.

[30] R. M. Davison, H. Chughtai, P. Nielsen, M. Marabelli, F. Iannacci, M. van Offenbeek, M. Tarafdar, M. Trenz, A. A. Techatassanasoontorn, A. Diaz Andrade, et al. The ethics of using generative ai for qualitative data analysis. *Information Systems Journal*, 34(5), 2024.

[31] K. Dehalwar and S. N. Sharma. *Fundamentals of research writing and uses of research methodologies*. Edupedia Publications Pvt Ltd, 2023.

[32] X. Ding, M. Gopannagari, K. Sun, A. Tao, D. L. Zhao, S. Varadhan, B. L. B. Hardy, D. Dalpiaz, C. Vogiatzis, L. Angrave, et al. Evaluation of llms and other machine learning methods in the analysis of qualitative survey responses for accessible engineering education research. In *2024 ASEE Annual Conference & Exposition*, 2024.

[33] R. Dominguez-Olmedo, M. Hardt, and C. Mendler-Dünner. Questioning the survey responses of large language models, 2024.

[34] S. Eger, T. Miller, et al. Transforming science with large language models: A survey on ai-assisted scientific discovery, experimentation, content generation, and evaluation. arxiv. 2025; 2502.05151. consultado el 26/04/2025.

[35] T. Fischer and C. Biemann. Exploring large language models for qualitative data analysis. In *Proceedings of the 4th International Conference on Natural Language Processing for Digital Humanities*, pages 423–437, 2024.

[36] S. N. Freund, B. Simon, E. D. Berger, and E. Jun. Flowco: Rethinking data analysis in the age of llms. *arXiv preprint arXiv:2504.14038*, 2025.

[37] A. Goldberg, I. Ullah, T. G. H. Khuong, B. K. Rachmat, Z. Xu, I. Guyon, and N. B. Shah. Usefulness of llms as an author checklist assistant for scientific papers: Neurips'24 experiment, 2024.

[38] E. Guo, M. Gupta, J. Deng, Y.-J. Park, M. Paget, and C. Naugler. Automated paper screening for clinical reviews using large language models: Data analysis study. *Journal of Medical Internet Research*, 26(1), 2024. Cited by: 53; All Open Access, Gold Open Access, Green Open Access.

[39] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

[40] P. Hämäläinen, M. Tavast, and A. Kunnari. Evaluating large language models in generating synthetic hci research data: a case study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[41] E. Hamerlik, M. Šuppa, M. Blšták, J. Kubík, M. Takáč, M. Šimko, and A. Findor. Chatgpt as your n-th annotator: Experiments in leveraging large language models for social science text annotation in slovak language. In *Proceedings of the 4th Workshop on Computational Linguistics for the Political and Social Sciences: Long and short papers*, pages 81–89, 2024.

[42] A. S. Hayes. "conversing" with qualitative data: Enhancing qualitative research through large language models (llms). *International Journal of Qualitative Methods*, 24, 2025. Cited by: 0; All Open Access, Gold Open Access, Green Open Access.

[43] C. Huang, Y. Wu, and K. Wang. Uncertainty quantification for llm-based survey simulations, 2025.

[44] J. A. Jansen, A. Manukyan, N. Al Khoury, and A. Akalin. Leveraging large language models for data analysis automation. *PloS one*, 20(2):e0317084, 2025.

[45] K. Kaiyrbekov, N. J. Dobbins, and S. D. Mooney. Automated survey collection with llm-based conversational agents. *arXiv preprint arXiv:2504.02891*, 2025.

[46] S. Kapania, W. Agnew, M. Eslami, H. Heidari, and S. Fox. 'simulacrum of stories': Examining large language models as qualitative research participants, 2024.

[47] S. Kapania, W. Agnew, M. Eslami, H. Heidari, and S. E. Fox. Simulacrum of stories: Examining large language models as qualitative research participants. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2025.

[48] S. Kapania, W. Agnew, M. Eslami, H. Heidari, and S. E. Fox. Simulacrum of stories: Examining large language models as qualitative research participants. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery.

[49] N. Kazi, I. Kahanda, S. I. Rupassara, and J. W. Kindt. Zero-shot information extraction with community-fine-tuned large language models from open-ended interview transcripts. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 932–937. IEEE, 2023.

[50] P. F. Ke and K. C. Ng. Human-ai synergy in survey development: Implications from large language models in business and research. *ACM Transactions on Management Information Systems*, 16(1):1–39, 2025.

[51] J. Kim and B. Lee. Ai-augmented surveys:

Leveraging large language models and surveys for opinion prediction, 2024.

[52] A. Kitadai, K. Ogawa, and N. Nishino. Examining the feasibility of large language models as survey respondents. In *2024 IEEE International Conference on Big Data (BigData)*, pages 3858–3864. IEEE, 2024.

[53] A. Kitadai, Y. Tsurusaki, Y. Fukasawa, and N. Nishino. Toward a novel methodology in economic experiments: Simulation of the ultimatum game with large language models. In *2023 IEEE International Conference on Big Data (BigData)*, pages 3168–3175. IEEE, 2023.

[54] B. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[55] Y. Lai, Y. Wu, Y. Wang, W. Hu, and C. Zheng. Instruct large language models to generate scientific literature survey step by step, 2024.

[56] Y. Lai, Y. Wu, Y. Wang, W. Hu, and C. Zheng. Instruct large language models to generate scientific literature survey step by step. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 484–496. Springer, 2024.

[57] M. M. Lang and S. Eskenazi. Telephone surveys meet conversational ai: Evaluating a llm-based telephone survey system at scale. *arXiv preprint arXiv:2502.20140*, 2025.

[58] X. Liang, J. Yang, Y. Wang, C. Tang, Z. Zheng, S. Song, Z. Lin, Y. Yang, S. Niu, H. Wang, et al. Surveyx: Academic survey automation via large language models. *arXiv preprint arXiv:2502.14776*, 2025.

[59] Z. Liao, M. Antoniak, I. Cheong, E. Y.-Y. Cheng, A.-H. Lee, K. Lo, J. C. Chang, and A. X. Zhang. Llms as research tools: A large scale survey of researchers' usage and perceptions. *arXiv preprint arXiv:2411.05025*, 2024.

[60] Z. Liao, M. Antoniak, I. Cheong, E. Y.-Y. Cheng, A.-H. Lee, K. Lo, J. C. Chang, and A. X. Zhang. Llms as research tools: A large scale survey of researchers' usage and perceptions, 2024.

[61] J.-L. Lieberum, M. Töws, M.-I. Metzendorf, F. Heilmeyer, W. Siemens, C. Haverkamp, D. Böhringer, J. J. Meerpohl, and A. Eisele-Metzger. Large language models for conducting systematic reviews: on the rise, but not yet ready for use—a scoping review. *Journal of Clinical Epidemiology*, 181, 2025. Cited by: 0; All Open Access, Green Open Access, Hybrid Gold Open Access.

[62] Y. Lin, Z. B. Yu, and S. Lee. A case study exploring the current landscape of synthetic medical record generation with commercial llms, 2025.

[63] H. Liu, Y. Tang, Z. Zhang, Z. Zheng, and T. Zhu. Large language model assisted experiment design with generative human-behavior agents. In *2024 Winter Simulation Conference (WSC)*, pages 2751–2762. IEEE, 2024.

[64] Y. Liu, S. Chen, H. Cheng, M. Yu, X. Ran, A. Mo, Y. Tang, and Y. Huang. Coquest: Exploring research question co-creation with an llm-based agent, 2024.

[65] X. Luo, F. Chen, D. Zhu, L. Wang, Z. Wang, H. Liu, M. Lyu, Y. Wang, Q. Wang, and Y. Chen. Potential roles of large language models in the production of systematic reviews and meta-analyses. *Journal of Medical Internet Research*, 26(1), 2024. Cited by: 13.

[66] Z. Luo, Z. Yang, Z. Xu, W. Yang, and X. Du. Llm4sr: A survey on large language models for scientific research. *arXiv preprint arXiv:2501.04306*, 2025.

[67] I. Mandal, J. Soni, M. Zaki, M. M. Smedskjaer, K. Wondraczek, L. Wondraczek, N. N. Gosvami, and N. Krishnan. Autonomous microscopy experiments through large language model agents. *arXiv preprint arXiv:2501.10385*, 2024.

[68] K. Matsui, T. Utsumi, Y. Aoki, T. Maruki, M. Takeshima, and Y. Takaesu. Human-comparable sensitivity of large language models in identifying eligible studies through title and abstract screening: 3-layer strategy using gpt-3.5 and gpt-4 for systematic reviews. *Journal of Medical Internet Research*, 26, 2024. Cited by: 5; All Open Access, Gold Open Access.

[69] L. McGinness and P. Baumgartner. Highlighting case studies in llm literature review of interdisciplinary system science, 2025.

[70] H. Meng, Y. Yang, Y. Li, J. Lee, and Y.-C. Lee. Exploring the potential of human-llm synergy in advancing qualitative analysis: A case study on mental-illness stigma, 2024.

[71] P. Mozelius and N. Humble. On the use of generative ai for literature reviews: An exploration of tools and techniques. volume 23, page 161 – 168, 2024. Cited by: 3; All Open Access, Green Open Access.

[72] M. Nejjar, L. Zacharias, F. Stiehle, and I. Weber. Llms for science: Usage for code generation and data analysis. *Journal of Software: Evolution and Process*, 37(1):e2723, 2025.

[73] D. Nissen, T. Yu, R. Babtista, and Y. Shang. Utilizing large language models (llms) in data analysis pipeline for digital phenotyping: Description, prediction, and visualization. In *2024 IEEE International Conference on Big Data (BigData)*, pages 7513–7520. IEEE, 2024.

[74] S. Noy and W. Zhang. Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654):187–192, 2023.

[75] J. L. Parker, V. M. Richard, and K. Becker. Guidelines for the integration of large language models in developing and refining interview protocols. *The Qualitative Report*, 28(12):3460–3474, 2023.

[76] M. J. Parker, C. Anderson, C. Stone, and Y. Oh. A large language model approach to educational survey feedback analysis. *International journal of artificial intelligence in education*, pages 1–38, 2024.

[77] D. Peasley, R. Kuplicki, S. Sen, M. Paulus, et al. Leveraging large language models and agent-based systems for scientific data analysis: Validation study. *JMIR Mental Health*, 12(1):e68135, 2025.

[78] M. Perkins and J. Roe. Generative ai tools in academic research: Applications and implications for qualitative and quantitative research methodologies, 2024.

[79] R. Qi, W. Li, and H. Lyu. Generation of scientific literature surveys based on large language models

(llm) and multi-agent systems (mas). In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 169–180. Springer, 2024.

[80] T. Reason, E. Benbow, J. Langham, A. Gimblett, S. L. Klijn, and B. Malcolm. Artificial intelligence to automate network meta-analyses: Four case studies to evaluate the potential application of large language models. *PharmacoEconomics-Open*, 8(2):205–220, 2024.

[81] S. Ren, P. Jian, Z. Ren, C. Leng, C. Xie, and J. Zhang. Towards scientific intelligence: A survey of llm-based scientific agents, 2025.

[82] S. Sarraf. Evaluating generative ai-enhanced content: A conceptual framework using qualitative, quantitative, and mixed-methods approaches, 2024.

[83] D. Scherbakov, N. Hubig, V. Jansari, A. Bakumenko, and L. A. Lenert. The emergence of large language models (llm) as a tool in literature reviews: an llm automated systematic review, 2024.

[84] H. Schroeder, M. A. L. Quéré, C. Randazzo, D. Mimno, and S. Schoenebeck. Large language models in qualitative research: Uses, tensions, and intentions, 2025.

[85] N. L. Schroeder, C. D. Jaldi, and S. Zhang. Large language models with human-in-the-loop validation for systematic review data extraction, 2025.

[86] G. Schryen, M. Marrone, and J. Yang. Exploring the scope of generative ai in literature review development. *Electronic Markets*, 35(1), Feb. 2025.

[87] F. Shah-Mohammadi and J. Finkelstein. Large language model-based architecture for automatic outcome data extraction to support meta-analysis. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0079–0085, 2024.

[88] F. Shah-Mohammadi and J. Finkelstein. Toward automated meta-analysis: Leveraging large language model-driven outcome alignment and data extraction. In *2024 IEEE 6th Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS)*, pages 482–487, 2024.

[89] N. Silva and D. Wickramaarachchi. Enhancing systematic literature reviews: Evaluating the performance of llm-based tools across key systematic literature review stages. In *2025 5th International Conference on Advanced Research in Computing (ICARC)*, pages 1–6, 2025.

[90] R. Sinha, I. Solola, H. Nguyen, H. Swanson, and L. Lawrence. The role of generative ai in qualitative research: Gpt-4's contributions to a grounded theory analysis. In *Proceedings of the 2024 Symposium on Learning, Design and Technology*, LDT '24, pages 17–25, New York, NY, USA, 2024. Association for Computing Machinery.

[91] T. Spinde, L. Lin, S. Hinterreiter, and I. Echizen. Leveraging large language models for automated definition extraction with taxomatic a case study on media bias, 2025.

[92] M. Sun, R. Han, B. Jiang, H. Qi, D. Sun, Y. Yuan, and J. Huang. A survey on large language model-based agents for statistics and data science. *arXiv preprint arXiv:2412.14222*, 2024.

[93] D. Tosi. Comparing generative ai literature reviews versus human-led systematic literature reviews: A case study on big data research. *IEEE Access*, 13:56210–56219, 2025.

[94] D. Tosi. Comparing generative ai literature reviews versus human-led systematic literature reviews: A case study on big data research. *IEEE Access*, 13:56210 – 56219, 2025. Cited by: 0; All Open Access, Gold Open Access.

[95] V. Veselovsky, M. H. Ribeiro, A. Arora, M. Josifoski, A. Anderson, and R. West. Generating faithful synthetic data with large language models: A case study in computational social science, 2023.

[96] P. Wang, H. Zou, H. Chen, T. Sun, Z. Xiao, and F. L. Oswald. Personality structured interview for large language model simulation in personality research, 2025.

[97] S. Wang, H. Scells, B. Koopman, and G. Zuccon. Reassessing large language model boolean query generation for systematic reviews, 2025.

[98] Y. Wang, Q. Guo, W. Yao, H. Zhang, X. Zhang, Z. Wu, M. Zhang, X. Dai, Q. Wen, W. Ye, et al. Autosurvey: Large language models can automatically write surveys. *Advances in Neural Information Processing Systems*, 37:115119–115145, 2024.

[99] Z. Wang, Z. Chu, T. V. Doan, S. Ni, M. Yang, and W. Zhang. History, development, and principles of large language models: an introductory survey. *AI and Ethics*, pages 1–17, 2024.

[100] Z. Wen, J. Cao, Z. Wang, B. Guo, R. Yang, and S. Liu. Interactivesurvey: An llm-based personalized and interactive survey paper generation system. *arXiv preprint arXiv:2504.08762*, 2025.

[101] Z. Wen, J. Cao, Z. Wang, B. Guo, R. Yang, and S. Liu. Interactivesurvey: An llm-based personalized and interactive survey paper generation system, 2025.

[102] L. Weng, X. Wang, J. Lu, Y. Feng, Y. Liu, H. Feng, D. Huang, and W. Chen. Insightlens: Augmenting llm-powered data analysis with interactive insight management and navigation. *IEEE Transactions on Visualization and Computer Graphics*, 2025.

[103] H. Xu, S. Yi, T. Lim, J. Xu, A. Well, C. Mery, A. Zhang, Y. Zhang, H. Ji, K. Pingali, et al. Tama: A human-ai collaborative thematic analysis framework using multi-agent llms for clinical interviews. *arXiv preprint arXiv:2503.20666*, 2025.

[104] J. J. Yang and S.-H. Hwang. Transforming hematological research documentation with large language models: an approach to scientific writing and data analysis. *Blood research*, 60(1):1–11, 2025.

[105] A. Ye, A. Maiti, M. Schmidt, and S. J. Pedersen. A hybrid semi-automated workflow for systematic and literature review processes with large language model analysis. *Future Internet*, 16(5), 2024. Cited by: 4; All Open Access, Gold Open Access.

[106] D. Yi, Y. Liu, L. Cao, M. Zhou, H. Dong, S. Han, and D. Zhang. Tablepilot: Recommending human-preferred tabular data analysis with large language models. *arXiv preprint arXiv:2503.13262*, 2025.

[107] M. Yuan, J. Chen, Z. Xing, G. Mohammadi, and A. Quigley. A case study of scalable content annotation using multi-llm consensus and human review, 2025.

[108] H. S. Yun, I. J. Marshall, T. A. Trikalinos, and B. C. Wallace. Appraising the potential uses and harms of llms for medical systematic reviews, 2023.

[109] H. Zhang, C. Wu, J. Xie, F. Rubino, S. Graver, C. Kim, J. M. Carroll, and J. Cai. When qualitative research meets large language model: Exploring the potential of qualigpt as a tool for qualitative coding, 2024.

[110] T. Zhang, S. Fu, D. M. Schultz, and Z. Zheng. Using large language models to produce literature reviews: Usages and systematic biases of microphysics parametrizations in 2699 publications, 2025.

[111] T. Zhang, X. Zhang, R. Cools, and A. Simeone. Focus agent: Llm-powered virtual focus group. In *Proceedings of the 24th ACM International Conference on Intelligent Virtual Agents*, pages 1–10, 2024.

[112] Y. Zhang, X. Chen, B. Jin, S. Wang, S. Ji, W. Wang, and J. Han. A comprehensive survey of scientific large language models and their applications in scientific discovery. *arXiv preprint arXiv:2406.10833*, 2024.

[113] F. Zhao, F. Yu, T. Trull, and Y. Shang. A new method using llms for keypoints generation in qualitative data analysis. In *2023 IEEE Conference on Artificial Intelligence (CAI)*, pages 333–334. IEEE, 2023.

[114] J.-P. Zhu, P. Cai, B. Niu, Z. Ni, K. Xu, J. Huang, J. Wan, S. Ma, B. Wang, D. Zhang, et al. Chat2query: A zero-shot automatic exploratory data analysis system with large language models. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 5429–5432. IEEE, 2024.

[115] J.-P. Zhu, B. Niu, P. Cai, Z. Ni, J. Wan, K. Xu, J. Huang, S. Ma, B. Wang, X. Zhou, et al. Towards automated cross-domain exploratory data analysis through large language models. *arXiv preprint arXiv:2412.07214*, 2024.

[116] T. Übellacker. Academiaos: Automating grounded theory development in qualitative research with large language models, 2024.